Contents

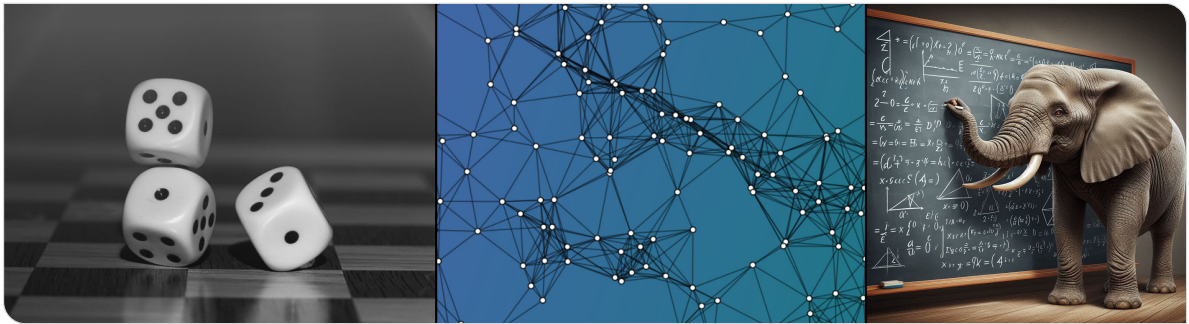
1. Basic Notions and Notation
2. The Power of Randomness
3. Important Random Variables and How to Sample Them
4. Randomised Complexity Classes
5. Probability Amplification
6. Concentration
7. Classic Hash Tables
8. Bloom Filters
9. Coupling, Balls into Bins, Poissonisation and the Poisson Point Process
10. Approximation Algorithms
11. Streaming
12. Game Theory and Yao's Principle
13. Probabilistic Method
14. Random Graphs
15. Cuckoo Hashing
16. The Peeling Algorithm
17. Retrieval

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Basic Notions and Notation

Stefan Walzer | WS 2024/2025



Basic Notions from Probability Theory

The following holds for *discrete* probability spaces.[1] Gray = typically supressed in notation.

German	English	standard notation & meaning
Ergebnismenge	sample space	set Ω
Ergebnis	outcome	element $\omega \in \Omega$
Wahrscheinlichkeitsfunktion	probability mass function	$p : \Omega \to [0,1]$ with $\sum_{\omega \in \Omega} p(\omega) = 1$
Wahrscheinlichkeitsraum	probability space	(Ω, p)
Ereignis	event	subset $E \subseteq \Omega$
Wahrscheinlichkeit	probability	$\Pr[E] = \sum_{\omega \in E} p(\omega)$ for event E
(reellwertige) Zufallsvariable	(real-valued) random variable	$X : \Omega \to \mathbb{R}$, a "random real number": "$X \geq 2$" means "$X(\omega) \geq 2$"
Erwartungswert	expectation	$\mathbb{E}[X] = \sum_{\omega \in \Omega} p(\omega) \cdot X(\omega) = \sum_x x \cdot \Pr[X = x]$
Varianz	variance	$Var(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$
bedingte Wahrscheinlichkeit	conditional probability	$\Pr[E \mid C] = \Pr[E \cap C] / \Pr[C]$
bedingter Erwartungswert	conditional expectation	$\mathbb{E}[X \mid C] = \sum_x x \cdot \Pr[X = x \mid C]$
Verteilungsfunktion	cumulative distribution function (CDF)	$x \mapsto \Pr[X \leq x]$

[1] Continuous probability spaces are more complicated. The probability mass function is often a probability density function and sums become integrals. We use continuous probability spaces informally in this lecture.

Some Calculation Rules for Probabilities

Linearity of Expectation

$$\mathbb{E}[X_1 + \ldots + X_n] \overset{\text{lin.}}{=} \mathbb{E}[X_1] + \ldots + \mathbb{E}[X_n].$$

Tail Sum Formula

If $X : \Omega \to \mathbb{N}_0$ is a random variable assuming natural numbers then

$$\mathbb{E}[X] = \sum_{i=1}^{\infty} \Pr[X \geq i].$$

Union Bound

For any events $E_1, \ldots, E_n \subseteq \Omega$

$$\Pr[E_1 \cup \ldots \cup E_n] \overset{\text{UB}}{\leq} \Pr[E_1] + \ldots + \Pr[E_n].$$

Law of Total Probability

If $E_1, \ldots, E_n \subseteq \Omega$ are disjoint events with $E_1 \cup \ldots \cup E_n = \Omega$ and F is any event then

$$\Pr[F] \overset{\text{LTP}}{=} \sum_{i=1}^{n} \Pr[F \mid E_i] \cdot \Pr[E_i].$$

Law of Total Expectation

If $E_1, \ldots, E_n \subseteq \Omega$ are disjoint events with $E_1 \cup \ldots \cup E_n = \Omega$ and X is a random variable then

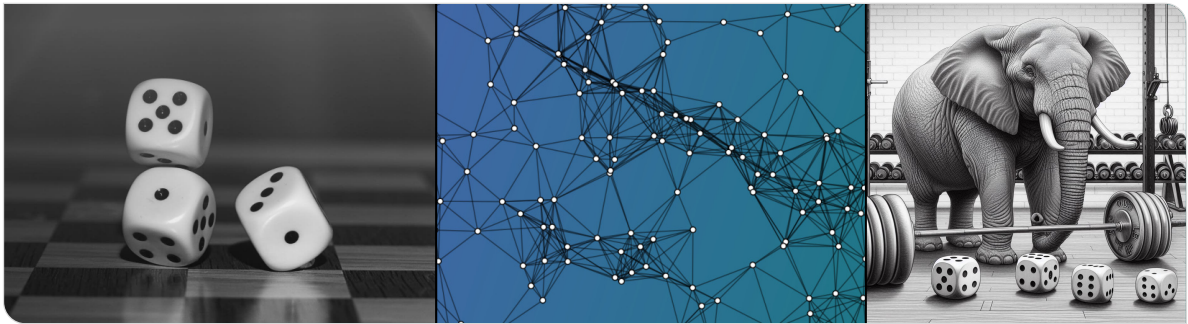
$$\mathbb{E}[X] \overset{\text{LTE}}{=} \sum_{i=1}^{n} \mathbb{E}[X \mid E_i] \cdot \Pr[E_i].$$

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – The Power of Randomness

Stefan Walzer | WS 2024/2025



Content

1. Organisation

2. The Power of Randomness
   - Improve (Worst-Case) Running Time
   - Model Performance in the Real-World – Average Case Analysis
   - Achieve Load Balancing with Pseudorandomness
   - Approximate in Sublinear Time using Random Sampling

3. Semester Outline

Organisation

Lecturer (this year + last year)

Dr. Stefan Walzer

likes randomised data structures

Lecturer (last year)

Dr. Max Katzmann

likes random geometric graphs

- lectures every Thursday, 11:30
- exercises every second Tuesday, 9:45
- Website: `https://ae.iti.kit.edu/4782.php`
- Discord Server
  - discuss exercises
  - ask questions
  - find study groups
  - report typos / mistakes

`https://discord.gg/ZQXUrQ7EPW`

- oral exam
- literature:
  - Probability and Computing (Mitzenmacher + Upfal)
  - Randomised Algorithms (Motwani + Raghavan)
  - Modern Discrete Probability (Roch)

Exercises

Organisation

- one sheet published with each lecture
- one exercises session every two weeks
  $\Rightarrow$ two sheets per exercises session
- solutions provided after the exercise session
- optional, no hand-in, no grading. *But:*
- content of sheets relevant for exam
  - you may be asked to reproduce/rediscover solutions in the exam

Recommendation

- **You should**, prior to the exercise session
  - **think about** the exerices or
  - **discuss** them in your study group.
- **You should** do **at least one of** the following
  - **solve** the exercises
  - **attend** the exercise sessions and follow along
  - **work through** the provided solutions
- **I hope** that some of you will
  - **present** your own solutions during sessions
  - **share**/**discuss** ideas on discord

Content

1. Organisation

2. The Power of Randomness
   - Improve (Worst-Case) Running Time
   - Model Performance in the Real-World – Average Case Analysis
   - Achieve Load Balancing with Pseudorandomness
   - Approximate in Sublinear Time using Random Sampling

3. Semester Outline

Can Randomness Improve (Worst-Case) Running Time?

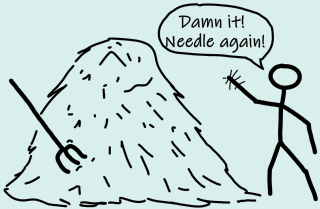
Karlsruhe Institute of Technology

it depends on what you mean by “worst case”...

Worst Input & Worst Luck

Any random decision is the worst decision.
↪ randomness is useless.

Finding Hay According to This View



Worst Input & Average Luck

Randomness *can* help. See next slide.

↑ this is what we mean in the following

In other words:

1. *We* fix a randomised algorithm.
2. *Adversary* fixes an input.
3. Random choices made independently.

Example 1: Finding an Empty Slot

Task

Input: array $A[1..n]$ where $n/2$ slots are empty
Output: $i \in [n]$ with $A[i] = \text{EMPTY}$

Observation

For any *deterministic* algorithm D there exists an input A such that D inspects $\geq n/2$ entries of A.

Observation

The randomised algorithm R that inspects slots of A at random finds an empty slot after X attempts where

$$\mathbb{E}[X] \overset{\text{TSF}}{=} \sum_{i \in \mathbb{N}_0} \Pr[X > i] = \sum_{i \in \mathbb{N}_0} 2^{-i} = 2.$$

Note

- the analysis of R holds for *any input*
- "$\mathbb{E}$" relates to choices of R (not to input)
- input is fixed *before* random choices

Example 2 and 3: Verifying Identities

Exercise: Verifying Polynomial Identities

Let f and g be two polynomial functions over a field $\mathbb{F}$. For instance:

$$f(x) = (x+1)(x-2)(x+3)(x-4)(x+5)(x-6) \text{ and } g(x) = x^6 - 7x^3 + 25.$$

Check whether $f \equiv g$ with a randomised algorithm![1]

Exercise: Verifying Matrix Identities (Freivalds' Algorithm)

Let $A, B, C \in \mathbb{F}^{n \times n}$ be matrices over the field $\mathbb{F}$. Check whether $A \cdot B = C$ with a randomised algorithm![1]

[1] The algorithm may occasionally accept incorrect identities. Precise statements on the exercise sheet.

Example 4: Evaluating Games without Draws

Three Types of Game States

$\text{value}(S) = 1 = \mathsf{W}$ // active player has winning strategy
$\text{value}(S) = 0 = \mathsf{L}$ // inactive player has winning strategy
~~$\text{value}(S) = D$ // draw in optimal play~~

Task: Evaluating a Game

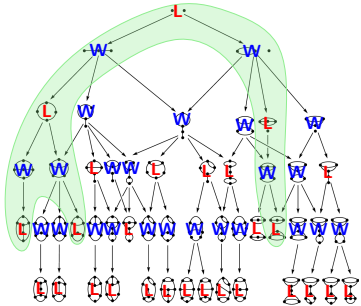
Input: (Implicit representation of) a game.
Output: value of start state.

Observation

A state S is winning if and only if some successor state is losing.

$$\text{value}(S) = \overline{\bigwedge_{S' \text{ successor of } S}} \text{value}(S').$$

Game of Sprouts (see wikipedia)



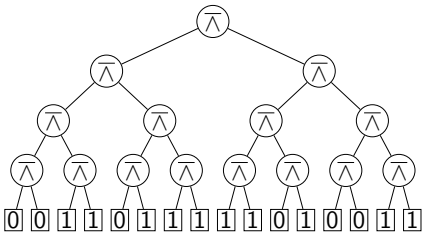
Observation

May not have to inspect entire tree to derive value at root.

Example 4 Simplified: Evaluating $\overline{\wedge}$-Trees

Problem

Input: $I \in \{0,1\}^n$ for $n = 2^d$.
Output: Value of complete binary $\overline{\wedge}$-tree with leaf values from I.
Cost Model: Number of inspected entries of I.



Exercise

For any deterministic algorithm A there exists an input $I_A \in \{0,1\}^n$ such that A inspects all n entries of I.

Our Goal

Randomised algorithm that, for any input, inspects only X entries with

$$\mathbb{E}[X] = \mathcal{O}(n^{0.793}).$$

Example 4 Simplified: Evaluating $\overline{\wedge}$-Trees

Algorithm randEval(T)**:**

- **if** $T = \text{Leaf}(b)$ **then**
  - **return** b
- $(T_0, T_1) \leftarrow T$
- // coin flip:
- sample $r \sim \mathcal{U}(\{0,1\})$
- $b_r \leftarrow \text{randEval}(T_r)$
- **if** $b_r = 0$ **then**
  - **return** 1
- **return** $1 - \text{randEval}(T_{1-r})$

Lemma

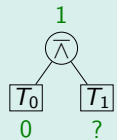
Assume randEval is excecuted for a tree T of depth $d \geq 2$. Let X be the number of resulting calls with subtrees of depth $d-2$. Then $\mathbb{E}[X] \leq 3$.

Proof.

Let $T = (T_0, T_1) = ((T_{00}, T_{01}), (T_{10}, T_{11}))$.

Case 1: $\text{value}(T) = 1$.

- Then $\text{value}(T_0) = 0$ or $\text{value}(T_1) = 0$.
- Assume (wlog) $\text{value}(T_0) = 0$.
- With probability $1/2$ we select $r = 0$ and T_1 need not be evaluated.



$$\Rightarrow \mathbb{E}[X] \leq \tfrac{1}{2} \cdot 2 + \tfrac{1}{2} \cdot 4 = 3.$$

□

Example 4 Simplified: Evaluating $\overline{\wedge}$-Trees

Algorithm randEval(T)**:**

- **if** $T = \text{Leaf}(b)$ **then**
  - **return** b
- $(T_0, T_1) \leftarrow T$
- // coin flip:
- sample $r \sim \mathcal{U}(\{0,1\})$
- $b_r \leftarrow \text{randEval}(T_r)$
- **if** $b_r = 0$ **then**
  - **return** 1
- **return** $1 - \text{randEval}(T_{1-r})$

Lemma

Assume randEval is excecuted for a tree T of depth $d \geq 2$. Let X be the number of resulting calls with subtrees of depth $d-2$. Then $\mathbb{E}[X] \leq 3$.

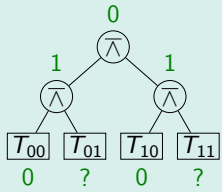
Proof.

Let $T = (T_0, T_1) = ((T_{00}, T_{01}), (T_{10}, T_{11}))$.

Case 2: $\text{value}(T) = 0$.

- Then $\text{value}(T_0) = \text{value}(T_1) = 1$.
- Like before: T_{01} and T_{11} only evaluated with probability $1/2$ each.

$$\Rightarrow \mathbb{E}[X] \leq 2 + \tfrac{1}{2} \cdot 1 + \tfrac{1}{2} \cdot 1 = 3.$$



□

Example 4 Simplified: Evaluating $\overline{\wedge}$-Trees

Algorithm randEval(T)**:**

- **if** $T = \text{Leaf}(b)$ **then**
  - **return** b
- $(T_0, T_1) \leftarrow T$
- // coin flip:
- sample $r \sim \mathcal{U}(\{0, 1\})$
- $b_r \leftarrow \text{randEval}(T_r)$
- **if** $b_r = 0$ **then**
  - **return** 1
- **return** $1 - \text{randEval}(T_{1-r})$

Lemma

Assume randEval is excecuted for a tree T of depth $d \geq 2$. Let X be the number of resulting calls with subtrees of depth $d - 2$. Then $\mathbb{E}[X] \leq 3$.

Corollary

Let T be a tree of depth $d \in \{0, 2, 4, \dots\}$, i.e. $n = 2^d$.
The number L of leafs visited by randEval(T) satisfies

$$\underbrace{\mathbb{E}[L] \leq 3^{d/2}}_{\text{proof on blackboard}} = 4^{\log_4(3^{d/2})} = 4^{d/2 \log_4(3)} = 2^{d \log_4(3)} = n^{\log_4(3)}.$$

Content

1. Organisation

2. The Power of Randomness
   - Improve (Worst-Case) Running Time
   - Model Performance in the Real-World – Average Case Analysis
   - Achieve Load Balancing with Pseudorandomness
   - Approximate in Sublinear Time using Random Sampling

3. Semester Outline

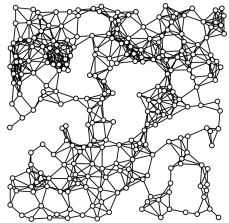
Average Case Analysis

Karlsruhe Institute of Technology

Theory-Practice Gap

SAT is NP-complete $\overset{???}{\longleftrightarrow}$ modern SAT-solvers handle relevant instances with millions of clauses

Similar observations for NP-hard graph problems on relevant graph classes, e.g. social networks.



Bridging the Gap

1. Define a distribution $\mathcal{I}$ on inputs.
   - $\mathcal{I}$ should be realistic, i.e. model real world instances
   - $\mathcal{I}$ should have simple mathematical structure
2. Show that time to solve $I \sim \mathcal{I}$ is small *in expectation*.

Goals

- model real world instances
- identify useful properties of these instances
- build algorithms exploiting these properties

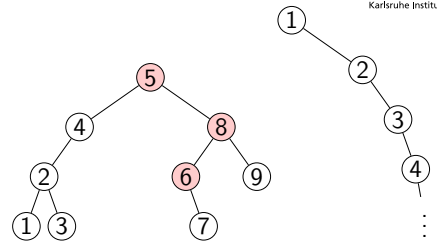
Toy Example: Unbalanced Search Trees

Setting

Inserted $1, \ldots, n$ into search tree *in some order*.
Consider: Depth of Element $y \in \{1, \ldots, n\}$.

Worst Case

Sorted order: $\text{depth}(y) = y$.



Possible Observation

- Alice sees good performance in her setting.
- Can we explain why that might be?

Average Case Analysis

1. Model: Elements of $\{1, \ldots, n\}$ are inserted in random order.
   ↪ Note: May or may not reflect Alice's setting...
2. Goal: Show that $y \in \{1, \ldots, n\}$ has expected depth $\mathcal{O}(\log n)$.
   ↪ Proved on next slide!

Toy Example: Unbalanced Search Trees – Analysis

Lemma

For any $x, y \in [n]$: $\Pr[E_{xy}] = \frac{1}{|y-x|+1}$.

Proof.

Assume wlog $x < y$.
Let v be the element of $\{x, \ldots, y\}$ inserted first.
Note: All elements of $\{x, \ldots, y\}$ are descendents of v.

Case 1: $v = x$. Then x is ancestor of y.

Case 2: $v = y$. Then y is ancestor of x.

Case 3: $v \notin \{x, y\}$. Then x is in left subtree of v and y in right subtree of v.

Hence E_{xy} occurs $\Leftrightarrow$ $x = v \Leftrightarrow$ Case 1.
Therefore: $\Pr[E_{xy}] = \Pr[\text{Case 1}] = \frac{1}{|\{x,\ldots,y\}|} = \frac{1}{y-x+1}$. □

Context

Elements $\{1, \ldots, n\}$ inserted into search tree in uniformly random order.

Definition

Event $E_{xy} = \{x \text{ is ancestor of } y\}$

// x counts as ancestor of x

Toy Example: Unbalanced Search Trees – Analysis

Lemma

For any $x, y \in [n]$: $\Pr[E_{xy}] = \frac{1}{|y-x|+1}$.

Theorem

Let $y \in [n]$ and ℓ_y the depth y. Then $\mathbb{E}[\ell_y] \leq 2\ln(n) + 2$.

Proof.

We have $\ell_y = \sum_{x \in [n]} \mathbb{1}_{E_{xy}}$. Hence:

$$\mathbb{E}[\ell_y] \overset{\text{lin.}}{=} \sum_{x \in [n]} \mathbb{E}[\mathbb{1}_{E_{xy}}] = \sum_{x \in [n]} \Pr[E_{xy}] = \sum_{x \in [n]} \frac{1}{|y-x|+1}$$

$$\leq 2 \sum_{i=1}^{n} \tfrac{1}{i} = 2 \cdot H_n \leq 2(\ln(n)+1). \qquad \square$$

Context

Elements $\{1, \ldots, n\}$ inserted into search tree in uniformly random order.

Definition

Event $E_{xy} = \{x \text{ is ancestor of } y\}$

// x counts as ancestor of x

Content

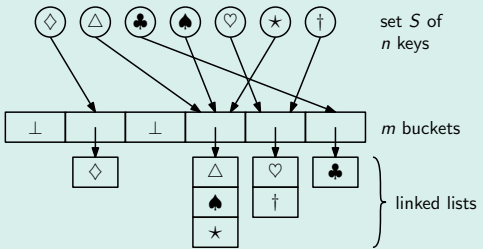
1. Organisation

2. The Power of Randomness
   - Improve (Worst-Case) Running Time
   - Model Performance in the Real-World – Average Case Analysis
   - Achieve Load Balancing with Pseudorandomness
   - Approximate in Sublinear Time using Random Sampling

3. Semester Outline

Achieve Load Balancing with Pseudorandomness

Hashing with Chaining:



Stay Tuned!

- Linear Probing
- Cuckoo Hashing
- Bloom Filters
- Retrieval
- Perfect Hashing

Content

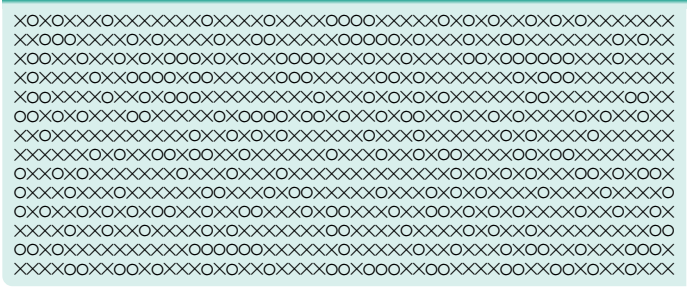
1. Organisation

2. The Power of Randomness
   - Improve (Worst-Case) Running Time
   - Model Performance in the Real-World – Average Case Analysis
   - Achieve Load Balancing with Pseudorandomness
   - Approximate in Sublinear Time using Random Sampling

3. Semester Outline

Approximate in Sublinear Time using Random Sampling

More $\times$ or more $\circ$?



Stay Tuned!

Approximation algorithms can estimate quantities by random sampling.

Content

1. Organisation

2. The Power of Randomness
   - Improve (Worst-Case) Running Time
   - Model Performance in the Real-World – Average Case Analysis
   - Achieve Load Balancing with Pseudorandomness
   - Approximate in Sublinear Time using Random Sampling

3. Semester Outline

Semester Outline

Tools from Probability Theory

- Concentration Bounds
- Random Coupling
- Yao's Principle
- Method of Bounded Differences

Random Graph Models

- Erdős-Renyi Random Graphs
- Branching Processes
- Random Geometric Graphs

Other Stuff

- Randomised Complexity Classes
- Probabilistic Method

Algorithm Design

- Random Sampling
- Approximation Algorithms
- Streaming Algorithms
- Probability Amplification

Randomised Data Structures

- Classic Hash Tables
- Cuckoo Hashing
- Bloom Filters
- Retrieval Data Structures
- Perfect Hash Functions

Conclusion

Avoiding the Worst Case with Randomness – Example: $\overline{\wedge}$-Tree Evaluation

Deterministic Algorithms:

- $\forall$Algo : $\exists$Input : Algo slow on Input.
- every algorithm is vulnerable to adversarial inputs

Our Randomised Algorithm:

- On *any* input: fast in expectation.
  on any input: slow if unlucky.
- not vulnerable to adversarial inputs

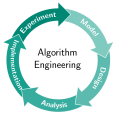
Average Case Analysis

- Model real world using probability distribution over inputs.
- In many cases random instances . . .
  - . . . are easier to solve than worst-case instances
    $\hookrightarrow$ NP-hard problems may be easy *on average*
  - . . . admit simpler algorithms and data structures
    $\hookrightarrow$ e.g. search trees with random insertion order need no load balancing

Anhang: Mögliche Prüfungsfragen I

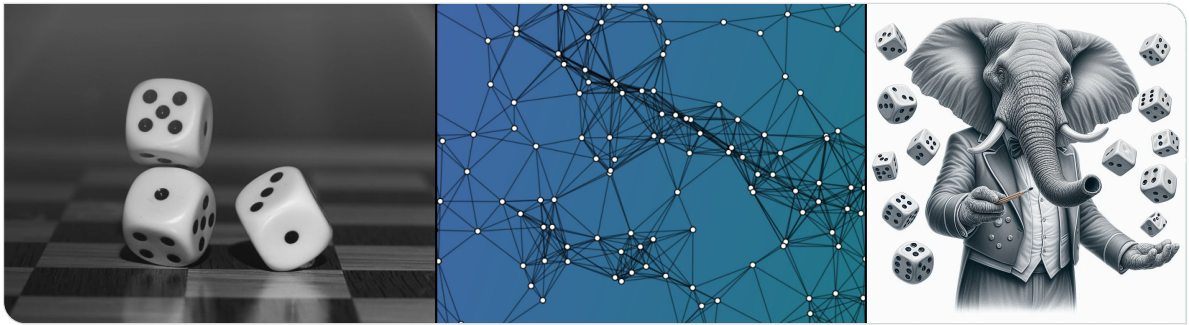
- Können wir mithilfe von Zufall Laufzeiten im Worst-Case verbessern?
  - In welchem Sinne?
  - Was ist ein Beispiel?
- Wie kann man mit einem randomisierten Algorithmus eine Polynomgleichung überprüfen?
- Wie kann man mit einem randomisierten Algorithmus ein Matrixmultiplikation überprüfen?
- In Bezug auf die Auswertung von $\overline{\wedge}$-Bäumen:
  - Was war unser Optimierungsziel?
  - Was lässt sich mit deterministischen Algorithmen erreichen?
  - Wie funktioniert unser randomisierter Ansatz?
  - Welche Laufzeit hat er und warum?
- Was ist und was soll Average Case Analyse?
- Wie verhalten sich Suchbäume bei Einfügungen in zufälliger Reihenfolge?
  - Was gilt für die erwartete Tiefe eines Knotens und warum?

Karlsruhe Institute of Technology



Probability and Computing – Important Random Variables and How to Sample Them

Stefan Walzer | WS 2024/2025



Content

1. What is Probability?

2. Bernoulli Distribution

3. Uniform Distribution

4. Rejection Sampling

5. Inverse Transform Sampling

6. Geometric Distribution

7. Sampling Without Replacement

8. Reservoir Sampling

What is a Probability?

Physical Accounts

Probabilities are persistent rates of outcomes when observing the same (random) process over and over again.

It's about **objective stuff**:

"The probability that the coin comes up heads is 50%*."*

Evidential / Bayesian Accounts

Probabilities reflect how much a rational agent believes in a proposition and about how much they are willing to bet on it.

It's about **what I subjectively know**:

"The probability that it is going to rain tomorrow is 33%*."*

See `https://en.wikipedia.org/wiki/Probability_interpretations`.
In this lecture, we use a naive notion.

Bernoulli Distribution

Definition: $Ber(p)$ for $p \in [0, 1]$

$B \sim Ber(p)$ is a random variable with

$$\Pr[B = 1] = p \text{ and } \Pr[B = 0] = 1 - p.$$

Standard Assumption: Access to Coin Flips

Algorithms have access to a sequence $B_1, B_2, \ldots \sim Ber(1/2)$ in independent uniformly random bits.

Exercise: $Ber(1/3)$ from $Ber(1/2)$

Design an algorithm that outputs B such that $B \sim Ber(1/3)$.

Uniform Distribution

Definition: $\mathcal{U}(D)$ on finite D

If $|D| < \infty$, then $X \sim \mathcal{U}(D)$ is a random variable with

$$\Pr[X = x] = \frac{1}{|D|} \text{ for all } x \in D.$$

Definition: $\mathcal{U}(D)$ on infinite D

If D is infinite but has finite measure[a] then $X \sim \mathcal{U}(D)$ is a random variable with uniform density function on D. Important example:

$$X \sim \mathcal{U}([0,1]) \Leftrightarrow \forall x \in [0,1] : \Pr[X < x] = x.$$

[a] Formal details: Not in this lecture.

Standard Assumption

Algorithms have access to $X_1, X_2, \ldots \sim \mathcal{U}([0,1])$.
In practice: Initialise the significand[a] of floating point number with random bits.

[a] Deutsch: Mantisse.

Exercise: $\mathcal{U}(\{1,\ldots,n\})$ from $\mathcal{U}([0,1])$

Design an algorithm that outputs X such that $X \sim \mathcal{U}(\{1,\ldots,n\})$.

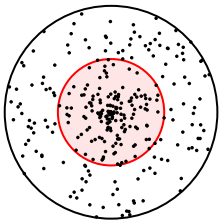
Uniform Distribution on a Disc

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Flawed Attempt

sample $\Phi \sim \mathcal{U}([0, 2\pi])$
sample $R \sim \mathcal{U}([0, 1])$
return $(R \cdot \cos \Phi, R \cdot \sin \Phi)$



Issue

Disc of half the radius is hit 50% of the time but makes up only $1/4$ of the area!

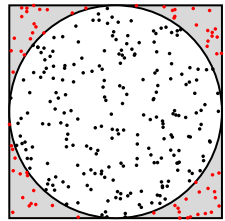
Uniform Distribution on a Disc with Rejection Sampling

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

Solution with Rejection Sampling

repeat
  sample $X \sim \mathcal{U}([-1, 1])$
  sample $Y \sim \mathcal{U}([-1, 1])$
until $X^2 + Y^2 \leq 1$
return (X, Y)



- Idea: $P \sim \mathcal{U}([-1, 1]^2)$ *conditioned* on $P \in D$ is uniform on D.
- Each sample is accepted with probability $\pi/4$.
- Expected number of rounds is $1/(\pi/4) = \mathcal{O}(1)$.

Spoiler alert: We'll get worst-case constant time with inverse transform sampling later.

Rejection Sampling in General Discrete Distributions

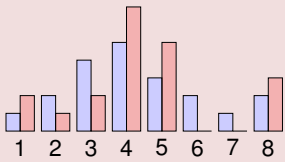
Exercise

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be distributions on a finite[a] set D. Assume

1. We can sample in constant time from $\mathcal{D}_1$.
2. There exists $C > 0$ such that for any $x \in D$ we have

$$\Pr_{X \sim \mathcal{D}_2}[X = x] \leq C \cdot \Pr_{X \sim \mathcal{D}_1}[X = x].$$

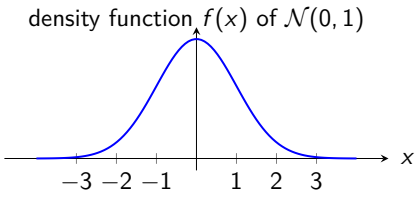
Design an algorithm that generates a sample from $\mathcal{D}_2$ in expected time $\mathcal{O}(C)$.



[a] This can be generalised.

Inverse Transform Sampling

- Let $\mathcal{D}$ be a distribution on $\mathbb{R}$.
  $\hookrightarrow$ e.g. $\mathcal{D} = \mathcal{N}(0,1)$
- Let $X \sim \mathcal{D}$ and $F_X(x) = \Pr[X \leq x]$.
  $\hookrightarrow$ F_X is the *cumulative distribution function* of X
  $\hookrightarrow$ the CDF of the normal distribution is called Φ
- Let $F_X^{-1}(u) := \inf\{x \in \mathbb{R} \mid F_X(x) \geq u\}$.
  $\hookrightarrow$ ordinary inverse for strictly monotone F_X



Theorem (Inverse Transform Sampling)

If $U \sim \mathcal{U}([0,1])$ then $F_X^{-1}(U) \overset{\mathrm{d}}{=} X$, i.e. $F_X^{-1}(U) \sim \mathcal{D}$.
("$\overset{\mathrm{d}}{=}$" means: "has the same distribution as")

Reason: $\Pr[F_X^{-1}(U) \leq x] = \Pr[U \leq F_X(x)] = F_X(x)$.

Uniform Distribution on a Disc with Inverse Transform Sampling

Task

Sample $P \sim \mathcal{U}(D)$ for $D = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$.

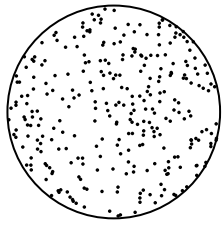
Preparation

If $(x, y) \sim \mathcal{U}(D)$ then $R = \sqrt{x^2 + y^2}$ satisfies

$$F_R(r) = \Pr[R \leq r] = r^2\pi/\pi = r^2 \text{ hence } F_R^{-1}(u) = \sqrt{u}.$$

Solution with Inverse Transform Sampling

sample $\Phi \sim \mathcal{U}([0, 2\pi])$
sample $U \sim \mathcal{U}([0, 1])$
$R \leftarrow \sqrt{U}$
return $(R \cdot \cos\Phi, R \cdot \sin\Phi)$



Geometric Distribution

Definition: $G \sim Geom_1(p)$ and $G' \sim Geom_0(p)$

Let $p \in (0, 1]$ and $B_1, B_2, \ldots \sim Ber(p)$.
Then we define the geometric random variables

$$G := \min\{i \in \mathbb{N} \mid B_i = 1\}$$

↪ number of $Ber(p)$ trials until (and including) the first success

$$G' := G - 1$$

↪ number of $Ber(p)$ failures before the first success

We write $G \sim Geom_1(p)$ and $G' \sim Geom_0(p)$.[a]

[a] In the literature *Geom* is used inconsistently.

Sampling $G \sim Geom_1(p)$ in time $\mathcal{O}(G)$

$i \leftarrow 0$
repeat
  $i \leftarrow i + 1$
  sample $X \sim Ber(p)$
until $X = 1$
return i

Quite bad: $\mathbb{E}[G] = 1/p$ might be large.

Exercise

Use inverse transform sampling to sample $G \sim Geom_1(p)$ in time $\mathcal{O}(1)$.

Sampling Without Replacement

Exercise

Design an algorithm that, given $k, n \in \mathbb{N}$ with $0 \leq k \leq n$ outputs a set $S \subseteq [n]$ of size $|S| = k$ uniformly at random.

Reservoir Sampling

Task: Maintain a fair sample of k items while reading a (possibly infinite) stream.

Algorithm init(k)**:**

allocate `reservoir`$[1..k]$

$n \leftarrow 0$

Algorithm observeItem(x)**:**

$n \leftarrow n + 1$

if $n \leq k$ **then**

`reservoir`$[n] \leftarrow x$

else

sample $I \sim \mathcal{U}(\{1, \ldots, n\})$

if $I \leq k$ **then**

`reservoir`$[I] \leftarrow x$

Theorem

Assume we call init(k) and then observeItem(x) for $x \in \{x_1, \ldots, x_n\}$ with $n \geq k$. Afterwards `reservoir` contains every subset of $\{x_1, \ldots, x_n\}$ of size k with equal probability.

Proof by induction (not here).

Example ($k = 3$)

Conclusion

General Techniques

- rejection sampling
- inverse transform sampling

Distributions

- Bernoulli distribution
- uniform distribution
- geometric distribution

Other Stuff

- sampling from a set without replacement
- reservoir sampling

Anhang: Mögliche Prüfungsfragen I

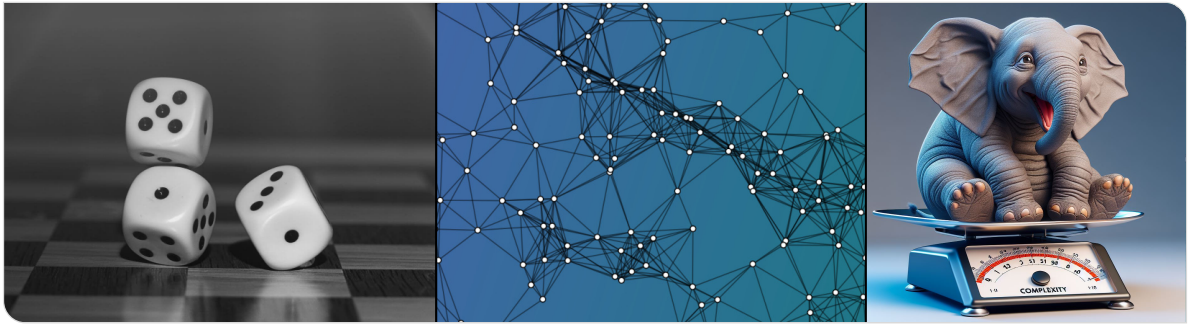
- Wie kann man $B \sim Ber(p)$ sampeln? Wie $X \sim \mathcal{U}(\{1, \ldots, n\})$? Unter welchen Annahmen?
- Wie funktioniert Rejection Sampling allgemein? Unter welchen Voraussetzungen führt Rejection Sampling zu einem effizienten Algorithmus?
- Wie funktioniert Inverse Transform Sampling allgemein? Unter welchen Voraussetzungen führt Inverse Transform Sampling zu einem effizienten Algorithmus?
- Wie kann man einen zufälligen Punkt einer Kreisscheibe sampeln? Nenne zwei Techniken und nenne Vor- bzw. Nachteile.
- Gegeben eine Menge der Größe n. Wie kann ich eine zufällige Teilmenge der Größe $k \leq n$ bestimmen und wie lange dauert das?
- Erkläre Reservoir Sampling. Ist das nicht einfach ein langsamerer Algorithmus für „Sampling without Replacement“?

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Randomised Complexity Classes

Stefan Walzer | WS 2024/2025



Lecture Notes

Lecture notes by Thomas Worsch available:

Today: Decision Problems Only

- ~~approximation algorithms~~
- ~~average case analysis~~
- ~~data structures~~
- ~~optimisation problems~~
- **decision problems**
  - for some language L such as $L = \text{PRIMES}$
  - decide for input x the question "is $x \in L$?"
  - can you do it in polynomial time?
  - does randomisation help?

Turing machines

(Non-) deterministic Turing machine

- S: finite state set
- B: finite tape alphabet including blank symbol $\square$
- $A \subseteq B - \{\square\}$: input alphabet
- one tape, one head
- transition functions
  - *deterministic:* one $\delta : S \times B \to (S \cup \{\text{YES}, \text{NO}\}) \times B \times \{-1, 0, 1\}$
  - *non-deterministic* two (or more) $\delta_0, \delta_1 : S \times B \to (S \cup \{\text{YES}, \text{NO}\}) \times B \times \{-1, 0, 1\}$ (alternatively: general transition *relation*)
  - in states YES and NO: "T halts"
- accepted language $L(T) = \{w \in A^+ \mid \exists \text{YES-computation for } w\}$

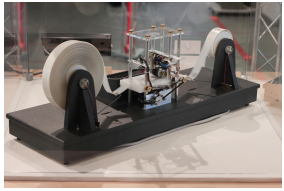


Photo: Rocky Acosta

Probabilistic Turing machine

- definition like non-deterministic TM
- uses δ_0 or δ_1 with probability $1/2$ in each step
- output $T(w)$ is random variable
- difference to NTM:
  - *quantified* non-determinism
  - can study e.g. *probability* of acceptance

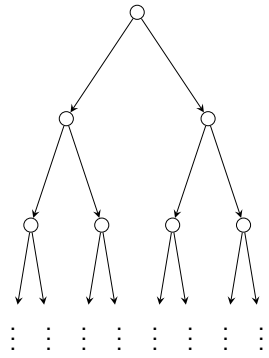
When is a PTM polynomial time?

Annoying

Running time for input x is random variable $T(x) \in \mathbb{N} \cup \{\infty\}$.

Simplification for Today: PTM in normal form

- For all inputs of length n, the PTM *halts* and does so after the *same number of steps* $t(n)$.
  $\hookrightarrow$ this is without loss of generality under weak conditions
- computation tree of PTM in normal form is complete binary tree of depth $t(n)$.
- call $t(n)$ the *running time*
- PTM runs in *polynomial time*, if $t(n) \leq p(n)$ for a polynomial $p(n)$.
- acceptance probability is the $\frac{\text{number of accepting computations}}{2^{t(n)}}$.



“Classic” Complexity Classes

class $\mathcal{C}$	requirement for $L \in \mathcal{C}$
P	polynomial time DTM can decide L
NP	polynomial time NTM can decide L
PSPACE	polynomial space TM can decide L

Complement Classes

For class $\mathcal{C}$ let $\text{co}-\mathcal{C} = \{L \mid \overline{L} \in \mathcal{C}\} = \{\overline{L} \mid L \in \mathcal{C}\}$, e.g.

- $\mathbf{P} = \text{co}-\mathbf{P}$
- $\mathbf{P} \subseteq \mathbf{NP} \cap \text{co}-\mathbf{NP}$
- relationship between **NP** and $\text{co}-\mathbf{NP}$ unknown
- $\mathbf{NP} \cup \text{co}-\mathbf{NP} \subseteq \mathbf{PSPACE}$

Polynomial time reduction from L_1 to L_2

- in polynomial time computable function $f : A^+ \to A^+$, such that
- $\forall w \in A^+$: $w \in L_1 \iff f(w) \in L_2$.

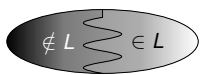
$\hookrightarrow$ then e.g. $L_2 \in \mathbf{NP}$ implies $L_1 \in \mathbf{NP}$.

Hardness

- A language H is $\mathcal{C}$-*hard*, if every language $L \in \mathcal{C}$ can be reduced to H in polynomial time.
- A language is $\mathcal{C}$-*complete*, if it is $\mathcal{C}$-hard and in $\mathcal{C}$.

Probabilistic Complexity Classes

A language L is in class **P/RP/BPP/PP**, if there exists a probabilistic polynomial time turing machine T such that...

class	name	requirement	visualisation	
P	polynomial time	$\forall w \notin L : \Pr[T(w) = \text{YES}] = 0$ $\forall w \in L : \Pr[T(w) = \text{YES}] = 1$		no error
RP	randomised polynomial time	$\forall w \notin L : \Pr[T(w) = \text{YES}] = 0$ $\forall w \in L : \Pr[T(w) = \text{YES}] \geq 1/2$		one-sided error
BPP	bounded-error probabilistic polynomial time	$\forall w \notin L : \Pr[T(w) = \text{YES}] < 1/4$ $\forall w \in L : \Pr[T(w) = \text{YES}] > 3/4$		two-sided error
PP	probabilistic polynomial time	$\forall w \notin L : \Pr[T(w) = \text{YES}] \leq 1/2$ $\forall w \in L : \Pr[T(w) = \text{YES}] > 1/2$		two-sided error

ZPP $:= \mathbf{RP} \cap \text{co}-\mathbf{RP}$. zero error probabilistic polynomial time

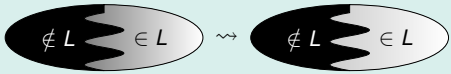
↪ requires *two* Turing machines, one for **RP**, one for co−**RP**.

We say a polynomial time PTM is an **RP**-PTM, **BPP**-PTM or **PP**-PTM if it is of the corresponding form.

Probability Amplification

Theorem

Instead of "$1/2$" we can use "$1 - 2^{-q(n)}$" in the definition of $\mathbf{RP}$ without affecting the class.



Proof.

Let T be the Turing machine witnessing $L \in \mathbf{RP}$.
By running T independently $q(n)$ times the error probability is $2^{-q(n)}$.
Running time increases by polynomial factor $q(n)$.

for $i = 1$ *to* $q(n)$ **do**
  if $T(w) =$ YES **then**
    return YES
return NO

□

Probability Amplification (2)

Theorem

Instead of "$1/4$" and "$3/4$" we can use "$2^{-q(n)}$" and "$1 - 2^{-q(n)}$" in the definition of **BPP** without affecting the class.

Proof.

Repeat $\mathcal{O}(q(n))$ times and take the majority answer.
See exercise sheet on probability amplification. □

ZPP: Zero-Error-Probabilistic Polynomial Time

Theorem: $L \in \mathbf{ZPP} \Rightarrow$ Las-Vegas Algorithm for L

If $L \in \mathbf{ZPP} := \mathbf{RP} \cap \mathrm{co}-\mathbf{RP}$ then there exists a PTM that

- decides L with no error
- has *expected* polynomial running time
  ↪ this PTM is not in normal form

Las Vegas Algorithm

Randomised Algorithm that never outputs an incorrect result.
Some definitions allow the algorithm to "give-up", reporting failure.

Proof

Let T be an $\mathbf{RP}$-PTM for L with running time $p(n)$.
↪ never errs for $x \notin L$

Let $\bar{T}$ be an $\mathbf{RP}$-PTM for $\bar{L}$ with running time $p(n)$.
↪ never errs for $x \notin \bar{L}$

repeat
  $r_1 \leftarrow T(w)$
  $r_2 \leftarrow$ **not** $\bar{T}(w)$
until $r_1 = r_2$
return r_1

- T and $\bar{T}$ never *both* answer incorrectly $\Rightarrow$ we always answer correctly.
- Every round gives $r_1 = r_2$ with probability $\geq 1/2$.

$$\mathbb{E}[\text{running time}] \leq 2p(|w|) \cdot \mathbb{E}[\#\text{rounds}] \overset{\text{TSF}}{=} 2p(|w|) \cdot \sum_{i \geq 1} \Pr[\#\text{rounds} \geq i] \leq 2p(n) \cdot \sum_{i \geq 1} 2^{-(i-1)} = 2p(n) \cdot \sum_{i \geq 0} 2^{-i} = 4p(n). \quad \square$$

Complete Problems?

Remark

The classes $\mathbf{RP}$, $\mathrm{co}-\mathbf{RP}$ and $\mathbf{BPP}$ are not believed to have complete problems unless, e.g. $\mathbf{BPP} = \mathbf{P}$.

A complete problem for $\mathbf{NP}$

$$L = \{(T, x) \mid T \text{ is an } \mathbf{NP}\text{-NTM in normal form and } T \text{ accepts } x\}$$

- L is $\mathbf{NP}$-hard ✓

  Assume $L' \in \mathbf{NP}$
  $\Rightarrow$ there exists $\mathbf{NP}$-NTM T for L' in normal form
  $\Rightarrow$ reduction: $x \in L' \Leftrightarrow (T, x) \in L$

- $L \in \mathbf{NP}$ ✓
  - check if T is $\mathbf{NP}$-NTM in normal form // $\in \mathbf{P}$
  - check if T accepts x // simulate

A complete problem for $\mathbf{RP}$?

$$L = \{(T, x) \mid T \text{ is an } \mathbf{RP}\text{-PTM in normal form and } \Pr[T \text{ accepts } x] \geq 1/2\}$$

- L is $\mathbf{RP}$-hard ✓

  Assume $L' \in \mathbf{RP}$
  $\Rightarrow$ there exists $\mathbf{RP}$-PTM T for L' in normal form
  $\Rightarrow$ reduction: $x \in L' \Leftrightarrow (T, x) \in L$

- $L \in \mathbf{RP}$ ✗
  - check if T is $\mathbf{RP}$-PTM in normal form ✗
    ⚠ **undecidable**!
  - check if T accepts x // simulate

Content

1. Prelimilaries

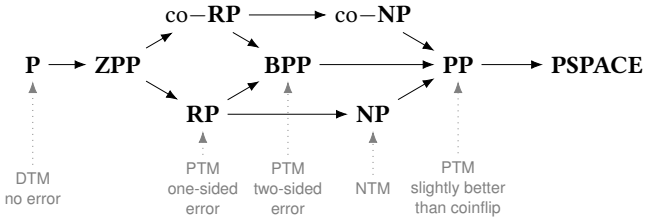
2. Probabilistic Turing Machines

3. Complexity Classes

4. Relationships between Complexity Classes

5. Conclusion

Beziehungen zwischen Komplexitätsklassen



Exercise

- $\mathbf{P} \subseteq \mathbf{ZPP}$
- $\mathbf{ZPP} \subseteq \mathbf{RP}$ and $\mathbf{ZPP} \subseteq \text{co}-\mathbf{RP}$
- $\mathbf{RP} \subseteq \mathbf{NP}$ and $\text{co}-\mathbf{RP} \subseteq \text{co}-\mathbf{NP}$
- $\mathbf{RP} \subseteq \mathbf{BPP}$ and $\text{co}-\mathbf{RP} \subseteq \mathbf{BPP}$
- $\mathbf{BPP} \subseteq \mathbf{PP}$

Following Slides

- $\mathbf{NP} \subseteq \mathbf{PP}$ and $\text{co}-\mathbf{NP} \subseteq \mathbf{PP}$
- $\mathbf{PP} \subseteq \mathbf{PSPACE}$

“Typecasting” Turing Machines

DTM as NTM

Given DTM T with transition function δ, consider NTM T' with transition functions $\delta_0 = \delta_1 = \delta$.
$\hookrightarrow$ No change in behaviour: $T(w) = \text{YES} \Leftrightarrow T'(w) = \text{YES}$.

NTM as PTM

Given NTM T, we can reinterpret it as a PTM T':

$$T(w) = \text{YES} :\Leftrightarrow \exists \text{YES-computation for } T \text{ and } w \Leftrightarrow \Pr[T'(w) = \text{YES}] > 0$$
$$T(w) = \text{NO} :\Leftrightarrow \nexists \text{YES-computation for } T \text{ and } w \Leftrightarrow \Pr[T'(w) = \text{YES}] = 0$$

PTM as DTM

Given PTM T, we can view it as DTM T' with random bitstring $b = b_1 b_2 \ldots$ as additional input.
In step i transition function δ_{b_i} is used.

$$\Pr[T(w) = \text{YES}] = \Pr_{b_1, b_2, \ldots \sim Ber(1/2)}[T'(w, b) = \text{YES}].$$

Theorem: $\mathbf{NP} \subseteq \mathbf{PP}$ (analogously $\mathrm{co}-\mathbf{NP} \subseteq \mathbf{PP}$)

i.e. show that each $L \in \mathbf{NP}$ satisfies $L \in \mathbf{PP}$

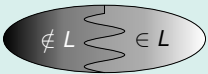
Have: NTM T certifying that $L \in \mathbf{NP}$

$w \in L \Leftrightarrow \exists$YES-computation for T and w

Use the NTM T as a PTM T':

$\forall w \notin L : \Pr[T'(w) = \text{YES}] = 0$
$\forall w \in L : \Pr[T'(w) = \text{YES}] > 0$

Want: PTM T'' certifying that $L \in \mathbf{PP}$



$\forall w \notin L : \Pr[T''(w) = \text{YES}] \leq 1/2$
$\forall w \in L : \Pr[T''(w) = \text{YES}] > 1/2$

T'' achieves this shift with a simple trick

$r \leftarrow T'(w)$ // T' is T as PTM
if $r = \text{YES}$ **then**
  return YES
else
  sample $b \sim \mathcal{U}(\{\text{YES}, \text{NO}\})$ // coinflip
  return b

Theorem: $\mathbf{PP} \subseteq \mathbf{PSPACE}$

i.e. show that each $L \in \mathbf{PP}$ satisfies $L \in \mathbf{PSPACE}$

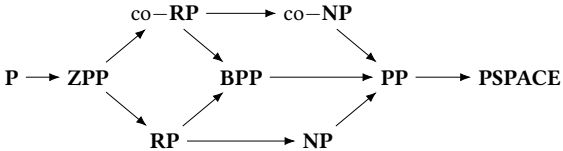
Proof

- Let T a $\mathbf{PP}$-PTM for L with running time $p(n)$.
- Consider DTM T' that simulates T for given w and random choices $b_1 b_2 \dots b_{p(n)}$.
- Consider DTM T'' that for input w runs $T'(w, b_1 b_2 \dots b_{p(n)})$ for all $2^{p(n)}$ possible $b_1 b_2 \dots b_{p(n)}$. Return YES if T' returns YES in majority of cases.
- space complexity:
  - $p(n)$ bits for counter a
  - $p(n)$ bits for $b_1, \dots, b_k$
  - $\mathcal{O}(p(n))$ space for simulating T
    (can only use $p(n)$ space in its $p(n)$ steps)

$\hookrightarrow$ T'' decides L in space $\mathcal{O}(p(n))$ (and time $\Omega(2^{p(n)})$). $\square$

$n \leftarrow |w|$
$k \leftarrow p(n)$
$a \leftarrow 0$ // k-bit counter
for $b_1 \dots b_k \leftarrow 00\dots0$ **to** $11\dots1$ **do**
  $r \leftarrow T'(w, b_1 \dots b_k)$
  if $r =$ YES **then**
    $a \leftarrow a + 1$
if $a > 2^{k-1}$ **then**
  return YES
else
  return NO

Conclusion



What we learned – not much

- Only “obvious” inclusions known
  ↪ e.g. one-sided error vs. two-sided error
- since $\mathbf{P} \stackrel{?}{=} \mathbf{PSPACE}$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
  - obviously: in $\mathrm{co}-\mathbf{NP}$.
  - 1976: in $\mathrm{co}-\mathbf{RP}$ (Rabin).
  - 1987: in $\mathbf{RP}$, hence in $\mathbf{ZPP}$ (Adleman, Huang).
  - 2002: in $\mathbf{P}$ (Agrawal, Kayal, Saxena).

A boring topic?

- People believe $\mathbf{BPP} = \mathbf{P}$
  ↪ “each $\mathbf{BPP}$ algorithm can be fully derandomised”
- $\mathbf{PP}$ is somewhat esoteric
  ↪ no interesting randomised classes remain?
- quantum computing may change the story.
  People suspect $\mathbf{NP} \not\subseteq \mathbf{BQP} \not\subseteq \mathbf{NP}$
  ↪ https://en.wikipedia.org/wiki/BQP

Anhang: Mögliche Prüfungsfragen

- Definiere: Was ist eine PTM? Was ist der Unterschied zu einer NTM?
- Definiere die Komplexitätsklassen $\mathbf{RP}$, $\mathrm{co}-\mathbf{RP}$ $\mathbf{BPP}$, $\mathbf{PP}$, $\mathbf{ZPP}$.
- Inwiefern spielen die Konstanten von $\frac{1}{2}$, $\frac{1}{4}$, $\frac{3}{4}$, die in den Definitionen vorkommen, einen Rolle? Inwiefern sind sie egal?
- Inwiefern steht die Klasse $\mathbf{ZPP}$ mit dem Konzept eines Las-Vegas Algorithmus in Verbindung? Wie sehen die Umwandlungen in die eine Richtung (Vorlesung) und in die andere Richtung (Übung) aus?
- Welche Inklusionsbeziehungen zwischen diesen Komplexitätsklassen sind bekannt?
- Begründe jede dieser Inklusionsbeziehungen. (In der tatsächlichen Prüfung würde man sich aus Zeitgründen nur eine oder zwei herausgreifen.)
- Gibt es Inklusionsbeziehungen von denen man weiß, dass sie strikt sind? Gibt es Klassen, von denen Experten vermuten, dass sie in Wirklichkeit identisch sind?

Karlsruhe Institute of Technology

Probability & Computing

Probability Amplification

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*
  - **Two-sided error**: *no bias*
- In optimization problems p is the probability of finding the optimum

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Definition: **Probability amplification** is the process of increasing the success probability of a Monte Carlo algorithm by using multiple runs.

Probability Amplification for true-biased algorithms

Exercise: For two-sided error.

- Execute independently t times.
  - If ✓at least once: Return ✓. (surely correct)
  - Otherwise: Return ✗. $\Pr[\text{“correct”}] \geq 1 - (1-p)^t \geq 1 - e^{-pt}$

$1 + x \leq e^x$ for $x \in \mathbb{R}$

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*
  - **Two-sided error**: *no bias*
- In optimization problems p is the probability of finding the optimum

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Definition: **Probability amplification** is the process of increasing the success probability of a Monte Carlo algorithm by using multiple runs.

Probability Amplification for optimization algorithms

- Execute independently t times.
  - output best result

$$\Pr[\text{“optimal”}] \geq 1 - (1-p)^t \geq 1 - e^{-pt}$$

The Segmentation Problem

Input

- Set P of points in a feature space (e.g., $\mathbb{R}^d$)
- Similarity measure $\sigma\colon P \times P \mapsto \mathbb{R}_+$

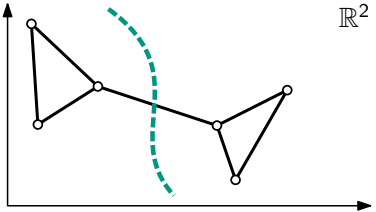
Output: $P_1, \ldots, P_k$ such that

- Points within a P_i have high similarity
- Points in distinct P_i, P_j have low similarity

Applications: Compression, medical diagnosis, etc.

Approach: Model as graph

- Each point is a node
- Edges between all node pairs, with the weight given by the similarity of the two nodes
- Find *cut-set* (edges to remove) of minimal weight such that the graph decomposes into k components.



Example

- six points in $\mathbb{R}^2$
- σ is the inversed Euclidean distance
- segment into two sets

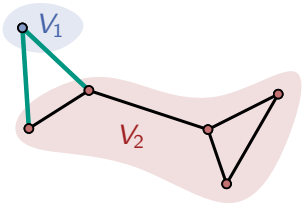
Today

$k = 2$ and $\sigma\colon P \times P \mapsto \{0, 1\}$

Computing Min Cuts

Cuts

- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of V into non-empty parts V_1, V_2 such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$.
- *Cut-set*: set of edges with an endpoints in V_1 and V_2
- *Weight of a cut*: size of the cut-set (or sum of weights in a weighted graph)



Today Goal: Compute a Min-Cut

i.e. a cut of minimum weight or cut-set of minimum size
the weight of the min-cut is known as the edge-connectivity of G

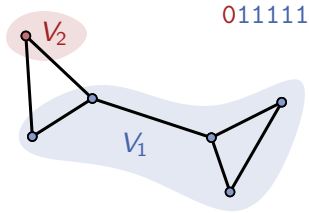
- Known deterministic strategies have worst case running time $\Omega(n^3)$.
- We'll see randomised algorithm with running time $O(n^2 \cdot \log^3(n))$.

A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$(2^n - 2)/2$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts
- Swapping parts does not yield a new partition



Algorithm: Random Cut

- Return a uniformly random cut.
- Minor challenge: How to uniformly sample cuts?
  - Represent cut using bit-string
  - Have to uniformly sample bit-string *while avoiding* `11...1` and `00...0`?
    - intution: sample from $\mathcal{U}(\{0,1\}^n)$ and use rejection sampling
    - actually for bounded running time: declare failure rather than sampling again
    - samples each cut with probability $1/2^{n-1}$

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "$=$" if there is only one min-cut.

$\rightarrow$ exponentially small!

Amplification

- Repeat the algorithm to obtain t independent random cuts, return the smallest

$$\Pr[\text{"min cut found"}] \geq 1 - \left(1 - 1/2^{n-1}\right)^t \geq 1 - e^{-t/2^{n-1}}$$

$1 + x \leq e^x$ for $x \in \mathbb{R}$

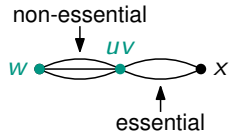
- For $t = 2^{n-1}$ min cut found with constant probability $1 - 1/e \approx 0.63$
- For $t = 2^{n-1} \cdot \ln(n)$ min cut found with high probability $1 - 1/n$

this is terrible so far...

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* (part of a min-cut) edges & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

for $i = 1$ to $n-2$ **do** // $O(n)$
  $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  $G_i = G_{i-1}$.**contract**(e) // $O(n)$
return unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

$$\begin{aligned}\Pr[\mathcal{E}_1] &= 1 - \tfrac{k}{m} \\ &\geq 1 - \tfrac{k}{nk/2} \\ &= 1 - \tfrac{2}{n}\end{aligned}$$

Observation: min-degree $\geq k$

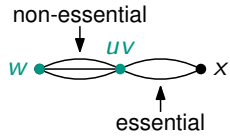
(holds for all G_i due to 1st observation)

$$m = \frac{1}{2}\sum_{v \in V} \deg(v) \geq \frac{1}{2}\sum_{v \in V} k \geq \frac{1}{2}nk$$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* (part of a min-cut) edges & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

for $i = 1$ to $n-2$ **do** // $O(n)$

  $e := \mathcal{U}(E_{i-1})$ // $O(1)$

  $G_i = G_{i-1}$.**contract**(e) // $O(n)$

return unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i" **Observation**: min-degree $\geq k$

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$ (holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

$$\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$$

$$\geq \left(\frac{\cancel{n-2}}{n}\right)\left(\frac{\cancel{n-3}}{n-1}\right)\left(\frac{\cancel{n-4}}{\cancel{n-2}}\right)\cdots\left(\frac{2}{\cancel{4}}\right)\left(\frac{1}{\cancel{3}}\right)$$

$$= \frac{2}{n(n-1)} \geq \frac{2}{n^2}$$

Karger's Algorithm Amplified

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp(-\tfrac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$$

for $t = \frac{n^2}{2} \ln(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(n^2 \log(n))$ Karger repetitions run in $O(n^4 \log(n))$ total time and return a min-cut with high probability. Much better than exp. time of Randomized Cut!

Sidenote: Number of minimum cuts

- Let $C_1, \ldots, C_\ell$ be all the min-cuts in G and $\underbrace{\mathcal{E}^i_{n-2}}$ for $i \in [\ell]$ be the event that C_i is returned by Karger's algorithm

  disjoint, since the algorithm returns only one cut

- Just seen: $\Pr[\mathcal{E}^i_{n-2}] \geq \frac{2}{n^2}$

$$1 \geq \Pr\left[\bigcup_{i \in [\ell]} \mathcal{E}^i_{n-2}\right] = \sum_{i \in [\ell]} \Pr[\mathcal{E}^i_{n-2}] \geq \frac{2 \cdot \ell}{n^2}$$

Observation: $\ell \leq \frac{n^2}{2}$.

More Amplification: Karger-Stein

Motivation

- Probability that a min-cut survives i contractions

$$\Pr[\mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}]$$
$$\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n-i+2}\right)\left(1-\frac{2}{n-i+1}\right)$$
$$= \left(\frac{\cancel{n-2}}{n}\right)\left(\frac{\cancel{n-3}}{n-1}\right)\left(\frac{\cancel{n-4}}{\cancel{n-2}}\right)\cdots\left(\frac{n-i}{\cancel{n-i+2}}\right)\left(\frac{n-i-1}{\cancel{n-i+1}}\right)$$
$$= \frac{(n-i)(n-i-1)}{n(n-1)} \geq \frac{(n-i-1)(n-i-1)}{n \cdot n} = \left(1-\frac{i+1}{n}\right)^2.$$

- Probability becomes very small only towards the very end.

KargerStein$(G_0 = (V_0, E_0))$
  if $|V_0| = 2$ **then** return unique cut-set
  for $i = 1$ to $s = |V_0| - \frac{|V_0|}{\sqrt{2}} - 1$ **do**
    $e := \mathcal{U}(E_{i-1})$
    $G_i = G_{i-1}$.**contract**(e)
$C_1 :=$ **KargerStein**(G_s) // independent runs
$C_2 :=$ **KargerStein**(G_s)
return smaller of C_1, C_2

- Idea: stop when a min-cut is still likely to exist and recurse
- After $s = n - n/\sqrt{2} - 1$ steps we have

$$\Pr[\mathcal{E}_s] \geq \left(1 - \frac{n - n/\sqrt{2}}{n}\right) = \left(1 - (1 - 1/\sqrt{2})\right)^2 = (1/\sqrt{2})^2 = \frac{1}{2}$$

Karger-Stein: Running Time

Recursion

- After $t = n - n/\sqrt{2} - 1$ steps the number of nodes is $n/\sqrt{2} + 1$

$$T(n) = 2T\left(\frac{n}{\sqrt{2}} + 1\right) + O(n^2)$$

Solution (essentially by Master Theorem)

$$T(n) = O(n^2 \log n)$$

```
KargerStein(G_0 = (V_0, E_0))
// O(1)  if |V_0| = 2 then return unique cut-set
// O(n)  for i = 1 to s = |V_0| - |V_0|/√2 - 1 do
// O(1)      e := U(E_{i-1})
// O(n)      G_i = G_{i-1}.contract(e)
C_1 := KargerStein(G_s)   // independent runs
C_2 := KargerStein(G_s)
return smaller of C_1, C_2
```

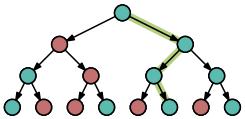
Karger-Stein: Success Probability

Know: Each call to Karger-Stein breaks the min-cut with probability at most $\frac{1}{2}$.

↑ before calling itself recursively

Auxiliary Problem

Given complete binary tree of height d where each node is randomly coloured red or green (with probability $\frac{1}{2}$ each).
What is the probability p_d that a green root-to-leaf path exists?



$p_0 = 1/2$ // root green $\quad p_d = \frac{1}{2}(1-(1-p_{d-1})^2)$ // root green, **not** no path in both left and right subtree

Claim: $p_d \geq \frac{1}{d+2}$. Proof by induction.

$$p_0 = \tfrac{1}{2} = \tfrac{1}{0+2} \quad \checkmark$$

$$p_d = \tfrac{1}{2}\left(1-(1-p_{d-1})^2\right) \geq \tfrac{1}{2}\left(1-(1-\tfrac{1}{d+1})^2\right) = \tfrac{1}{2}\left(\tfrac{2}{d+1} - \tfrac{1}{(d+1)^2}\right)$$

$$= \tfrac{1}{2}\cdot\tfrac{2d+2-1}{(d+1)^2} = \tfrac{1}{2}\cdot\tfrac{2d+1}{d^2+2d+1} \geq \tfrac{1}{2}\cdot\tfrac{2d}{d^2+2d} = \tfrac{1}{d+2} \quad \text{// for } 1 \leq a \leq b \text{ we have } \tfrac{a}{b} \geq \tfrac{a-1}{b-1}$$

Corollary: Karger-Stein succeeds with probability at least $p_{\log_{\sqrt{2}}(n)} = \frac{1}{O(\log n)}$.

Karger-Stein Amplified

Theorem: On a graph with n nodes, Karger-Stein runs in $O(n^2 \log(n))$ time and returns a minimum cut with probability at least $1/O(\log(n))$.

Amplification

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp\left(-\frac{t}{O(\log(n))}\right) = 1 - O\left(\frac{1}{n}\right) \quad \text{for } t = \log^2(n)$$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

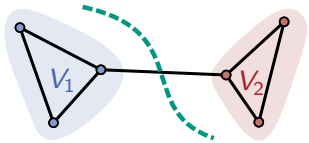
Corollary: On a graph with n nodes, $O(\log^2(n))$ repetitions of Karger-Stein run in $O(n^2 \log^3(n))$ total time and return a minimum cut with high probability.

- Compared to $O(n^4 \log(n))$ for Karger
- Compared to $\Omega(n^3)$ for deterministic approaches

Conclusion

Minimum Cut

- Fundamental graph problem
- Many deterministic flow-based algorithms ...
- ... with worst-case running times in $\Omega(n^3)$



Randomized Algorithms

- Karger's edge-contraction algorithm

Probability Amplification

- Monte Carlo algorithms with and without biases
- Repetitions amplify success probability
- Karger-Stein: Amplify before failure probability gets large

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Outlook

"Minimum cuts in near-linear time", Karger, J.Acm. '00

Success w.h.p. in time $O(m \log^3(n))$

"Faster algorithms for edge connectivity via random 2-out contractions", Ghaffari & Nowicki & Thorup, SODA'20

Success w.h.p. in time $O(m \log(n))$ and $O(m + n \log^3(n))$

Mögliche Prüfungsfragen

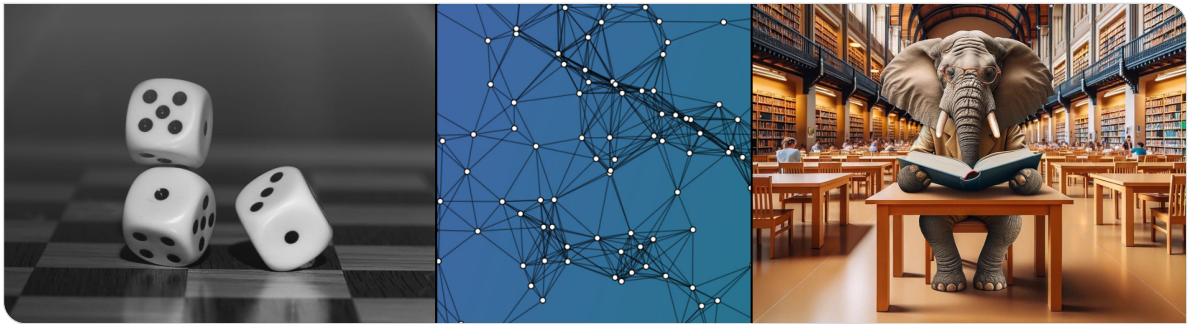
- Was ist ein Monte-Carlo-Algorithmus?
  - Welche Varianten gibt es?
- Was versteht man unter Probability Amplification?
- Wie funktioniert Probability Amplification...
  - ... bei einseitigem Fehler?
  - ... bei zweiseitigem Fehler?
  - ... bei Optimierungsproblemen?
  - Wie hängt die Fehlerwahrscheinlichkeit mit der Anzahl Wiederholungen zusammen?
- Was ist das Minimum Cut Problem?
  - Was leisten die besten bekannten deterministischen Algorithmen?
  - Was sind Erfolgswahrscheinlichkeit und Laufzeit des trivialen Random Cut Algorithmus?
  - Wie funktioniert der Algorithmus von Karger?
    - Was bedeutet $\Pr[\mathcal{E}_t]$ und wie haben wir diese Wahrscheinlichkeit abgeschätzt?
    - Was ergibt sich für die Laufzeit und die Erfolgswahrscheinlichkeit?
  - Wie ergibt sich der Algorithmus von Karger und Stein aus dem Algorithmus von Karger?
    - Wie haben wir die Erfolgswahrscheinlichkeit und Laufzeit abgeschätzt?
    - Wie erreiche ich eine Erfolgswahrscheinlichkeit von $1 - \frac{1}{n}$?

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Concentration Bounds

Stefan Walzer | WS 2024/2025



Content

1. What is a Concentration Bound?

2. Markov's Inequality

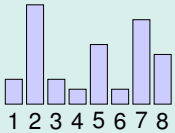
3. Chebyshev's Inequality

4. General Chernoff Bound

5. Simplified Chernoff Bounds

How to describe the shape of a real-valued distribution?

In general: Cannot be summarise in a few numbers



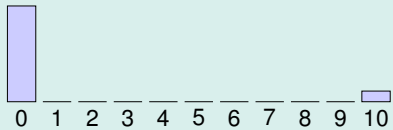
distribution on $[n]$ is given by $\vec{p} \in \mathbb{R}^n_{\geq 0}$ with $\sum_{i=1}^{n} p_i = 1$

- $n - 1$ degrees of freedom
- a few numbers convey little information

Expectation is not always meaningful

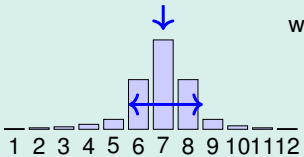
sniper "expected" to hit the target

I "expect" one hair in my soup



How to describe the shape of a real-valued distribution?

Many distributions: Unimodal, i.e. one “bump”



well summarised by

- where is the bump?
- how wide is the bump?
- how much is outside the bump?

Concentration Bound

Statement of the form:

$$\Pr[X \notin [a, b]] \leq \varepsilon.$$

Bound is strong if $b - a$ and ε are small.

Why is this relevant for us?

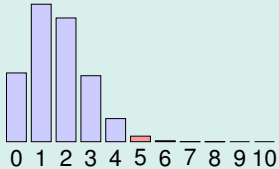
Randomised algorithm might work only if X falls into “nice” interval $[a, b]$. Want to bound failure probabily by ε.

Special Cases

Concentration around Expectation:	$\Pr[\lvert X - \mathbb{E}[X]\rvert > c] \leq \varepsilon$	$([a, b] = [\mathbb{E}[X] - c, \mathbb{E}[X] + c])$
Tail bound:	$\Pr[X > b] \leq \varepsilon$	$(a = -\infty)$

Benchmark for What Follows

Running Example



Let $X \sim Bin(10, \frac{1}{6})$ the number of sixes when rolling a fair die 10 times.

$\Pr[X \geq 5] \approx 0.016.$

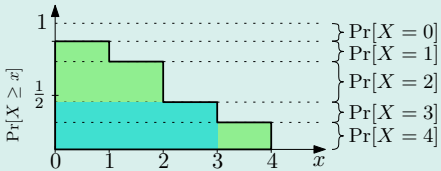
- 0.016 is calculated explicitly // only feasible for tiny examples
- later: we use general methods for obtaining bounds on $\Pr[X \geq 5]$

Markov: Tail bound for non-negative random variables

Markov Inequality

Let X be a non-negative random variable and $b \geq 0$. Then $\Pr[X \geq b] \leq \mathbb{E}[X]/b$.

Proof by Picture



From the picture ($b = 3$):
$b \cdot \Pr[X \geq b] \leq \mathbb{E}[X]$

Rearranging gives:
$\Pr[X \geq b] \leq \mathbb{E}[X]/b$.

Note: Tight if and only if $\Pr[X \in \{0, b\}] = 1$.

Actual Proof

$$\mathbb{E}[X] \overset{\text{LTE}}{=} \underbrace{\mathbb{E}[X \mid X \geq b]}_{\geq b} \cdot \Pr[X \geq b] + \underbrace{\mathbb{E}[X \mid X < b]}_{\geq 0} \cdot \Pr[X < b] \geq b \cdot \Pr[X \geq b]. \quad \square$$

Markov's Inequality: Benchmark

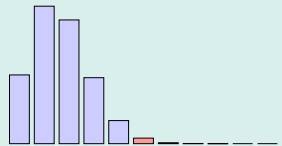
Markov Inequality

$\Pr[X \geq b] \leq \mathbb{E}[X]/b.$

Bound from Markov Inequality

$$\Pr[X \geq 5] \leq \frac{\mathbb{E}[X]}{5} = \frac{10/6}{5} \approx 0.333.$$

Running Example



Let $X \sim Bin(10, \frac{1}{6})$ the number of sixes when rolling a fair die 10 times.

$\Pr[X \geq 5] \approx 0.016.$

Chebyshev's inequality ~~Corollaries to Markov's Inequality~~

Karlsruhe Institute of Technology

Markov Inequality

Let X be a non-negative random variable and $b \geq 0$. Then $\Pr[X \geq b] \leq \mathbb{E}[X]/b$.

Corollary for non-negative X and strictly increasing $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$

$\Pr[X \geq b] = \Pr[f(X) \geq f(b)] \leq \mathbb{E}[f(X)]/f(b)$.

Corollary for possibly negative X

$\Pr[|X - \mathbb{E}X| \geq b] \leq \mathbb{E}[|X - \mathbb{E}[X]|]/b$. // but absolute values can be a pain to work with...

Chebyshev's Inequality

$\Pr[|X - \mathbb{E}X| \geq b] = \Pr[|X - \mathbb{E}X|^2 \geq b^2] \leq \mathbb{E}[|X - \mathbb{E}X|^2]/b^2 = Var(X)/b^2$.

Moments[1]

Definitions

Let X be real-valued random variable and $n \in \mathbb{N}$. Then

$\mathbb{E}[X^n]$ is the nth **(raw) moment**, $\mathbb{E}[(X - \mathbb{E}[X])^n]$ is the nth **central moment**,
$\mathbb{E}[|X|^n]$ is the nth absolute[a] moment, $\mathbb{E}[|X - \mathbb{E}[X]|^n]$ is the nth absolute central moment.

[a] Note: $|\cdot|$ makes a difference only for odd n.

What's so special about the second central moment $Var(X) := \mathbb{E}[(X - \mathbb{E}[X])^2]$?

Intuitive Meaning

It's the mean squared distance.

Exercise: Nice mathematical properties

If X, Y are independent, then
$Var(aX + bY) = a^2 \cdot Var(X) + b^2 \cdot Var(Y)$.

[1] deutsch: *das* Moment

Chebyshev's Inequality: Benchmark

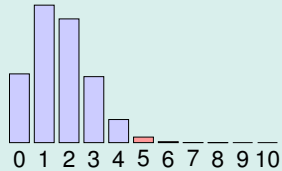
Chebyshev's Inequality

$\Pr[|X - \mathbb{E}X| \geq b] \leq Var(X)/b^2$.

Variance Calculation

- let $X_i = [i\text{th roll is a six}]$, $X = \sum_{i=1}^{10} X_i$
- $\mathbb{E}[X_i] = \frac{1}{6}$, $\mathbb{E}[X] = \frac{10}{6} = \frac{5}{3}$
- $Var(X_i) = \frac{1}{6} \cdot (\frac{5}{6})^2 + \frac{5}{6} \cdot (\frac{1}{6})^2 = \frac{5}{36}$.
- $Var(X) = 10 \cdot Var(X_1) = \frac{50}{36}$.

Running Example



Let $X \sim Bin(10, \frac{1}{6})$ the number of sixes when rolling a fair die 10 times.

$\Pr[X \geq 5] \approx 0.016$.

Bound from Chebyshev's Inequality

$\Pr[X \geq 5] = \Pr[X - \mathbb{E}[X] \geq 5 - \frac{5}{3} = \frac{10}{3}] \leq \Pr[|X - \mathbb{E}[X]| \geq \frac{10}{3}] \leq Var(X)/(\frac{10}{3})^2 = \frac{50 \cdot 9}{36 \cdot 100} = \frac{1}{8} = 0.125$.

Moment Generating Function

Definition

For real-valued random variable X call $M_X(t) = \mathbb{E}[e^{tX}]$ its *moment generating function*.

Remark: Why it's called "moment generating function".

$$M_X(t) = \mathbb{E}[e^{tX}] = \mathbb{E}\Big[\sum_{i=0}^{\infty} \frac{(tX)^i}{i!}\Big] = \sum_{i=0}^{\infty} \frac{t^i}{i!} \cdot \mathbb{E}[X^i].$$ // generating function[a] for $(\mathbb{E}[X^0], \mathbb{E}[X^1], \mathbb{E}[X^2], \dots)$

[a] https://en.wikipedia.org/wiki/Generating_function#Exponential_generating_function_(EGF)

Chernoff Bound

Let X be a real-valued RV and $b \geq 0$. Then

- $\Pr[X \geq b] \leq \inf_{t>0} \mathbb{E}[e^{tX}]/e^{tb}$ and
- $\Pr[X \leq b] \leq \inf_{t<0} \mathbb{E}[e^{tX}]/e^{tb}$.

Proof of first variant (second works simlarly)

Let $t > 0$ be arbitrary. Then $x \mapsto e^{tx}$ is increasing.

$$\Pr[X \geq b] = \Pr[e^{tX} \geq e^{tb}] \overset{\text{Markov}}{\leq} \mathbb{E}[e^{tX}]/e^{tb}.$$

Done, since $t > 0$ was arbitrary.

Generic Chernoff Bound: Benchmark

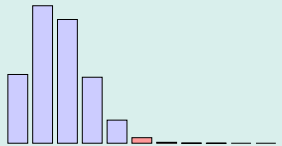
Chernoff Bound

$\Pr[X \geq b] \leq \inf_{t>0} \mathbb{E}[e^{tX}]/e^{tb}$

Calculations

- let $X_i = [i\text{th roll is a six}]$, $X = \sum_{i=1}^{10} X_i$
- $\mathbb{E}[e^{t \cdot X_i}] = \frac{1}{6} \cdot e^t + \frac{5}{6} = \frac{e^t+5}{6}$
- $\mathbb{E}[e^{t \cdot X}] = \mathbb{E}[e^{t \cdot X_1}]^{10} = (\frac{e^t+5}{6})^{10}$

Running Example



Let $X \sim Bin(10, \frac{1}{6})$ the number of sixes when rolling a fair die 10 times.

$\Pr[X \geq 5] \approx 0.016.$

Note: For independent $X_1, \ldots, X_n$

$\mathbb{E}[e^{t(X_1+\cdots+X_n)}] = \mathbb{E}[e^{tX_1} \cdot \ldots \cdot e^{tX_n}] = \mathbb{E}[e^{tX_1}] \cdot \ldots \cdot \mathbb{E}[e^{tX_n}].$

Resulting Chernoff Bound

$$\Pr[X \geq 5] \leq \inf_{t>0} \mathbb{E}[e^{tX}]/e^{5t} = \inf_{t>0} \left(\frac{e^t+5}{6}\right)^{10}/e^{5t} \overset{\text{Wolfram Alpha}}{\approx} 0.053. \quad // \text{ with } t = \ln(5)$$

What just happened?

Let $X \in \mathbb{N}$ be random variable and $p_i = \Pr[X = i]$ for $i \in \mathbb{N}$.
Consider bounds for $\Pr[X \geq b]$.

Markov

The bound is $\frac{\mathbb{E}[X]}{b}$. The contribution of "$X = i$" to is $\frac{p_i \cdot i}{b}$.

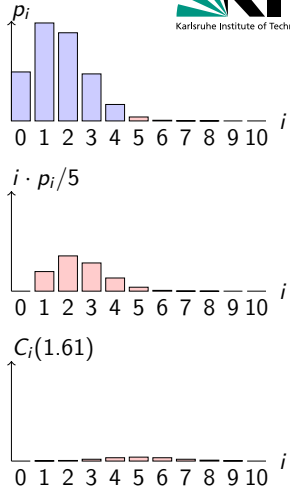
Chernoff

The bound is $\frac{\mathbb{E}[e^{tX}]}{e^{tb}}$. The contribution of "$X = i$" is $C_i(t) := \frac{p_i \cdot e^{ti}}{e^{tb}} = p_i \cdot e^{t(i-b)}$.

- if $i < b$ then $C_i(t)$ is decreasing in t
  $\hookrightarrow$ If $\Pr[X \geq b] = 0$ then Chernoff can prove this with $t \to \infty$.
- if $i > b$ then $C_i(t)$ is increasing in t
  $\hookrightarrow$ that's okay for a while as long as p_i are very small for $i > b$.

Remark: Finding the best t numerically is easy

$C_i''(t) > 0$ for all t, so $t \mapsto \mathbb{E}[e^{tX}]/e^{tb}$ is convex.



The Chernoff Bound for *your* Favourite Random Variable

Maybe someone already did the work...?

- The general Chernoff bound is hard to use
  - How to compute $\mathbb{E}[e^{tX}]$?
  - How to choose t?
- Many variants for special cases exist.

Multiplicative Form for Sums of Bernoulli RV

Theorem

Let $X = X_1 + \cdots + X_n$ with $X_i \sim Ber(p_i)$ and $\mu := \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. Then for any $\delta > 0$:

$$\Pr[X \geq (1+\delta)\mu] \leq \Bigg(\underbrace{\frac{e^\delta}{(1+\delta)^{1+\delta}}}_{f(\delta)} \Bigg)^\mu.$$

Note: $f(0) = 1$ and f is decreasing on $(0, \infty)$.
Hence for constant $\delta > 0$ the bound is exponentially small in μ.

Proof.

- $\mathbb{E}[e^{tX_i}] = p_i \cdot e^t + (1 - p_i) = 1 + p_i(e^t - 1) \leq e^{p_i(e^t-1)}$.
- $\mathbb{E}[e^{tX}] = \prod_{i=1}^{n} \mathbb{E}[e^{tX_i}] \leq \prod_{i=1}^{n} e^{p_i(e^t-1)} = e^{\sum_{i=1}^{n} p_i(e^t-1)} = e^{\mu(e^t-1)}$.
- $\Pr[X \geq (1+\delta)\mu] \overset{\text{Chernoff}}{\leq} \frac{\mathbb{E}[e^{tX}]}{e^{(1+\delta)\mu t}} \leq \frac{e^{\mu(e^t-1)}}{e^{(1+\delta)\mu t}} = \left(\frac{e^{e^t-1}}{e^{(1+\delta)t}}\right)^\mu \overset{t=\ln(1+\delta)}{=} \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$.

Consider $\ln(f(\delta)) = \delta - (1+\delta)\ln(1+\delta) =: g(\delta)$. Note that $g'(\delta) = 1 - \frac{1+\delta}{1+\delta} - \ln(1+\delta) = -\ln(1+\delta) < 0$ for $\delta > 0$.
Hence $g(\delta)$ is decreasing on $\mathbb{R}_{\geq 0}$ and so $f(\delta)$ is also decreasing on $\mathbb{R}_{\geq 0}$. □

Multiplicative Form for Sums of Bernoulli RV (ii)

The function $f(\delta) = \frac{e^\delta}{(1+\delta)^{1+\delta}}$ is still inconvenient to work with...

Theorem (from last slide, slightly generalised)

Let $X = X_1 + \cdots + X_n$ with $X_i \sim Ber(p_i)$ and $\mu := \mathbb{E}[X]$.

Then $\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$ for any $\delta \geq 0$

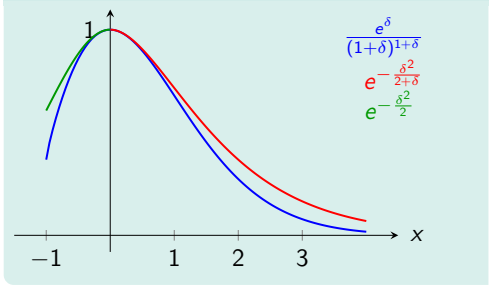
and $\Pr[X \leq (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^\mu$ for any $\delta \in [0,1)$.

Corollary (in the same setting)

- $\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta^2}{2+\delta}\mu}$ for $\delta \geq 0$,
- $\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{\delta^2}{2}\mu}$ for $\delta \in [0,1)$, and
- $\Pr[|X-\mu| \geq \delta\mu] \leq 2e^{-\frac{\delta^2}{3}\mu}$ for $0 \leq \delta \leq 1$

See also `https://en.wikipedia.org/wiki/Chernoff_bound`

"Proof by Plot"



Simplified Chernoff Bounds: Benchmark

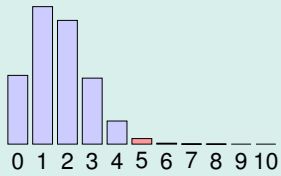
A Simplified Chernoff Bound

$\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta^2}{2+\delta}\mu}$ for $0 \leq \delta$.

Calculations

- $\mu = \mathbb{E}[X] = \frac{10}{6}$
- $5 = 3 \cdot \frac{10}{6} = \mu + 2\mu$ // $\delta = 2$

Running Example



Let $X \sim Bin(10, \frac{1}{6})$ the number of sixes when rolling a fair die 10 times.

$\Pr[X \geq 5] \approx 0.016$.

Resulting Chernoff Bound

$\Pr[X \geq 5] = \Pr[X \geq (1+2)\mu] \overset{\text{Chernoff}}{\leq} e^{-\frac{2^2}{2+2}\mu} = e^{-\frac{10}{6}} \approx 0.189$.

How do the methods compare?

Exercise

When throwing a fair die n times, bound the probability for seeing twice as many sixes as expected...

- ... using Markov,
- ... using Chebyshev,
- ... using Chernoff (simplified).

Compare the results.

Conclusion

Summary: Methods for Obtaining Concentration Bounds for real-valued X

Method	Assumptions	Formula	Challenges	$\Pr[X \geq 5]$ in Benchmark
—	—	$\Pr[X \geq b] = \sum_{i=b}^{\infty} \Pr[X = i]$	tedious calculation	0.016
Markov	$X \geq 0$	$\Pr[X \geq b] \leq \frac{\mathbb{E}[X]}{b}$	compute $\mathbb{E}[X]$	0.333
Chebyshev	—	$\Pr[\lvert X - \mathbb{E}[X]\rvert \geq b] \leq \frac{Var(X)}{b^2}$	compute $Var(X)$	0.125
Chernoff	—	$\Pr[X \geq b] \leq \inf_{t>0} \frac{\mathbb{E}[e^{tX}]}{e^{tb}}$	compute $\mathbb{E}[e^{tX}]$, choose t	0.053
simpl. Ch.	X is sum of Bernoulli	$\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\frac{\delta^2}{2+\delta}\mathbb{E}[X]}$	compute $\mathbb{E}[X]$	0.189

Further Chernoff-flavoured concentration bounds (X aggregates independent $(X_i)_{i \in \mathbb{N}}$)

Hoeffding's inequality, McDiarmid's inequality, Bernstein inequalities.

Anhang: Mögliche Prüfungsfragen I

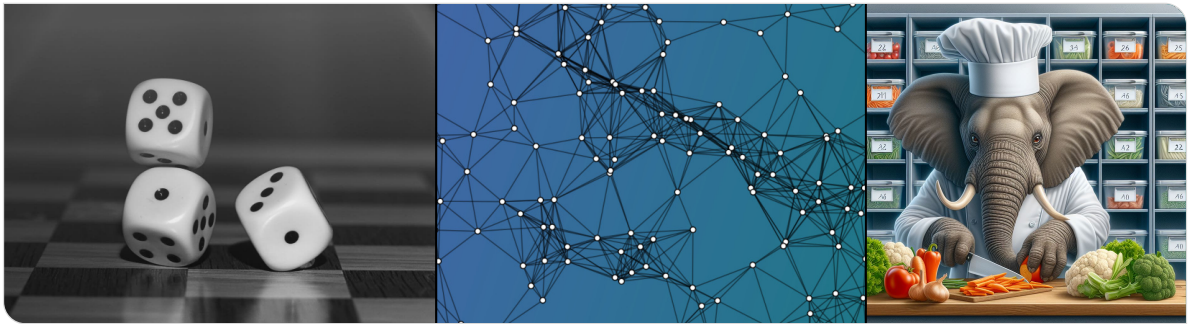
- Was versteht man unter einen Konzentrationsschranke?
- Was besagt die Markov-Ungleichung / Tschebyscheffsche Ungleichung / Chernoff-Ungleichung?
  - Was sind die jeweiligen Vorraussetzungen?
  - Wie schwierig sind die Schranken anzuwenden?
  - Wie gut oder schlecht sind die resultierenden Konzentrationsschranken im Vergleich?
- Detailfragen:
  - Beweise die Markov-Ungleichung.
  - Beweise die Tschebyscheffsche Ungleichung (aus der Markov-Ungleichung).
    - Bieten sich statt des zweiten auch höhere Momente zur Ableitung einer Ungleichung an?
  - Beweise die Chernoff-Ungleichung (aus der Markov-Ungleichung).
  - Was ist die momenterzeugende Funktion und warum heißt sie so?
  - Wie konnten wir $\mathbb{E}[e^{tX}]$ beschränken, als X Summe von Bernoulli Zufallsvariablen war?

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Classic Hash Tables

Stefan Walzer | WS 2024/2025



Content

1. Conceptions: What is a Hash Function?
- Hashing in the Wild
- What should a Theorist do?

2. Use Case 1: Hash Table with Chaining
- Using SUHA
- Using Universal Hashing

3. Use Case 2: Linear Probing
- Using SUHA
- Using Universal Hashing

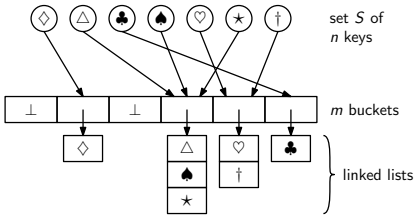
4. Conclusion

Hash Table with Chaining

e.g. `std::unordered_set`, `java.util.HashMap`

Terminology

D:	*Universe* (or domain) of *keys* (strings, integers, game states in chess)
$S \subseteq D$:	set of n keys (possibly with associated data)
$h : D \to R$:	hash function, range usually $R = [m]$
$\alpha = \frac{n}{m}$:	*load factor*, $\alpha = \mathcal{O}(1)$



Goal

Operations in time t with $\mathbb{E}[t] = \mathcal{O}(1)$.
Randomness comes from the hash function.

Ideal Hash Functions

Every function from D to R is equally likely to be h.

Ideal Hash Functions are Impractical

Naive Idea

- Let R^D denote all functions from D to R. We pick $h \sim \mathcal{U}(R^D)$.
- There are $|R|$ options for the hash of each $x \in D$
- Hence: $|R^D| = |R|^{|D|}$

$x \in D$	x_1	x_2	x_3	...	$x_{\lvert D\rvert}$
$h(x) \in R$	?	?	?	...	?

Why $h \sim \mathcal{U}(R^D)$ is desirable

- $h \sim \mathcal{U}(R^D) \Leftrightarrow \forall x_1, \ldots, x_n \in D : h(x_1), h(x_2), \ldots, h(x_n)$ are *independent* and uniformly random in R.
  $\hookrightarrow$ independence is very useful in an analysis
- In particular: $\forall x_1, \ldots, x_n \in D, \forall i_1, \ldots, i_n : \Pr_{h \sim \mathcal{U}(R^D)}[h(x_1) = i_1 \wedge \ldots \wedge h(x_n) = i_n] = |R|^{-n}$.

Why $h \sim \mathcal{U}(R^D)$ is unwieldy

$\log_2(|R|^{|D|}) = |D| \cdot \log_2(|R|)$ bits to store $h \sim \mathcal{U}(R^D)$ $\quad\rightsquigarrow\quad$ for $D = \{0,1\}^{64}$: more than 2^{64} bits.

Content

1. Conceptions: What is a Hash Function?
   - Hashing in the Wild
   - What should a Theorist do?
2. Use Case 1: Hash Table with Chaining
   - Using SUHA
   - Using Universal Hashing
3. Use Case 2: Linear Probing
   - Using SUHA
   - Using Universal Hashing
4. Conclusion

What is a Hash Function?

(it depends on who you ask)

Cryptographic Hash Function

A **collision resistant** function such as $h =$ `sha256sum`

```
$ sha256sum myfile.txt
018a7eaee8a...3e79043e21ab4  myfile.txt
```

Range $R = \{0,1\}^{256}$. It is hard to find x, y with $h(x) = h(y)$.
$\hookrightarrow$ Files with equal hashes are likely the same.

Cryptographic Pseudorandom Function

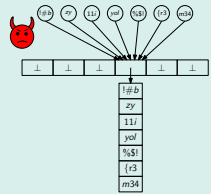
A function f : Seeds $\times D \to R$ where $\log_2 |\text{Seeds}|$ is small and no efficient algorithm can distinguish

- $f(s, \cdot)$ for $s \sim \mathcal{U}(\text{Seeds})$ and
- $h(\cdot)$ for $h \sim \mathcal{U}(R^D)$,

except with negligible probability.

Hash Function in Algorithm Engineering

- typically small range $|R| = \mathcal{O}(n)$
  $\hookrightarrow$ cannot be collision resistant
- should **behave like** $h \sim \mathcal{U}(R^D)$ **in my application**
- should be **fast** to evaluate
- adversarial settings rarely considered, although:



⚠ HashDoS is a thing.
However: Hash function and hash values need not be public.

High-Speed Hashing in Practical Data Structures

Black Magic, do not touch!

MurmurHash

Bitshifts, Magic Constants, . . .

```
uint32_t murmur3_32(const uint8_t* key,
        size_t len, uint32_t seed) {
    uint32_t h = seed;
    uint32_t k;
    for (size_t i = len >> 2; i; i--) {
        memcpy(&k, key, sizeof(uint32_t));
        key += sizeof(uint32_t);
        h ^= murmur_32_scramble(k);
        h = (h << 13) | (h >> 19);
        h = h * 5 + 0xe6546b64;
    }
    [...]
    return h;
}
static inline uint32_t murmur_32_scramble(uint32_t k) {
    k *= 0xcc9e2d51;
    k = (k << 15) | (k >> 17);
    k *= 0x1b873593;
    return k;
}
```

Usage

For $R = [m]$, pick seed $\sim \mathcal{U}(\{0,1\}^{32})$ and use

$$h(x) = \texttt{murmur3\_32}(x, \text{seed}) \bmod m.$$

(should avoid modulo in practice, see `https://github.com/lemire/fastrange`)

Does *h* behave like a random function?

- **YES,** with respect to many statistical tests.
  see `https://github.com/aappleby/smhasher`
- **NO,** HashDoS attacks are known.
  see `https://en.wikipedia.org/wiki/MurmurHash#Vulnerabilities`
- **MAYBE,** for your favourite application.

Content

1. Conceptions: What is a Hash Function?
   - Hashing in the Wild
   - What should a Theorist do?
2. Use Case 1: Hash Table with Chaining
   - Using SUHA
   - Using Universal Hashing
3. Use Case 2: Linear Probing
   - Using SUHA
   - Using Universal Hashing
4. Conclusion

What should a Theorist do?

Approach 1: Ignore the Problem

Simple Uniform Hashing Assumption (SUHA)

- We have access to $h \sim \mathcal{U}(R^D)$ for any R and D.
- h takes $\mathcal{O}(1)$ time to evaluate.
- h takes no space to store.

How to Analyse your Algorithm

1. *Assume* SUHA holds.
2. *Analyse* algorithm under SUHA.
3. *Hope* that algorithm still works with real hash functions.

SUHA is "wrong" but adequate

- *Modelling* assumption.
  $\hookrightarrow$ like e.g. ideal gas law in physics
- Excellent track record in non-adversarial settings.

What should a Theorist do?

Approach 2: Bring your own Hash Functions

Analyse Algorithm using Universal Hashing

1. Define family $\mathcal{H} \subseteq R^D$ of hash functions with $\log(|\mathcal{H}|)$ not too large.
   $\hookrightarrow$ sampling and storing $h \in \mathcal{H}$ is cheap
2. Proof that algorithm with $h \sim \mathcal{U}(\mathcal{H})$ has good expected behaviour.

Remarks

- Mathematical structure of $\mathcal{H}$ must be amenable to analysis.
- *Rigorously* covers non-adversarial settings.
- Proofs often difficult.
  $\hookrightarrow$ Wider theory practice gap than with SUHA.

What should a Theorist do?

Approach 3: Let the Cryptographers do the Work

Karlsruhe Institute of Technology

How to Analyse your Algorithm using Cryptographic Assumptions

1. Analyse algorithm under *SUHA*.
2. Actually use *cryptographic pseudorandom function f*.
   - **Case 1:** Everything still works. Great! :-)
   - **Case 2:** Something fails.
     $\Rightarrow$ Your use case can tell the difference between f and true randomness.
     $\hookrightarrow$ The cryptographers said this is impossible. $\lightning$

Should we use cryptographic pseudorandom functions?

- **YES.** Algorithms become robust even in some adversarial settings.
  $\hookrightarrow$ e.g. Python, Haskell, Ruby, Rust use **SipHash** by default
  `https://en.wikipedia.org/wiki/SipHash`
- **NO.** Too slow in high-performance settings.

Hash Function	MiB / sec
SipHash	944
Murmur3F	7623
xxHash64	12109

(source: `https://github.com/rurban/smhasher`)

Content

1. Conceptions: What is a Hash Function?
   - Hashing in the Wild
   - What should a Theorist do?

2. Use Case 1: Hash Table with Chaining
   - Using SUHA
   - Using Universal Hashing

3. Use Case 2: Linear Probing
   - Using SUHA
   - Using Universal Hashing

4. Conclusion

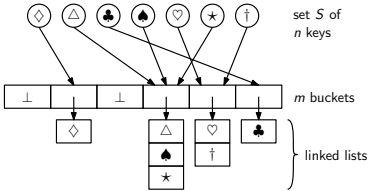
Hash Table with Chaining

Search Time under Chaining

For $n, m \in \mathbb{N}$ and a family $\mathcal{H} \subseteq [m]^D$ of hash functions the *maximum expected search time* is at most

$$T_{\text{chaining}}(n, m, \mathcal{H}) = \max_{\substack{S \subseteq D \\ |S|=n}} \max_{x \in D} \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})} \Big[1 + |\{y \in S \mid h(y) = h(x)\}| \Big]$$

⚠ Key set is *worst case*. Only $h \in \mathcal{H}$ is random. Key set is fixed *before* h is chosen.



Theorem: Hash Table with Chaining under SUHA

If $\mathcal{H} = [m]^D$ then $T_{\text{chaining}}(n, m, \mathcal{H}) \leq 2 + \alpha = \mathcal{O}(1)$ if $\alpha \in \mathcal{O}(1)$.

Analysis of Hash Table with Chaining under SUHA

Theorem: Hash Table with Chaining under SUHA

Let $\mathcal{H} = [m]^D$, $S \subseteq D$ with $|S| = n$ and $x \in D$ then

$$\mathbb{E}_{h\sim\mathcal{U}(\mathcal{H})}\Big[1 + |\{y \in S \mid h(y) = h(x)\}|\Big] \leq 2 + \alpha$$

Proof.

$$\begin{aligned}
&\mathbb{E}_{h\sim\mathcal{U}(\mathcal{H})}\Big[1 + |\{y \in S \mid h(y) = h(x)\}|\Big] \\
&= \mathbb{E}_{h\sim\mathcal{U}(\mathcal{H})}\Big[1 + \sum_{y\in S}[h(y) = h(x)]\Big] \\
&= 1 + \sum_{y\in S}\mathbb{E}_{h\sim\mathcal{U}(\mathcal{H})}\Big[[h(y) = h(x)]\Big] \\
&= 1 + \sum_{y\in S}\Pr_{h\sim\mathcal{U}(\mathcal{H})}[h(y) = h(x)] \\
&\leq 1 + 1 + \sum_{y\in S\setminus\{x\}}\Pr_{h\sim\mathcal{U}(\mathcal{H})}[h(y) = h(x)] \\
&= 2 + \sum_{y\in S\setminus\{x\}}\frac{1}{m} \leq 2 + \frac{n}{m} = 2 + \alpha. \quad \square
\end{aligned}$$

Content

1. Conceptions: What is a Hash Function?
   - Hashing in the Wild
   - What should a Theorist do?

2. Use Case 1: Hash Table with Chaining
   - Using SUHA
   - Using Universal Hashing

3. Use Case 2: Linear Probing
   - Using SUHA
   - Using Universal Hashing

4. Conclusion

A Universal Hash Family

Definition: c-universal hash family

A class $\mathcal{H} \subseteq [m]^D$ is called *c-universal* if: $\forall x \neq y \in D : \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x) = h(y)] \leq \frac{c}{m}$.

Note: $\mathcal{H} = [m]^D$ is 1-universal.

Reminder (?): Finite Fields

Let $\mathbb{F}_p = \{0, \ldots, p-1\}$ for a prime number p. Then $(\mathbb{F}_p, \times, \oplus)$ is a field where

$$a \times b := (a \cdot b) \bmod p \qquad \textit{and} \qquad a \oplus b := (a + b) \bmod p.$$

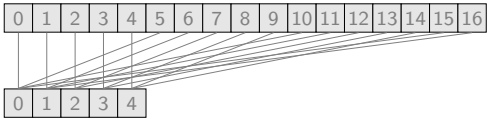
In particular $(\mathbb{F}_p^* := \mathbb{F}_p \setminus \{0\}, \times)$ is a group.

The class of Linear Hash Functions

Assume $D \subseteq \mathbb{F}_p$ for prime p. Then the following class is 1-universal:

$$\mathcal{H}_{p,m}^{\text{lin}} := \{x \mapsto ((a \times x) \oplus b) \bmod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}.$$

Can’t we use a simpler class?



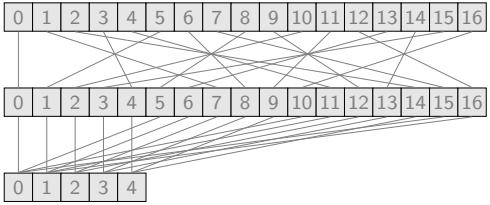
$\bmod m$

$p = 17$
$m = 5$

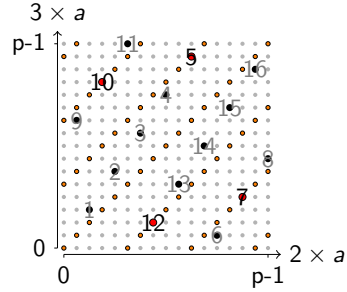
What about just $x \mapsto x \bmod m$?

Nothing is random. We have $h(0) = h(5)$ but this should hold only with probability $1/5$.

Can't we use a simpler class?



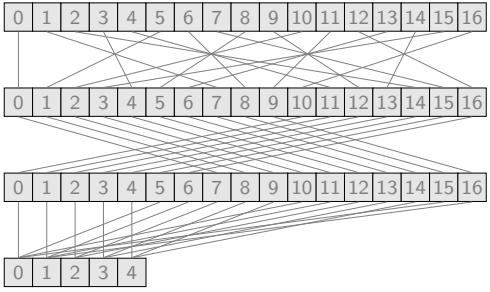
$p = 17$
$m = 5$
$a = 7$ // $\sim \mathcal{U}(\mathbb{F}_p^*)$



What about just $x \mapsto (a \times x) \bmod m$?

- Example: Do 2 and 3 collide? Picture $\{(a \times 2, a \times 3) \mid a \in \mathbb{F}_p^*\}$.
  $\Pr_{a \sim \mathcal{U}(\mathbb{F}_p^*)}[h(2) = h(3)] = \Pr[a \in \{5, 7, 10, 12\}] = \frac{4}{16} > \frac{3}{15} = \frac{1}{5}$.
  $\Rightarrow$ not 1-universal (but 2-universal)
- Also note: $h(0) = 0$ is not random.

Can’t we use a simpler class?



$p = 17$
$m = 5$
$a = 7$ // $\sim \mathcal{U}(\mathbb{F}_p^*)$
$b = 7$ // $\sim \mathcal{U}(\mathbb{F}_p)$

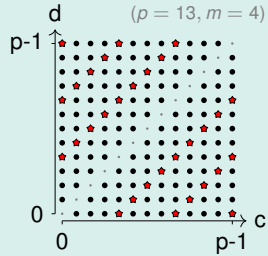
Back to $x \mapsto ((a \times x) \oplus b) \bmod m$

Mathematically “cleaner”. Proof of 1-universality on next slide.

Proof that $\mathcal{H}^{\text{lin}}_{p,m} := \{x \mapsto ((a \times x) \oplus b) \bmod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}$ is 1-universal.

Let $x \neq y \in \mathbb{F}_p$. (To show: $\Pr_{h \sim \mathcal{H}^{\text{lin}}_{p,m}}[h(x) = h(y)] \leq 1/m$.)

- Define $\begin{matrix} c = (a \times x) \oplus b \\ d = (a \times y) \oplus b \end{matrix} \Leftrightarrow \begin{pmatrix} c \\ d \end{pmatrix} = \underbrace{\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix}}_{\text{regular!}} \begin{pmatrix} a \\ b \end{pmatrix}$.
- The mapping $(a, b) \mapsto (c, d)$ is a bijection (for every $x \neq y$) from $\mathbb{F}_p^* \times \mathbb{F}_p \to P$. // $(a, b) \sim \mathcal{U}(\mathbb{F}_p^* \times \mathbb{F}_p) \Rightarrow (c, d) \sim \mathcal{U}(P)$
- Define *bad set* $B := \{(c, d) \in P \mid c \bmod m = d \bmod m\}$.
  ↪ from picture: $\frac{|B|}{|P|} \leq \frac{1}{m}$.



$P := \mathbb{F}_p \times \mathbb{F}_p \setminus \{(b, b) \mid b \in \mathbb{F}_p\}$

$$\Pr_{a,b \sim \mathcal{U}(\mathbb{F}_p^* \times \mathbb{F}_p)}[h(x) = h(y)] = \Pr_{a,b}[((a \times x) \oplus b) \bmod m = ((a \times y) \oplus b) \bmod m]$$

$$= \Pr_{a,b}[c \bmod m = d \bmod m] = \Pr_{a,b}[(c, d) \in B] = \Pr_{c,d \sim \mathcal{U}(P)}[(c, d) \in B] = \frac{|B|}{|P|} \leq \frac{1}{m}. \quad \square$$

Analysis of Hash Table with Chaining

. . . using a Universal Hash Family

Theorem

If $\mathcal{H} \subseteq [m]^D$ is a c-universal hash family then $T_{\text{chaining}}(n, m, \mathcal{H}) \leq 2 + c\alpha = \mathcal{O}(1)$ if $\alpha \in \mathcal{O}(1)$ and $c \in \mathcal{O}(1)$.

Proof: Mostly the same.

$$\forall S \subseteq [D], \forall x \in D: \quad \mathbb{E}_{h \sim \mathcal{U}(\mathcal{H})}\left[1 + |\{y \in S \mid h(y) = h(x)\}|\right]$$
$$\leq \ldots \leq 2 + \sum_{y \in S \setminus \{x\}} \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(y) = h(x)]$$
$$= 2 + \sum_{y \in S \setminus \{x\}} \frac{c}{m} \leq 2 + \frac{cn}{m} = 2 + c\alpha. \quad \square$$

More Universal Families

Examples for Universal Hash Families

- "$((ax + b) \bmod p) \bmod m$" is 1-universal

  as discussed: $D = \mathbb{F}_p, \qquad R = [m],$

  $\mathcal{H}^{\text{lin}}_{p,m} := \{x \mapsto ((a \times b) \oplus b) \bmod m \mid a \in \mathbb{F}_p^*, b \in \mathbb{F}_p\}$

- "$(ax \bmod p) \bmod m$" is only 2-universal:

  $$D = \mathbb{F}_p, \qquad R = [m],$$
  $$\mathcal{H} = \{x \mapsto (a \times b) \bmod m \mid a \in \mathbb{F}_p^*\}$$

- **Multiply-Shift** is 1-universal:

  $D = \{0, \ldots, 2^w - 1\}, \qquad R = \{0, \ldots, 2^\ell - 1\}$

  $\mathcal{H} = \{x \mapsto \lfloor((a \cdot x + b) \bmod 2^{2w})/2^{2w-\ell}\rfloor \mid$
  $a, b \in \{0, \ldots, 2^{2w} - 1\}\}.$

Selling point of multiply shift:

- "$x \bmod 2^{2w}$" drops some higher order bits
- "$\lfloor x/2^{2w-\ell}\rfloor$ drops some lower order bits
- No division or modulo operation needed!

For $w = 32$ (taken from Thorup 2015):

```
uint32_t hash(uint32_t x, uint32_t l,
              uint64_t a, uint64_t b) {
        return (a * x + b) >> (64-l);
}
```

Content

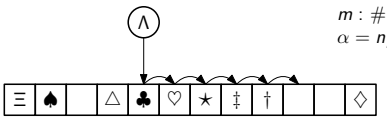
1. Conceptions: What is a Hash Function?
   - Hashing in the Wild
   - What should a Theorist do?

2. Use Case 1: Hash Table with Chaining
   - Using SUHA
   - Using Universal Hashing

3. Use Case 2: Linear Probing
   - Using SUHA
   - Using Universal Hashing

4. Conclusion

Hash Table with Linear Probing



Operations

For key x *probe* buckets $h(x), h(x)+1, h(x)+2, \ldots \pmod m$.

- **Insert.** Put x into first empty bucket.
- **Lookup.** Look for x, abort when encountering empty bucket.
- **Delete.** Lookup and remove x and ⚠ check if a key to the right wants to move into the hole.[a]

↪ For details see `https://en.wikipedia.org/wiki/Linear_probing`

[a] Alternative implementations leaves special *tombstones* markers.

Running Times

- Lookup($x \in S$): At most x's insertion time.
- Lookup($x \notin S$): At most the time it *would take* to insert x now.
- Delete($x \in S$): At most the time it *would take* to insert $y \notin S$ with $h(y) = h(x)$.

↪ It suffices to understand insertion times!

Theorem: Linear Probing under SUHA

Let $T_{n,m}$ be the random insertion time into a linear probing hash table. If $\frac{1}{2} \leq \alpha < 1$ then under SUHA we have

$$\mathbb{E}[T_{n,m}] = \mathcal{O}\left(\tfrac{1}{(1-\alpha)^2}\right) = \mathcal{O}(1). \quad \text{(not here)}$$

Content

1. Conceptions: What is a Hash Function?
   - Hashing in the Wild
   - What should a Theorist do?

2. Use Case 1: Hash Table with Chaining
   - Using SUHA
   - Using Universal Hashing

3. Use Case 2: Linear Probing
   - Using SUHA
   - Using Universal Hashing

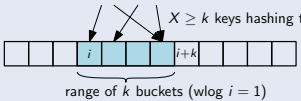
4. Conclusion

Preparation: A concentration bound

Chernoff

For $X \sim Bin(n,p)$ and $\delta \in [0,1]$ we have $\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq \exp(-\delta^2\mathbb{E}[X]/3)$.

Lemma: $\Pr[\geq k$ hits in segment of length $k]$



Let $k \in \mathbb{N}$ and $X = |\{y \in S \mid h(y) \in \{1,\dots,k\}\}|$.

Then $\Pr_{h \sim \mathcal{U}(R^D)}[X \geq k] \leq \exp(-(1-\alpha)^2 k/3)$.

Proof

Let $S = \{x_1,\dots,x_n\}$ and $X_i = [h(x_i) \in \{1,\dots,k\}] \sim Ber(\frac{k}{m})$.
Then $X = \sum_{i \in [n]} X_i \sim Bin(n, \frac{k}{m})$ with $\mathbb{E}[X] = \frac{kn}{m} = \alpha k$.

$$\begin{aligned}\Pr[X \geq k] &= \Pr[X \geq \tfrac{1}{\alpha}\mathbb{E}[X]] \\ &= \Pr[X \geq (1+\tfrac{1-\alpha}{\alpha})\mathbb{E}[X]] \\ &\leq \exp(-(\tfrac{1-\alpha}{\alpha})^2 \alpha k/3) \\ &\leq \exp(-(1-\alpha)^2 k/3).\end{aligned}$$

Proof: Expected LP-Insertion Time under SUHA is $\mathcal{O}(1)$

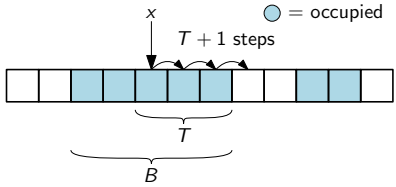
$$\mathbb{E}[T] \leq \mathbb{E}[B] = \sum_{k\geq 1} k \cdot \Pr[B = k] = \sum_{k\geq 1} k \cdot \Pr\left[\bigcup_{d=0}^{k-1} A_{h(x)-d,h(x)-d+k-1}\right]$$

$$\overset{(1)}{\leq} \sum_{k\geq 1} k \cdot \sum_{d=0}^{k-1} \Pr\left[A_{h(x)-d,h(x)-d+k-1}\right] \overset{(2)}{=} \sum_{k\geq 1} k \cdot k \cdot \Pr[A_{1,k}]$$

$$\overset{(3)}{\leq} \sum_{k\geq 1} k^2 \cdot \Pr[|\{y \in S \mid h(y) \in \{1,\ldots,k\}| \geq k]$$

$$\overset{(4)}{\leq} \sum_{k\geq 1} k^2 \cdot \exp(-(1-\alpha)^2 k/3) = \mathcal{O}(1).$$

Wolfram Alpha gives: $\int_0^\infty k^2 \exp(-(1-\alpha)^2 k/3) = \frac{54}{(1-\alpha)^6}$.



$A_{u,v}$: $\{u, v\}$ is maximal occupied block:

Reasoning:

(1) Union Bound.
(2) $h(x)$ is independent of keys in the table and hash distribution is invariant under cyclic shifts.
(3) Note: Keys stored in block cannot come in from the left.
(4) Chernoff argument from previous slide.

Content

1. Conceptions: What is a Hash Function?
   - Hashing in the Wild
   - What should a Theorist do?

2. Use Case 1: Hash Table with Chaining
   - Using SUHA
   - Using Universal Hashing

3. Use Case 2: Linear Probing
   - Using SUHA
   - Using Universal Hashing

4. Conclusion

Degrees of Independence

(Mutual / Collective) Independence

A family $\mathcal{E}$ of **events** is **independent** if $\forall k \in \mathbb{N}$ and distinct $E_1, \ldots, E_k \in \mathcal{E}$ we have

$$\Pr\left[\bigcap_{i=1}^{k} E_i\right] = \prod_{i=1}^{k} \Pr[E_i].$$

A family $\mathcal{X}$ of discrete **random variables** is **independent** if $\forall k \in \mathbb{N}$, distinct $X_1, \ldots, X_k \in \mathcal{X}$ and all $x_1, \ldots, x_k$ we have

$$\Pr\left[\bigwedge_{i=1}^{k} X_i = x_i\right] = \prod_{i=1}^{k} \Pr[X_i = x_i].$$

Pairwise Independence

A family of **events** is **pairwise independent** if any subfamily of size 2 is independent.

A family of **random variables** is **pairwise independent** if any subfamily of size 2 is independent.

d-wise Independence

A family of **events** is **d-wise independent** if any subfamily of size at most d is independent.

A family of **random variables** is **d-wise independent** if any subfamily of size at most d is independent.

d-Independent Hash Family

Definition: d-Independent Hash Family

A family $\mathcal{H} \subseteq R^D$ of hash functions is *d-independent* if for distinct $x_1, \ldots, x_d \in D$ and any $i_1, \ldots, i_d \in R$: (grey is implied by black)

$$\Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x_1) = i_1 \wedge \ldots \wedge h(x_d) = i_d] = \prod_{j=1}^{d} \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x_j) = i_j] = |R|^{-d}.$$

Alternative Definition

$\mathcal{H}$ is d-independent if for $h \sim \mathcal{U}(\mathcal{H})$

- the family $(h(x))_{x \in D}$ of random variables is d-independent *and*
- $h(x) \sim \mathcal{U}(R)$ for each $x \in D$.

Theorem

Let $D = R = \mathbb{F}$ be a finite field. Then

$$\mathcal{H} := \{x \mapsto \sum_{i=0}^{d-1} a_i x^i \mid a_0, \ldots, a_{d-1} \in \mathbb{F}\}$$

is a d-independent family.

Note: $\mathcal{H} \subseteq \mathbb{F}^{\mathbb{F}} \rightsquigarrow$ not yet useful.

Corollary: Smaller Ranges (proof omitted)

- If m divides $|\mathbb{F}|$, then adding "$\bmod m$" gives a d-independent family $\mathcal{H}' \subseteq [m]^{\mathbb{F}}$.
- If m does not divide $|\mathbb{F}|$, then adding "$\bmod m$" gives a family $\mathcal{H}' \subseteq [m]^{\mathbb{F}}$ such that for $h \sim \mathcal{U}(\mathcal{H}')$ the family $(h(x))_{x \in \mathbb{F}}$ is d-independent but only *approximately* uniformly distributed in $[m]$.

Proof: $\mathcal{H} := \{x \mapsto \sum_{i=0}^{d-1} a_i x^i \mid a_0, \ldots, a_{d-1} \in \mathbb{F}\}$ is d-independent

Let $x_1, \ldots, x_d \in \mathbb{F}$ be distinct keys and $i_1, \ldots i_d \in \mathbb{F}$ arbitrary.

$\hookrightarrow$ to show : $\Pr_{h \sim \mathcal{U}(\mathcal{H})}[\forall j \in [d] : h(x_j) = i_j] = |\mathbb{F}|^{-d}$.

For $h \in \mathcal{H}$ (given via $a_0, \ldots, a_{d-1}$) the following is equivalent:

$$\begin{array}{l} h(x_1) = i_1 \\ h(x_2) = i_2 \\ \vdots \\ h(x_d) = i_d \end{array} \iff \begin{array}{l} a_0 + a_1 x_1 + \cdots + a_{d-1} x_1^{d-1} = i_1 \\ a_0 + a_1 x_2 + \cdots + a_{d-1} x_2^{d-1} = i_2 \\ \vdots \\ a_0 + a_1 x_d + \cdots + a_{d-1} x_d^{d-1} = i_d \end{array} \iff \underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{d-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{d-1} \\ \vdots & & & \ddots & \vdots \\ 1 & x_d & x_d^2 & \ldots & x_d^{d-1} \end{pmatrix}}_{\text{Vandermonde matrix } M \Rightarrow \text{ regular}} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{pmatrix} = \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_d \end{pmatrix}$$

Exactly one vector $\vec{a} = M^{-1} \cdot \vec{i}$ solves the equation.

$$\Rightarrow \Pr_{h \sim \mathcal{U}(\mathcal{H})}[\forall j : h(x_j) = i_j] = \Pr_{a_0, \ldots, a_{d-1} \sim \mathcal{U}(\mathbb{F})}[\vec{a} = M^{-1} \cdot \vec{i}] = |\mathbb{F}|^{-d}. \quad \square$$

Concentration Bound for d-Independent Variables

(Tricky) Exercise

Let d be even and $X_1, \ldots, X_n \sim Ber(p)$ a d-independent family of random variables with $p = \Omega(1/n)$.
Let $X = \sum_{i=1}^{n} X_i$. Then for any $\delta > 0$ we have

$$\Pr[X - \mathbb{E}[X] \geq \delta \mathbb{E}[X]] = \mathcal{O}(\delta^{-d} \mathbb{E}[X]^{-d/2}).$$

Remark: Weaker than Chernoff, stronger than Chebyshev

- Chebycheff gives $\Pr[X - \mathbb{E}[X] \geq \delta \mathbb{E}[X]] \leq \frac{1-p}{\delta^2 \mathbb{E}[X]}$. (requires $d = 2$)
  $\hookrightarrow$ uses that $Var(X_1 + \cdots + X_n) = Var(X_1) + \cdots + Var(X_n)$ for pairwise independent $X_1, \ldots, X_n$.
- Chernoff gave $\Pr[X - \mathbb{E}[X] \geq \delta \mathbb{E}[X]] \leq \exp(-\delta^2 \mathbb{E}[X]/3)$. (requires $d = n$).

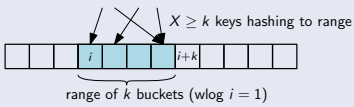
Preparation: A Concentration Bound

again for d-independence

Lemma (last slide)

For d-independent $X_1, \dots, X_n \sim Ber(p)$ and $X = \sum_{i \in [n]} X_i$ we have $\Pr[X \geq (1+\delta)\mathbb{E}[X]] = \mathcal{O}(\delta^{-d}\mathbb{E}[X]^{-d/2})$.

Lemma: $\geq k$ hits in segment of length k



Let $\mathcal{H}$ be a d-independent hash family and $h \sim \mathcal{U}(\mathcal{H})$.
Let $k \in \mathbb{N}$ and $X = |\{y \in S \mid h(y) \in \{1, \dots, k\}\}|$.

Then $\Pr[X \geq k] \leq \mathcal{O}((1-\alpha)^{-d}k^{-d/2})$.

Proof

Let $S = \{x_1, \dots, x_n\}$ and
$X_i = [h(x_i) \in \{1, \dots, k\}] \sim Ber(\frac{k}{m})$.
Then $X = \sum_{i \in [n]} X_i$ fits the Lemma with $\mathbb{E}[X] = \frac{kn}{m} = \alpha k$.

$$\begin{aligned}
\Pr[X \geq k] &= \Pr[X \geq \tfrac{1}{\alpha}\mathbb{E}[X]] \\
&= \Pr[X \geq (1 + \tfrac{1-\alpha}{\alpha})\mathbb{E}[X]] \\
&= \mathcal{O}\Big(\Big(\tfrac{1-\alpha}{\alpha}\Big)^{-d}(\alpha k)^{-d/2}\Big) \\
&\leq \mathcal{O}((1-\alpha)^{-d}k^{-d/2}). \quad \text{(using } \alpha \leq 1\text{)}
\end{aligned}$$

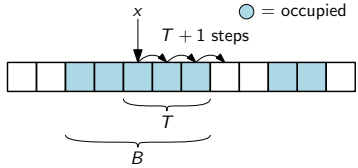
Theorem: Linear Probing with d-independence

Under the same conditions as before, except with 9-independent hash functions, the insertion time $T_{n,m}$ for linear probing satisfies:

$$\mathbb{E}[T_{n,m}] = \mathcal{O}(1)$$

Proof Sketch

$$\begin{aligned}
\mathbb{E}[T] &\leq \mathbb{E}[B] \leq \dots \\
&\overset{(1)}{\leq} \sum_{k \geq 1} k^2 \cdot \Pr[|\{y \in S \mid h(y) \in \{1, \dots, k\}\}| \geq k] \\
&\overset{(2)}{\leq} \sum_{k \geq 1} k^2 \cdot \mathcal{O}((1-\alpha)^{-8} k^{-8/2}) \\
&\leq \sum_{k \geq 1} k^{-2} \cdot \mathcal{O}((1-\alpha)^{-8}) \\
&\overset{(3)}{=} \tfrac{\pi^2}{6} \mathcal{O}((1-\alpha)^{-8}) = \mathcal{O}(1). \quad \square
\end{aligned}$$



Reasoning:

(1) Same as before, except we have to condition on $h(x)$ and may only use 8-independence in the following. (this is the hand wavy part!)

(2) Concentration bound from previous slide for $d = 8$.

(3) If interested, see 3Blue1Brown video:
`https://www.youtube.com/watch?v=d-o3eB9sfls`

Final Remarks on Linear Probing + Universal Hashing

Much more is known about insertion times of linear probing:

- Any 5-independent family gives $\mathcal{O}(\frac{1}{(1-\alpha)^2})$.
  $\hookrightarrow$ A. Pagh, R. Pagh, and Ruzic 2011
- An (artificially bad) 4-independent family gives $\Omega(\log n)$.
  $\hookrightarrow$ Pătraşcu and Thorup 2016
- A (well-designed) 4-independent family gives $\mathcal{O}(\frac{1}{(1-\alpha)^2})$.
  $\hookrightarrow$ Puatracscu and Thorup 2013

We Glossed over many Modern Insights

Storing Elements Naively is Inefficient (Cleary 1984)

Example: If $D = [2n]$ and $|S| = n$, then bitvector of $2n$ bits suffices.

$\hookrightarrow$ Much smaller than Array with $\mathcal{O}(n)$ entries of $\log_2 |D|$ bits!

In General: Should aim for $\log_2 \binom{|D|}{n}$ bits rather than $n \cdot \log_2 |D|$.

Tabulation Hashing offers Chernoff-type Concentration Bounds (Pătraşcu and Thorup 2012)

Pairs good practical performance with rigorous mathematical guarantees.

In Practice: Linear Probing can be Supercharged

E.g. using SIMD instructions. *Don't implement high-performance hash tables yourself.*

In Theory: Go Beyond Linear Probing (Bender, Farach-Colton, et al. 2022; Bender, Kuszmaul, and Zhou 2024)

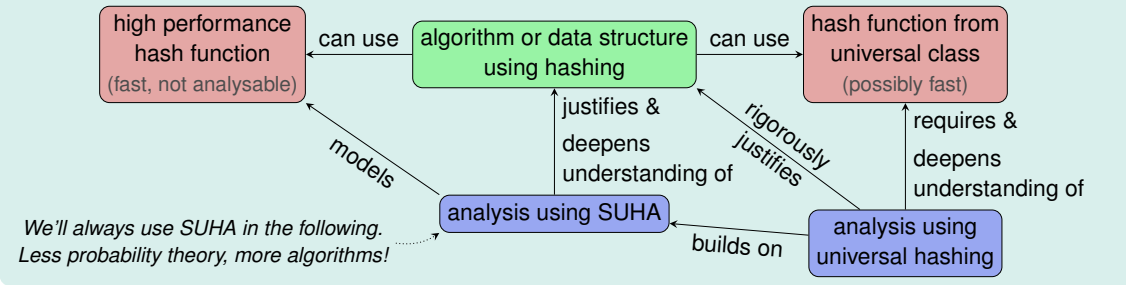
Goals: Improve worst-case access times, improve running time for $\alpha \to 1$.

Conclusion

Technical Takeaway: Performance of Hash Tables

For both an **ideal hash function** (SUHA) and a random hash function from a suitable **universal class**, a hash table using **linear probing** or **chaining** provably has an expected running time of $\mathcal{O}(\mathbf{1})$ per operation.

Non-Technical Takeaway: Approaches to analyse hashing based algorithms



Anhang: Mögliche Prüfungsfragen I

- Was könnte eine Idealvorstellung einer Hashfunktion sein? Inwiefern wäre eine ideale Hashfunktion nützlich? Was ist das Problem an dieser Vorstellung?
- Was ist die Simple Uniform Hashing Assumption (SUHA)? Was spricht dafür diese Annahme zu treffen? Welche Alternativen gibt es?
- Inwiefern ist eine pseudozufällige Funktion mit kryptographischen Ununterscheidbarkeitsgarantien nützlich für uns? Wie ist der Zusammenhang zur SUHA?*
- Universelles Hashing:
  - Wie ist c-Universalität definiert?
  - Welche c-universellen Hashklasse haben wir kennengelernt? Wie haben wir die c-Universalität bewiesen?
  - Wie ist d-Unabhängigkeit für eine Hashklasse definiert?
  - Welche d-universelle Hashklasse haben wir kennengelernt?
  - Welcher Zusammenhang besteht zwischen d-Unabhängigkeit und c-Universalität? (Übungsaufgabe)
  - Chernoff Schranken sind für Summen unabhängiger Zufallsvariablen gedacht. Was kann man machen, wenn die Zufallsvariablen nur d-unabhängig sind?*

Anhang: Mögliche Prüfungsfragen II

- Betrachten wir Hashing mit verketteten Listen:
  - Welche Schranke an die erwartete Einfügezeit haben wir bewiesen? Wie?
  - An welcher Stelle spielt die Verteilung der Hashfunktion eine Rolle?
  - Nenne eine hinreichende Eigenschaft, die eine universelle Hashklasse haben sollte, damit der Beweis funktioniert.
- Betrachten wir Hashing mit linearem Sondieren:
  - Welche Schranke an die erwartete Laufzeit haben wir bewiesen? Wie?
  - An welcher Stelle spielt die Verteilung der Hashfunktion eine Rolle?
  - Nenne eine hinreichende Eigenschaft, die eine universelle Hashklasse haben sollte, damit der Beweis funktioniert.
  - Wie wir diese Eigenschaft ausgenutzt?*

References I

[1] Michael A. Bender, Martin Farach-Colton, et al. “On the optimal time/space tradeoff for hash tables”. In: *54th STOC*. 2022, pp. 1284–1297. DOI: 10.1145/3519935.3519969.

[2] Michael A. Bender, William Kuszmaul, and Renfei Zhou. “Tight Bounds for Classical Open Addressing”. In: *CoRR* abs/2409.11280 (2024). DOI: 10.48550/ARXIV.2409.11280.

[3] John G. Cleary. “Compact Hash Tables Using Bidirectional Linear Probing”. In: *IEEE Trans. Computers* 33.9 (1984), pp. 828–834. DOI: 10.1109/TC.1984.1676499.

[4] Anna Pagh, Rasmus Pagh, and Milan Ruzic. “Linear Probing with 5-wise Independence”. In: *SIAM Rev.* 53.3 (2011), pp. 547–558. DOI: 10.1137/110827831. URL: https://doi.org/10.1137/110827831.

[5] Mihai Pătraşcu and Mikkel Thorup. “On the k-Independence Required by Linear Probing and Minwise Independence”. In: *ACM Trans. Algorithms* 12.1 (2016), 8:1–8:27. DOI: 10.1145/2716317. URL: https://doi.org/10.1145/2716317.

[6] Mihai Pătraşcu and Mikkel Thorup. “The Power of Simple Tabulation Hashing”. In: *J. ACM* (2012). DOI: 10.1145/2220357.2220361.

References II

[7] Mihai Puatracscu and Mikkel Thorup. “Twisted Tabulation Hashing”. In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*. Ed. by Sanjeev Khanna. SIAM, 2013, pp. 209–228. DOI: 10.1137/1.9781611973105.16. URL: https://doi.org/10.1137/1.9781611973105.16.

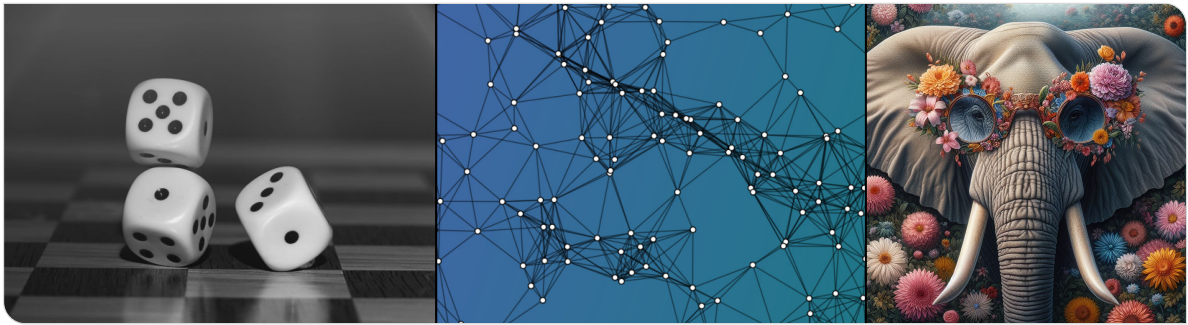
[8] Mikkel Thorup. “High Speed Hashing for Integers and Strings”. In: *CoRR* abs/1504.06804 (2015). arXiv: 1504.06804. URL: http://arxiv.org/abs/1504.06804.

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Bounded Differences and Bloom Filters

Stefan Walzer | WS 2024/2025



Content

1. Method of Bounded Differences

2. What is a Filter or AMQ?
   - Applications of Filters

3. The Bloom Filter Data Structure

4. Analysis of Bloom Filters
   - Expected fraction of zeroes in Bloom filters
   - Optimal tuning for Bloom filters
   - Main Theorem on Bloom filters

5. Conclusion

More Concentration Bounds

Hoeffding: Let $X = X_1 + \cdots + X_n$ where $a_i \leq X_i \leq b_i$, and $X_1, \ldots, X_n$ independent

$\Pr[X - \mathbb{E}[X] \geq t] \leq \exp(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2})$. // proof combines Chernoff with a different lemma

Special Case: $X_1, \ldots, X_n$ are Bernoulli random variables

$\Pr[X - \mathbb{E}[X] \geq t] \leq \exp(-2t^2/n)$. // incomparable to the Chernoff bound we saw: $\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$

Generalisation: McDiarmid / Method of Bounded Differences

- Assume $X_1, \ldots, X_n$ are independent and $X = f(X_1, \ldots, X_n)$.
- Assume for each i, a change in X_i changes $f(X_1, \ldots, X_n)$ by at most c_i.

$\Pr[X - \mathbb{E}[X] \geq t] \leq \exp(-\frac{2t^2}{\sum_{i=1}^{n} c_i^2})$ // same bound holds for $\Pr[\mathbb{E}[X] - X \geq t]$.

Proof uses Martingales... not here.

A Simple Use Case for the Method of Bounded Differences

Setting

- fixed graph $G = (V, E)$ and $s, t \in V$
- independent edge lengths $X_1, \ldots, X_m \sim \mathcal{U}([0, 1])$
- let P be the shortest $s - t$ path. // unique with probability 1

McDiarmid / Bounded Differences

- $X_1, \ldots, X_n$ independent and $X = f(X_1, \ldots, X_n)$.
- changing X_i changes $f(X_1, \ldots, X_n)$ by at most c_i.

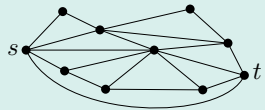
$\Rightarrow \Pr[X - \mathbb{E}[X] \geq t] \leq \exp(-\frac{2t^2}{\sum_{i=1}^{n} c_i^2})$.

Example where McDiarmid might be useful

Let $L :=$ length of P.

- $L = f(X_1, \ldots, X_m)$ for independent $X_1, \ldots, X_m$. ✓
- Changing X_i changes L by at most 1. $\rightsquigarrow c_i = 1$. ✓

$\Pr[L - \mathbb{E}[L] \geq t] \leq \exp(-\frac{2t^2}{m})$ // might be useful



A Simple Use Case for the Method of Bounded Differences

Setting

- fixed graph $G = (V, E)$ and $s, t \in V$
- independent edge lengths $X_1, \ldots, X_m \sim \mathcal{U}([0, 1])$
- let P be the shortest $s - t$ path. // unique with probability 1

McDiarmid / Bounded Differences

- $X_1, \ldots, X_n$ independent and $X = f(X_1, \ldots, X_n)$.
- changing X_i changes $f(X_1, \ldots, X_n)$ by at most c_i.

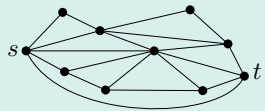
$\Rightarrow \Pr[X - \mathbb{E}[X] \geq t] \leq \exp(-\frac{2t^2}{\sum_{i=1}^{n} c_i^2})$.

Example where McDiarmid seems useless

Let $H :=$ number of edges in P. // "hops"

- $H = f(X_1, \ldots, X_m)$ for independent $X_1, \ldots, X_m$. ✓
- Changing X_i might change H by $n - 2$. $\rightsquigarrow c_i = O(n)$.

$\Pr[H - \mathbb{E}[H] \geq t] \leq \exp(-\frac{2t^2}{\mathcal{O}(mn^2)})$ // too weak



Content

1. Method of Bounded Differences

2. What is a Filter or AMQ?
   - Applications of Filters

3. The Bloom Filter Data Structure

4. Analysis of Bloom Filters
   - Expected fraction of zeroes in Bloom filters
   - Optimal tuning for Bloom filters
   - Main Theorem on Bloom filters

5. Conclusion

Filter = Approximate Membership Query Data Structure

Karlsruhe Institute of Technology

Setting

- universe D of possible keys
- a set $S \subseteq D$ of $n = |S|$
- a false positive probability ε

Want: Data structure representing S.

Space Requirement

- want $\mathcal{O}(n \log_2(1/\varepsilon))$ bits
- *much* smaller than $\mathcal{O}(n \log_2 |D|)$ bits needed for hash table

Operations

- **insert** elements to S and **delete** elements from S (optional)
- **query**: given $x \in D$ answer "is $x \in S$?" *approximately*:

$$\textbf{query}(x) = \textbf{YES} \text{ for } x \in S$$
$$\Pr[\textbf{query}(x) = \textbf{NO}] \geq 1 - \varepsilon \text{ for } x \notin S$$

a set S:

Anja Blancani
Peter Sanders
Florian Kurpicz
Hape Lehmann
Thomas Worsch

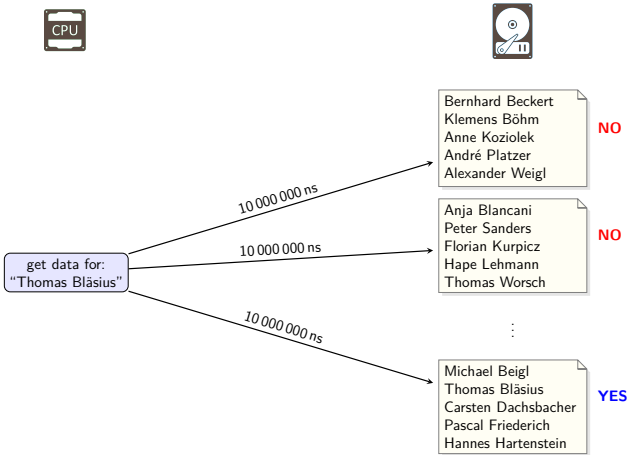
a filter for S:

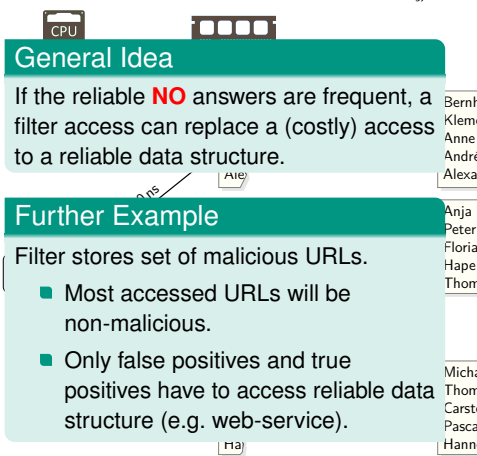
Anj
Pet
Flo
Hap
Tho

query(Peter Sanders) = **YES**
query(Petra Mutzel) = **YES**
query(Donald Knuth) = **NO**

The **YES** answers are **unreliable**.
The **NO** answers are **reliable**.

Simplified Use Case for Filters





Content

1. Method of Bounded Differences

2. What is a Filter or AMQ?
   - Applications of Filters

3. The Bloom Filter Data Structure

4. Analysis of Bloom Filters
   - Expected fraction of zeroes in Bloom filters
   - Optimal tuning for Bloom filters
   - Main Theorem on Bloom filters

5. Conclusion

Reminder: SUHA

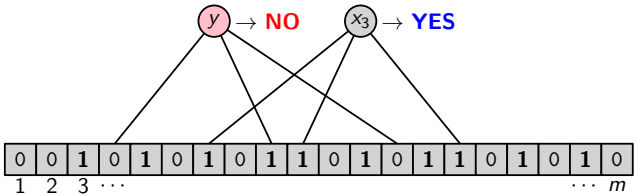
Simple Uniform Hashing Assumption (SUHA)

- We have access to $h \sim \mathcal{U}(R^D)$ for any R and D.
- h takes $\mathcal{O}(1)$ time to evaluate.
- h takes no space to store.

The Bloom Filter Data Structure

Parameters

m	length of a bit array $A[1..m]$ that we use
$k \in \mathcal{O}(1)$	number of hash functions $h_1, \ldots, h_k \sim \mathcal{U}([m]^D)$
n	number of keys in $S \subseteq D$ (dynamic)
$\alpha \in \mathcal{O}(1)$	load n/m (dynamic)



insert(x):

for $i \in [k]$ **do**
  $A[h_i(x)] = 1$

query(x):

for $i \in [k]$ **do**
  if $A[h_i(x)] = 0$ **then**
    return NO

return YES

delete(x):

⟨ not supported ⟩

Content

1. Method of Bounded Differences

2. What is a Filter or AMQ?
   - Applications of Filters

3. The Bloom Filter Data Structure

4. Analysis of Bloom Filters
   - Expected fraction of zeroes in Bloom filters
   - Optimal tuning for Bloom filters
   - Main Theorem on Bloom filters

5. Conclusion

Preparation

Exercise: Some approximations of e

$$\forall n \in \mathbb{N} : (1 + \tfrac{1}{n})^n \leq e \leq (1 + \tfrac{1}{n})^{n+1}$$
$$\text{and} \quad (1 - \tfrac{1}{n})^n \leq e^{-1} \leq (1 - \tfrac{1}{n})^{n-1}.$$

Corollaries

$$\forall n \in \mathbb{N} : (1 + \tfrac{1}{n})^n = e - \mathcal{O}(1/n)$$
$$\text{and} \quad (1 - \tfrac{1}{n})^n = e^{-1} - \mathcal{O}(1/n).$$

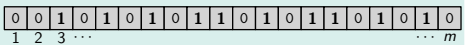
Bloom Filter Analysis (i)

Lemma

Assume $S = \{x_1, \ldots, x_n\}$ is inserted into the Bloom filter. Let $(A_1, \ldots, A_m) \in \{0,1\}^m$ be the random filter state and $Z := \sum_{i=1}^{m}(1 - A_i)$ the number of zeroes. Then

i $\mathbb{E}[\frac{Z}{m}] = (1 - \frac{1}{m})^{m\alpha k} = e^{-\alpha k} - o(1)$

ii For $y \notin S$: $\Pr[\textbf{query}(y) = \textbf{YES} \mid Z = z] = (1 - \frac{z}{m})^k$



Proof of (i).

$$\mathbb{E}[\tfrac{Z}{m}] = \tfrac{1}{m}\mathbb{E}[\sum_{i=1}^{m}(1 - A_i)] = \tfrac{1}{m}\sum_{i=1}^{m}\Pr[A_i = 0] = \tfrac{1}{m}\sum_{i=1}^{m}\Pr[A_1 = 0] = \Pr[A_1 = 0]$$

$$= \Pr[\forall x \in S : \forall i \in [k] : h_i(x) \neq 1] \overset{\text{SUHA}}{=} \prod_{x \in S}\prod_{i \in [k]}\Pr[h_i(x) \neq 1] \overset{\text{SUHA}}{=} \prod_{x \in S}\prod_{i \in [k]}(1 - \tfrac{1}{m})$$

$$= (1 - \tfrac{1}{m})^{nk} = (1 - \tfrac{1}{m})^{m\alpha k} = (e^{-1} - o(1))^{\alpha k} = e^{-\alpha k} - o(1).$$

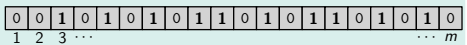
Bloom Filter Analysis (i)

Lemma

Assume $S = \{x_1, \ldots, x_n\}$ is inserted into the Bloom filter. Let $(A_1, \ldots, A_m) \in \{0,1\}^m$ be the random filter state and $Z := \sum_{i=1}^{m}(1 - A_i)$ the number of zeroes. Then

i $\mathbb{E}[\frac{Z}{m}] = (1 - \frac{1}{m})^{m\alpha k} = e^{-\alpha k} - o(1)$

ii For $y \notin S$: $\Pr[\textbf{query}(y) = \textbf{YES} \mid Z = z] = (1 - \frac{z}{m})^k$



Proof of (ii).

$$\Pr[\textbf{query}(y) = \textbf{YES} \mid Z = z] = \Pr[\forall i \in [k] : A_{h_i(y)} = 1 \mid Z = z] \overset{\text{SUHA}}{=} \prod_{i \in [k]} \left(\frac{m - z}{m}\right) = (1 - \tfrac{z}{m})^k.$$

How should a Bloom filter be configured?

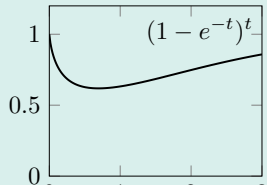
Approximate false positive rate

From the previous Lemma we get for $y \notin S$:

$$\varepsilon = \Pr[\textbf{query}(y) = \textbf{YES}] \approx \Pr[\textbf{query}(y) = \textbf{YES} \mid Z = \mathbb{E}[Z]]$$
$$\overset{\text{ii}}{=} \left(1 - \frac{\mathbb{E}[Z]}{m}\right)^k \overset{\text{i}}{=} (1 - e^{-\alpha k} + o(1))^k \approx (1 - e^{-\alpha k})^k.$$

Which k minimises ε? (when α is fixed)

$$\begin{aligned} & \arg\min_{k \in \mathbb{N}} (1 - e^{-\alpha k})^k \\ = \; & \arg\min_{k \in \mathbb{N}} (1 - e^{-\alpha k})^{\alpha k} \\ \approx \; & \tfrac{1}{\alpha} \arg\min_{t \in \mathbb{R}_+} (1 - e^{-t})^t \\ = \; & \tfrac{1}{\alpha} \arg\min_{t \in \mathbb{R}_+} t \ln(1 - e^{-t}) \end{aligned}$$



- plot $(1 - e^{-t})^t \rightsquigarrow$ one global minimum.
- deriving $t \ln(1 - e^{-t})$ gives $\ln(1 - e^{-t}) + \frac{te^{-t}}{1 - e^{-t}}$
- $t = \ln(2)$ is root of the derivative.

$\hookrightarrow k = \ln(2)/\alpha$ is optimal for fixed α.
$\hookrightarrow$ choose α and k such that $\alpha k = \ln(2)$

False Positive Probability and Space

Intuition for optimality of $\alpha k = \ln(2)$

- gives $\mathbb{E}[\frac{Z}{m}] \approx e^{-\alpha k} = \frac{1}{2}$
- maximises *entropy* of the filter bits

Theorem

A Bloom filter with $k \in \mathbb{N}$ hash functions and load factor $\alpha = \ln(2)/k$ has

i. a **space requirement** of $m = n/\alpha = \frac{kn}{\ln 2} \approx 1.44kn$ bits
ii. and a **false positive probability** of $\varepsilon = 2^{-k} + o(1)$.

(i) Space Requirement

Nothing to proof.

(ii) False Positive Probability

- **Know:** $\mathbb{E}[\frac{Z}{m}] \approx e^{-\alpha k} = \frac{1}{2}$ by choice of α
- **Know:** $\frac{Z}{m} = \frac{1}{2} \Rightarrow \varepsilon = 2^{-k}$
- **Need:** $\frac{Z}{m} \approx \mathbb{E}[\frac{Z}{m}]$, i.e. good concentration

Concentration bound for Z

Lemma

i. $\Pr[Z \leq \mathbb{E}[Z] - t \quad] \leq \exp(-\Theta(t^2/m))$ for any $t > 0$,

ii. $\Pr[Z \leq \mathbb{E}[Z] - m^{2/3}] \leq \exp(-\Theta(m^{1/3}))$ by setting $t = m^{2/3}$.

McDiarmid / Bounded Differences

- $X_1, \ldots, X_n$ independent and $X = f(X_1, \ldots, X_n)$.
- changing X_i changes $f(X_1, \ldots, X_n)$ by at most c_i.

$\Rightarrow \Pr[\mathbb{E}[X] - X \geq t] \leq \exp(-\frac{2t^2}{\sum_{i=1}^{n} c_i^2})$.

Difficulty: Filter bits $A_1, \ldots, A_m$ are not independent

- note e.g. that $A_1 = \ldots = A_{m-1} = 0 \Rightarrow A_m = 1$
- Generally: $A_1, \ldots, A_m$ are *negatively associated*:
  - Knowing that some bits are zero makes it more likely that others are 1 and vice versa.
- Standard Chernoff bounds don't apply.

Concentration bound for Z

Lemma

i. $\Pr[Z \leq \mathbb{E}[Z] - t \quad] \leq \exp(-\Theta(t^2/m))$ for any $t > 0$,

ii. $\Pr[Z \leq \mathbb{E}[Z] - m^{2/3}] \leq \exp(-\Theta(m^{1/3}))$ by setting $t = m^{2/3}$.

McDiarmid / Bounded Differences

- $X_1, \ldots, X_n$ independent and $X = f(X_1, \ldots, X_n)$.
- changing X_i changes $f(X_1, \ldots, X_n)$ by at most c_i.

$\Rightarrow \Pr[\mathbb{E}[X] - X \geq t] \leq \exp(-\frac{2t^2}{\sum_{i=1}^{n} c_i^2})$.

Proof of (i) using the method of bounded differences.

- Z is a function of kn independent hash values // SUHA
- each hash value can change Z by at most 1

$$\Rightarrow \Pr[Z \leq \mathbb{E}[Z] - t] \leq \Pr[\mathbb{E}[Z] - Z \geq t] = \exp\left(\frac{-2t^2}{nk}\right) = \exp\left(\frac{-2t^2}{m\alpha k}\right) = \exp\left(\frac{-2t^2}{m\ln(2)}\right). \quad \square$$

False Positive Probability of Bloom filters

Proof of Theorem, part (ii)

By choice of k and α we have $\mathbb{E}[\frac{Z}{m}] = e^{-\alpha k} - o(1) = \frac{1}{2} - o(1)$.
Let $y \notin S$ and $B = \lfloor \mathbb{E}[Z] - m^{2/3} \rfloor$.

$$\varepsilon = \Pr[\textbf{query}(y) = \textbf{YES}] \overset{\text{LTP}}{=} \sum_{z=1}^{m} \Pr[Z = z] \cdot \Pr[\textbf{query}(y) = \textbf{YES} \mid Z = z] = \sum_{z=1}^{m} \Pr[Z = z] \cdot (1 - \tfrac{z}{m})^k$$

$$\leq \sum_{z=1}^{B} \Pr[Z = z] + \sum_{z=B+1}^{m} \Pr[Z = z](1 - \tfrac{B+1}{m})^k \leq \Pr[Z \leq B] + (1 - \tfrac{B+1}{m})^k$$

$$\leq \Pr[Z \leq \mathbb{E}[Z] - m^{2/3}] + \big(1 - \tfrac{\mathbb{E}[Z] - m^{2/3}}{m}\big)^k \overset{\text{ii}}{\leq} \exp(-\Theta(m^{1/3})) + (1 - \tfrac{1}{2} + o(1))^k = 2^{-k} + o(1). \quad \square$$

How to Configure Your Bloom Filter

Theorem

A Bloom filter with $k \in \mathbb{N}$ hash functions and load factor $\alpha = \ln(2)/k$ has

i. a **space requirement** of $m = n/\alpha = \frac{kn}{\ln 2} \approx 1.44kn$ bits
ii. and a **false positive probability** of $\varepsilon = 2^{-k} + o(1)$.

How to determine m and k (the parameters you actually need)

1. n: determined by input
2. ε: choose a trade-off between space usage and false positive probability
   - If utility comes from negative answers "$x \notin S$, definitely" and running time is negligible, then:
     - want to maximise utility − disutility, where: ($\propto$ means "proportional to")
     - utility $\propto$ negative answers $=$ queries $\cdot \Pr_{x \sim \text{QueryDistribution}}[x \notin S] \cdot (1 - \varepsilon)$
     - disutility $\propto$ space consumption $= 1.44 \log_2(1/\varepsilon) n$ bits of RAM or cache
3. compute $k = \lceil \log_2(1/\varepsilon) \rceil$ // effectively restricts ε to powers of 2
4. compute $\alpha = \ln(2)/k$ and $m = \lceil n/\alpha \rceil$

Remarks

Much, much more is known

- more functionality
  - ↪ counting Bloom filters support deletions
- better query times
  - ↪ blocked Bloom filters improve cache efficiency
- better space efficiency
  - ↪ cuckoo filters use $n \log_2(1/\varepsilon) + \mathcal{O}(n)$ bits rather than $\approx 1.44 n \log_2(1/\varepsilon)$ bits
  - ↪ static filters (no insertions or deletions) use $n \log_2(1/\varepsilon) + o(n)$ bits.
- ...

Conclusion

Method of Bounded Differences

- requires $X = f(X_1, \ldots, X_n)$ for independent $X_1, \ldots, X_n$
- requires that changing X_i changes X only a little
- gives Chernoff-like concentration bounds

Approximate Membership Data Structure

- decides "is $x \in S$?" with *false positive probability* ε
- much smaller than exact membership data structure
  - $\approx 1.44 \log_2(1/\varepsilon)$ bits per element for Bloom filters
  - often fits into cache or RAM
- used to avoid costly access to exact data structure
  - reliable on **NO** answers
  - useful if **NO** answers are frequent

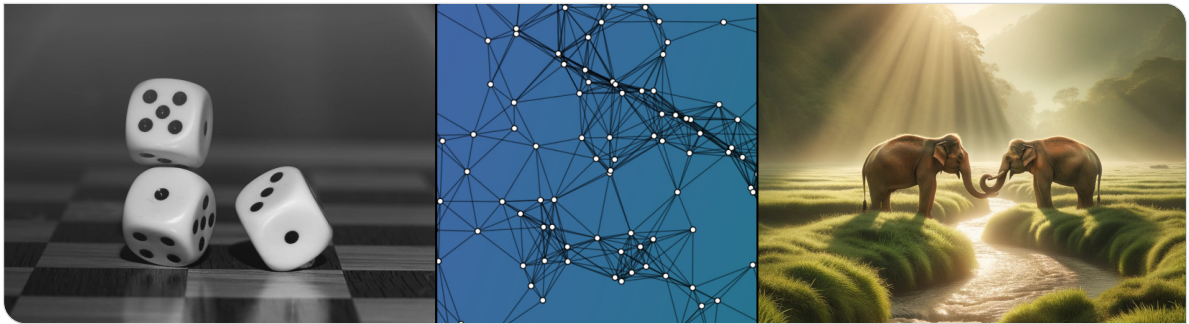
Anhang: Mögliche Prüfungsfragen I

- Methode der beschränkten Differenzen
  - Was sind die Voraussetzungen und Garantien von McDiarmids Ungleichung?
  - Was wäre ein Beispiel wo sie gute bzw. schlechte Garantien liefert?
- Approximate-Membership-Query Datenstrukturen im Allgemeinen
  - Welche Aufgabe hat eine AMQ Datenstruktur?
  - Was ist der Vorteil gegenüber einer exakten Datenstruktur?
  - Was wäre ein Anwendungsfall, in dem eine AMQ Datenstruktur nützlich ist?
- Bloomfilter
  - Wie ist ein Bloomfilter aufgebaut und welche Operationen unterstützt er?
  - Welche Parameter gibt es, und wie hängen diese zusammen?
  - Was hat unsere Analyse zur geschickten Wahl der Parameter zu sagen? Wie werden die übrigen Parameter gewählt? Welcher Speicherverbrauch ergibt sich?
  - Fragen zur Analyse
    - Welche Anzahl von Nullen bzw. Einsen erwarten wir?
    - Wie hängt die falsch-positiv Wahrscheinlichkeit mit der Anzahl Nullen bzw. Einsen zusammen?
    - Wir kann man argumentieren, dass die Anzahl Nullen bzw. Einsen im Bloomfilter nahe am Erwartungswert liegt?

Karlsruhe Institute of Technology

Probability and Computing
Coupling, Balls into Bins, Poissonisation and the Poisson Point Process

Stefan Walzer | WS 2024/2025



Content

1. Coupling
   - Motivating Examples
   - Definition
2. Balls into Bins
3. Poissonisation
4. Poisson Point Process

An easy choice?

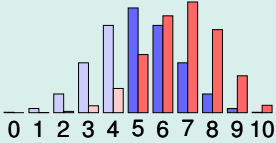
A Simple Game

You win if you get ≥ 5 heads in 10 coin tosses. Choose:

i. a fair coin with $\Pr[\text{“heads”}] = \frac{1}{2}$

ii. a biased coin with $\Pr[\text{“heads”}] = \frac{2}{3}$

How to prove that (ii) is the better choice?



fair coin

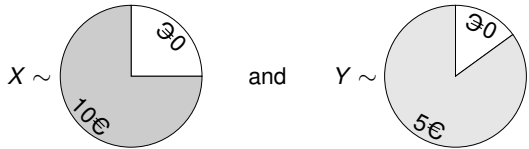
biased coin

$$\sum_{i=5}^{10} \binom{10}{i} \left(\frac{1}{2}\right)^i \left(\frac{1}{2}\right)^{10-i} \stackrel{?}{<} \sum_{i=5}^{10} \binom{10}{i} \left(\frac{2}{3}\right)^i \left(\frac{1}{3}\right)^{10-i}$$

Shouldn't there be an answer that needs no calculation?

Which Lottery do you prefer?

Consider two “wheels of fortune”:

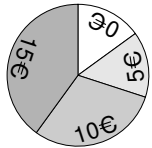


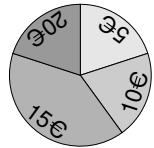
Both can be rationally preferred

- $\mathbb{E}[X] > \mathbb{E}[Y]$ // maximises expected reward
- $\Pr[Y \geq 5€] > \Pr[X \geq 5€]$ // maximises probability that you can afford ice cream

See `https://en.wikipedia.org/wiki/Von_Neumann%E2%80%93Morgenstern_utility_theorem` to get started on rational choice theory.

Which Lottery do you prefer?

$X \sim$ 

and $Y \sim$ 

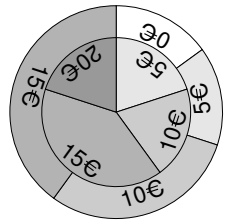
Formal Reason you should prefer Y

For every c we have:

$$\Pr[X \geq c] \leq \Pr[Y \geq c].$$

Intuitive Reason you should prefer Y

Glueing the wheels together guarantees $X < Y$.

$(X, Y) \sim$ 

Content

1. Coupling
   - Motivating Examples
   - Definition
2. Balls into Bins
3. Poissonisation
4. Poisson Point Process

Equality in Distribution

Notation

We write $X \overset{\mathrm{d}}{=} X'$ for two random variables if X and X' have the same distribution.

Equivalent Definitions

$$X \overset{\mathrm{d}}{=} X' \Leftrightarrow \forall x : \Pr[X = x] = \Pr[X' = x] \qquad \text{(for discrete R.V. } X \text{ and } X'\text{)}$$

$$\Leftrightarrow \forall x : \Pr[X \leq x] = \Pr[X' \leq x] \qquad \text{(for real-valued R.V. } X \text{ and } X'\text{)}$$

To Clarify:

If $X, Y \sim \mathcal{U}([0,1])$ are independent then

- $X \overset{\mathrm{d}}{=} Y$
- $\Pr[X = Y] = 0$

Definition: Coupling of X and Y

A random variable X | A random variable Y

↓ ↓

A Coupling of X and Y

A random variable (X', Y') with

- $X' \overset{\mathrm{d}}{=} X$
- $Y' \overset{\mathrm{d}}{=} Y$

$X \sim$  | $Y \sim$ 

↓ ↓

A Coupling of X and Y

$(X', Y') \sim$ 

- $X' \overset{\mathrm{d}}{=} X$ ✓
- $Y' \overset{\mathrm{d}}{=} Y$ ✓

Remarks

- No assumption on joint distribution of X and Y. Might be independent, correlated or undefined.
- X' and Y' should be correlated in an interesting/useful way.
- Example coupling shows:

$$\Pr[X \geq c] \overset{X \overset{\mathrm{d}}{=} X'}{=} \Pr[X' \geq c] \overset{X' \leq Y'}{\leq} \Pr[Y' \geq c] \overset{Y \overset{\mathrm{d}}{=} Y'}{=} \Pr[Y \geq c]$$

An easy choice!

Karlsruhe Institute of Technology

A Simple Game (Generalised)

You win if your random variable exceeds $c \in \mathbb{N}$. Choose:

i $X \sim \mathrm{Bin}(n, \frac{1}{2})$ // number of heads of fair coin

ii $Y \sim \mathrm{Bin}(n, \frac{2}{3})$ // number of heads of biased coin

Prove that Y is better than X using a Coupling

Let $R_1, \ldots, R_n \sim \mathcal{U}([6])$ be n fair dice rolls.

- $X' = |\{i \in [n] \mid R_i \in \{1,2,3\}\}|$
- $Y' = |\{i \in [n] \mid R_i \in \{1,2,3,4\}\}|$

Observe:

- $X' \overset{\mathrm{d}}{=} X$
- $Y' \overset{\mathrm{d}}{=} Y$
- $X' \leq Y'$ guaranteed

Hence: $\Pr[X \geq c] = \Pr[X' \geq c] \leq \Pr[Y' \geq c] = \Pr[Y \geq c]$.

Content

1. Coupling
   - Motivating Examples
   - Definition

2. Balls into Bins

3. Poissonisation

4. Poisson Point Process

Balls Into Bins

General Terminology

- m balls are randomly distributed among n bins
- the *load* of a bin is the number of balls in it

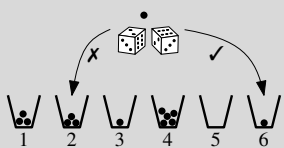
$m = 13, \quad n = 6$

Fully Random Allocation

- $X_1, \ldots, X_m \sim \mathcal{U}([n])$ independent
- $L_i := |\{j \in [m] \mid X_j = i\}|$ is the load of bin $i \in [m]$
- $(L_1, \ldots, L_n)$ follows a (specific) *multinomial distribution*

Example for Partially Random Allocation (not in this lecture)

- balls are placed sequentially
- each ball chooses the *least loaded* among two randomly chosen bins (ties broken randomly)



Balls into Bins: Many Interesting Questions

- What is the expected/distribution/concentration of
  - the load $L_{\max}$ of the most loaded bin
  - the load $L_{\min}$ of the least loaded bin
  - $L_{\max} - L_{\min}$
  - the number of empty bins
  - ...
- Can we make the allocation more balanced by intervening in some way?
  - e.g. with partially random allocation from last slide $\mathbb{E}[L_{\max} - L_{\min}]$ stays bounded when $m \to \infty$ while n is fixed.

Countless variants exist...

“Balls into Bins” is everywhere

Hashing with Chaining $\longleftrightarrow$ n Balls into m Bins

length of the list in bucket i $\longleftrightarrow$ number of balls in bin i

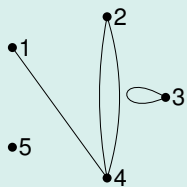
Bloom Filter with k Hash Functions $\longleftrightarrow$ kn Balls into m Bins

a filter bit is set to 1 $\longleftrightarrow$ ith bin is non-empty

Degree Sequence of Random (Multi-)Graph $\longleftrightarrow$ $2m$ Balls into n bins

Given independent $v_1, \ldots, v_{2m} \sim \mathcal{U}([n])$ let
$G = (V = [n], E = \{\{v_1, v_2\}, \ldots, \{v_{2m-1}, v_{2m}\}\})$
(we allow multiedges and loops in G)

degree of vertex i $\longleftrightarrow$ load of bin i



$n = 5, \quad m = 4$

$v_1 = 1, \quad v_2 = 4$
$v_3 = 4, \quad v_4 = 2$
$v_5 = 3, \quad v_6 = 3$
$v_7 = 2, \quad v_8 = 4$

“Balls into Bins” is the standard language for discussing underlying mathematical questions.

Content

1. Coupling
   - Motivating Examples
   - Definition

2. Balls into Bins

3. Poissonisation

4. Poisson Point Process

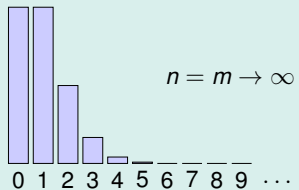
Load of a Single Bin

Setting: Expected Constant Load per Bin

- fully random allocation
- $m = \lambda n$ balls n bins for large n
- λ fixed constant

Load of the First Bin

Consider $L^{(n)} \sim \text{Bin}(\lambda n, \frac{1}{n})$. For $\lambda = 1$:



Poisson Distribution

For $\lambda \in \mathbb{R}_{\geq 0}$, $X \sim \text{Pois}(\lambda)$ is a random variable with

$$\Pr[X = i] = e^{-\lambda} \frac{\lambda^i}{i!}$$ // note: probabilities sum to 1

Theorem (proof on blackboard)

$\lim_{n \to \infty} \Pr[L^{(n)} = i] = \Pr[X = i]$.

Remarks

- we say "$L^{(n)}$ converges in distribution to X"
- we write $L^{(n)} \xrightarrow{d} X$
- this formally refers to convergence of CDFs

Properties of the Poisson Distribution

Exercise: $X \sim \text{Pois}(\lambda)$ has Nice Properties

i. $\mathbb{E}[X] = \lambda$.

ii. $\text{Var}(X) = \lambda$.

iii. Let $Y \sim \text{Pois}(\rho)$ be independent of X. Then $X + Y \sim \text{Pois}(\lambda + \rho)$.

iv. Let $X' \sim \text{Bin}(X, p)$. Then $X' \sim \text{Pois}(\lambda p)$.

Poissonised Balls into Bins

λn Balls into n Bins Model

- $X_1, \dots, X_{\lambda n} \sim \mathcal{U}([n])$
- $L_i := |\{j \in [m] \mid X_j = i\}| \sim \mathrm{Bin}(\lambda n, \frac{1}{n})$
- $(L_i)_{i \in [n]}$ *not independent*
  - e.g. large L_1 is (weak) evidence for small L_2
  - annoying in analysis
- number λn of balls *fixed*

"Poissonised" Model

- $L_1, \dots, L_n \sim \mathrm{Pois}(\lambda)$ independent
  - extremely convenient for analysis
- $\mathbb{E}[L_1 + \dots + L_n] = \lambda n$
- number of balls *random* $\sim \mathrm{Pois}(\lambda n)$
  - unusual setting in practice

Wouldn't it be nice...

... if we could switch between the models whenever convenient?

Connection: Poissonised and Regular Balls into Bins

Lemma 1

Let $n \in \mathbb{N}$ and $\lambda > 0$. Consider two variants of Poissonised balls into bins:

Regular Variant:

- sample $L_1, \ldots, L_n \sim \text{Pois}(\lambda)$

Sum-First-Variant:

- sample $M \sim \text{Pois}(\lambda n)$
- perform a regular M balls into n bins experiment
  - sample $X_1, \ldots, X_M \sim \mathcal{U}([n])$
  - let $L'_i := |\{j \in [M] \mid X_j = i\}|$

Both variants are equivalent, i.e. $(L_1, \ldots, L_n) \overset{\text{d}}{=} (L'_1, \ldots, L'_n)$.

What we need to show (calculation on blackboard):

For arbitrary $(\ell_1, \ldots, \ell_n) \in \mathbb{N}^n$: $\Pr[(L_1, \ldots, L_n) = (\ell_1, \ldots, \ell_n)] = \Pr[(L'_1, \ldots, L'_n) = (\ell_1, \ldots, \ell_n)]$.

Some Concentration Bounds

Lemma 2

i. Let $\Lambda > 0$ and $X \sim \text{Pois}(\Lambda)$. Then $\Pr[|X - \Lambda| > t] \leq \frac{\Lambda}{t^2}$ for any $t > 0$. // Chebyschev

ii. Let $\lambda = \Theta(1)$, and $X \sim \text{Pois}(\lambda n)$ then $\Pr[X = \lambda n \pm \mathcal{O}(n^{2/3})] = 1 - o(1)$.

iii. Let $\lambda = \Theta(1)$, $\lambda^+ := \lambda + n^{-1/3}$ and $X^+ \sim \text{Pois}(\lambda^+ n)$ then $\Pr[X^+ \geq \lambda n] = 1 - o(1)$.

iv. Let $\lambda = \Theta(1)$, $\lambda^- := \lambda - n^{-1/3}$ and $X^- \sim \text{Pois}(\lambda^- n)$ then $\Pr[X^- \leq \lambda n] = 1 - o(1)$.

v. In particular: $\Pr[X^- \leq \lambda n \leq X^+] = 1 - o(1)$.

Coupling of Poissonised and Regular Balls into Bins

Karlsruhe Institute of Technology

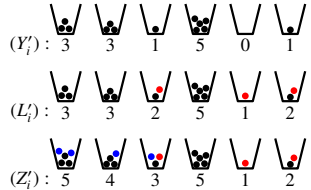
Theorem

Let $n, \lambda, \lambda^+, \lambda^-$ be as before. Consider three "balls into bins" models:

1. $Y_1, \dots, Y_n \sim \text{Pois}(\lambda^-)$ // poissonised with reduced λ
2. $L_1, \dots, L_n$ arising from regular $m = \lambda n$ balls into n bins
3. $Z_1, \dots, Z_n \sim \text{Pois}(\lambda^+)$ // poissonised with increased λ

There is a coupling $(Y_i', L_i', Z_i')_{i\in[n]}$ of $(Y_i)_{i\in[n]}$, $(L_i)_{i\in[n]}$, $(Z_i)_{i\in[n]}$ such that

with probability $1 - o(1)$: $Y_i' \le L_i' \le Z_i'$ for all $i \in [n]$.



Proof.

Let $X_1, X_2, \dots \sim \mathcal{U}([n])$, $M^- \sim \text{Pois}(\lambda^- n)$, $M^+ \sim \text{Pois}(\lambda^+ n)$.

- $Y_i' := |\{j \in [M^-] \mid X_j = i\}|$ for $i \in [n]$.
- $L_i' := |\{j \in [m\ \ ] \mid X_j = i\}|$ for $i \in [n]$.
- $Z_i' := |\{j \in [M^+] \mid X_j = i\}|$ for $i \in [n]$.

This is indeed a coupling as claimed:

- $(Y_i')_{i\in[n]} \overset{\text{d}}{=} (Y_i)_{i\in[n]}$ by Lemma 1.
- $(L_i')_{i\in[n]} \overset{\text{d}}{=} (L_i)_{i\in[n]}$ by construction.
- $(Z_i')_{i\in[n]} \overset{\text{d}}{=} (Z_i)_{i\in[n]}$ by Lemma 1.

By the Corollary we have $M^- \le m \le M^+$ with probability $1 - o(1)$. In that case clearly $Y_i' \le L_i' \le Z_i'$ for all $i \in [n]$. □

Coupling of Poissonised and Regular Balls into Bins

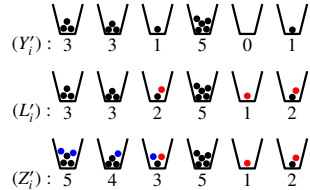
Theorem

Let $n, \lambda, \lambda^+, \lambda^-$ be as before. Consider three "balls into bins" models:

1. $Y_1, \dots, Y_n \sim \text{Pois}(\lambda^-)$ // poissonised with reduced λ
2. $L_1, \dots, L_n$ arising from regular $m = \lambda n$ balls into n bins
3. $Z_1, \dots, Z_n \sim \text{Pois}(\lambda^+)$ // poissonised with increased λ

There is a coupling $(Y_i', L_i', Z_i')_{i\in[n]}$ of $(Y_i)_{i\in[n]}$, $(L_i)_{i\in[n]}$, $(Z_i)_{i\in[n]}$ such that

with probability $1 - o(1)$: $Y_i' \le L_i' \le Z_i'$ for all $i \in [n]$.



Application involving Monotonous Functions

Let $f : \mathbb{N}_0^n \to \mathbb{R}$ be non-decreasing in each argument.
Examples:

- maximum load of a bin
- longest run of non-empty bins
- collision number // numbers of pairs of co-located balls

For some bound $B \in \mathbb{R}$ let

- $p^- := \Pr[f((Y_i)_{i\in[n]}) \ge B]$ // easier to compute
- $p \;\; := \Pr[f((L_i)_{i\in[n]}) \ge B]$ // what we want
- $p^+ := \Pr[f((Z_i)_{i\in[n]}) \ge B]$ // easier to compute

Then $p \in [p^- - o(1), p^+ + o(1)]$.

Back to Bloom Filters

Exercise:

Analyse Bloom filters in a “Poissonised” model and discuss how the results can be transferred to the exact model.

Content

1. Coupling
   - Motivating Examples
   - Definition

2. Balls into Bins

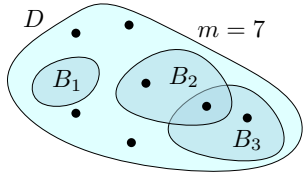
3. Poissonisation

4. Poisson Point Process

What about “Balls into Continuous Domain”?

Setting

- D is space of finite measure
- $m \in \mathbb{N}$ // number of balls
- $X_1, \ldots, X_m \sim \mathcal{U}(D)$ // randomly thrown into D



Note: If $D = \{1, \ldots, n\}$ we have discrete balls into bins.

Same annoying issue

If $B_1, B_2 \subseteq D$ with $B_1 \cap B_2 = \varnothing$ are two “bins” then the numbers L_1 and L_2 of “balls” in B_1 and B_2 are correlated.

Similar elegant solution

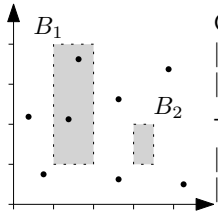
- We can “Poissonise” the setting.
- But we drop “balls into bins” terminology:
  - we allow infinite domains D
  - we allow infinite number of balls

Definitions of the Poisson Point Process

General Definition

Let D be a measurable space with measure μ. // e.g. $D = \mathbb{R}^2$ and $\mu =$"area"
The Poisson point process with parameter $\lambda \in \mathbb{R}_{\geq 0}$ is a random set $P \subseteq D$ such that

1. $|P \cap B| \sim \text{Pois}(\lambda\mu(B))$ for any $B \subseteq D$ with $\mu(B) < \infty$
2. $|P \cap B_1|$ and $|P \cap B_2|$ are independent whenever $B_1 \cap B_2 = \varnothing$



Generally:
$|B_1 \cap P| \sim \text{Po}(3\lambda)$
$|B_2 \cap P| \sim \text{Po}(\frac{1}{2}\lambda)$

This outcome:
$|B_1 \cap P| = 2$
$|B_2 \cap P| = 0$

Equivalent Definition if $\mu(D) < \infty$

- sample $M \sim \text{Pois}(\lambda\mu(D))$
- sample $X_1, \ldots, X_M \sim \mathcal{U}(D)$

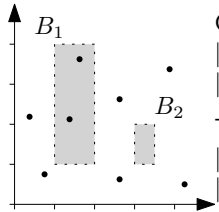
Then $P \overset{\text{d}}{=} \{X_1, X_2, \ldots, X_M\}$.

Definitions of the Poisson Point Process

General Definition

Let D be a measurable space with measure μ. // e.g. $D = \mathbb{R}^2$ and $\mu =$"area"
The Poisson point process with parameter $\lambda \in \mathbb{R}_{\geq 0}$ is a random set $P \subseteq D$ such that

1. $|P \cap B| \sim \text{Pois}(\lambda\mu(B))$ for any $B \subseteq D$ with $\mu(B) < \infty$
2. $|P \cap B_1|$ and $|P \cap B_2|$ are independent whenever $B_1 \cap B_2 = \varnothing$

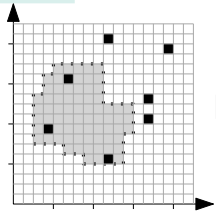


Generally:
$|B_1 \cap P| \sim \text{Po}(3\lambda)$
$|B_2 \cap P| \sim \text{Po}(\frac{1}{2}\lambda)$

This outcome:
$|B_1 \cap P| = 2$
$|B_2 \cap P| = 0$

Construction as a limit

- subdivide D into pieces of measure ε
- let each piece contain a point with probability $\varepsilon\lambda$
- consider the limit for $\varepsilon \to 0$



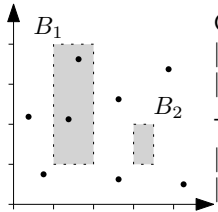
$$|B \cap P| \sim \text{Bin}(\mu(B)/\epsilon, \lambda\epsilon) \xrightarrow{\epsilon \to 0} \text{Po}(\lambda\mu(B))$$

Definitions of the Poisson Point Process

General Definition

Let D be a measurable space with measure μ. // e.g. $D = \mathbb{R}^2$ and $\mu =$"area"
The Poisson point process with parameter $\lambda \in \mathbb{R}_{\geq 0}$ is a random set $P \subseteq D$ such that

1. $|P \cap B| \sim \text{Pois}(\lambda\mu(B))$ for any $B \subseteq D$ with $\mu(B) < \infty$
2. $|P \cap B_1|$ and $|P \cap B_2|$ are independent whenever $B_1 \cap B_2 = \varnothing$

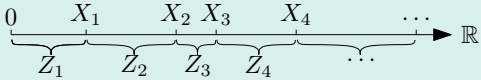


Generally:
$|B_1 \cap P| \sim \text{Po}(3\lambda)$
$|B_2 \cap P| \sim \text{Po}(\frac{1}{2}\lambda)$

This outcome:
$|B_1 \cap P| = 2$
$|B_2 \cap P| = 0$

Equivalent Definition if $D = \mathbb{R}_{\geq 0}$ (where μ is the Borel measure)

- sample $Z_1, Z_2, \ldots \sim \text{Exp}(\lambda)$
- define $X_i = \sum_{j=1}^{i} Z_j$



Then $P \overset{\text{d}}{=} \{X_1, X_2, \ldots\}$.

Proof idea: $\Pr[\min P > t] = \Pr[|P \cap [0,t]| = 0] = \Pr_{X \sim \text{Pois}(\lambda t)}[X = 0] = e^{-\lambda t} \overset{\text{def}}{=} \Pr[Z_1 > t]$.

Conclusion

Coupling

- embedding of two random variables X and Y into a common probability space
- relationships between distributions of X and Y become visible as relationships between outcomes of X' and Y'

Balls into Bins

- standard language when m objects are randomly assigned to n other objects

Poissonisation

- the act of replacing multinomially distributed $(L_1, \ldots, L_n)$ with independent Poisson random variables $(L'_1, \ldots, L'_n)$
- often results in model with nicer mathematical properties
- often formally justifiable

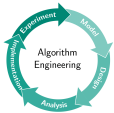
Poisson Point Process

- important model where points from a continuous space occur independently from each other with fixed density λ

Anhang: Mögliche Prüfungsfragen I

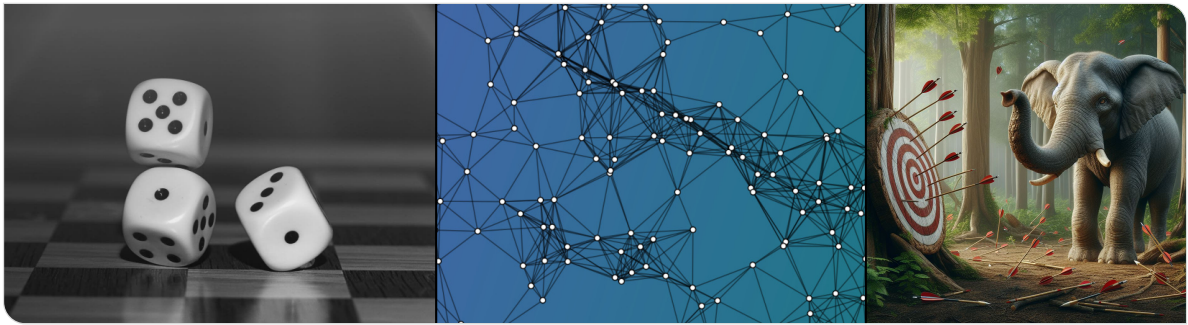
- Was ist ein Coupling?
  - Nenne Beispiele in denen ein Coupling nützlich sein kann.
  - Was bedeutet Gleichheit in Verteilung?
- Wo in der Vorlesung haben wir (implizit oder explizit) Balls-into-Bins-Prozesse betrachtet?
- Poissonisierung:
  - Welche lästige Eigenschaft hat die Verteilung der Beladungen in Balls-into-Bins Prozessen? Was ist in einem poissonisierten Modell anders?
  - Wie lässt sich in einem Balls-into-Bins Setting die Poissonverteilung wiederfinden?
  - Wie haben wir das poissonisierte und das reguläre Balls-into-Bins-Modell miteinander in Verbindung gebracht? Inwiefern lässt sich ein Wechsel zwischen den Modellen formal rechtfertigen?
- Poisson-Punktprozesse
  - Wie sind Poisson-Punktprozesse definiert?

Karlsruhe Institute of Technology



Probability and Computing – Approximation Algorithms

Stefan Walzer | WS 2024/2025



Lecture Notes by Worsch

This lecture's content is covered in Thomas Worsch's notes from 2019.

Content

1. What is Randomised Approximation?

2. Approximately counting satisfying assignments for Boolean formulas

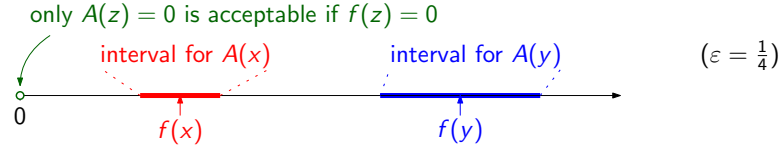
Randomised Approximate Counting

Definition

A randomised algorithm A *approximates* a quantity $f(x)$ if for any input x the output $A(x)$ satisfies:

$$\Pr[|A(x) - f(x)| \leq \varepsilon \cdot f(x)] \geq 1 - \delta.$$

The parameters are the *relative error* ε and the *failure probability* δ.



Remark: Related Complexity Classes

PRAS. Problems admitting A with running time polynomial in $|x|$, but not necessarily in $\frac{1}{\varepsilon}$ (for $\delta = 1/4$).

FPRAS. Problems admitting A with running time polynomial in $|x|$ and $\frac{1}{\varepsilon}$ (for $\delta = 1/4$).

Note: Also defined where $f(x)$ is not a *number*. For instance: Want to compute a *vertex cover* with a size close to optimal.

Counting Satisfiable Assignments of Boolean Formulas

A counting problem

For Boolean formula $B(x_1, \ldots, x_n)$ let $\#B$ be the number of satisfying assignments:

$$\#B = |\{(x_1, \ldots, x_n) \in \{0,1\}^n \mid B(x_1, \ldots, x_n) = 1\}|.$$

Example

$$B = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_3)$$
$$\#B = |\{(0,0,0), (0,0,1), (1,0,1), (1,1,1)\}| = 4$$

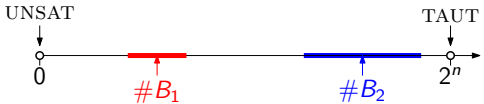
Approximation algorithm for $\#B$ in general? Unlikely.

Assume A satisfies $\Pr[|A(B) - \#B| \leq \varepsilon(\#B)] \leq 1 - \delta$ for $\varepsilon = \frac{1}{2}$ and $\delta = \frac{1}{4}$. Then

$$B \text{ is UNSAT} \Leftrightarrow \#B = 0 \Leftrightarrow \Pr[A(B) = 0] \geq \tfrac{3}{4}$$
$$B \text{ is SAT} \Leftrightarrow \#B > 0 \Leftrightarrow \Pr[A(B) > 0] \geq \tfrac{3}{4}$$

If A is polynomial time then A is BPP algorithm for SAT.
Then SAT $\in$ BPP and NP $\subseteq$ BPP. Hard to believe...

What could be a tractable special case?



We must distinguish: B is UNSAT $\Leftrightarrow \#B = 0$ from B is SAT $\Leftrightarrow \#B \geq 1$.
We need not distinguish: B is TAUT $\Leftrightarrow \#B = 2^n$ from B is NON-TAUT $\Leftrightarrow \#B < 2^n$.

CNF is hard on the wrong end

$$(x_1 \vee \bar{x_2} \vee x_{42}) \wedge \ldots \wedge (\bar{x_1} \vee x_3 \vee \bar{x_{37}})$$

Deciding UNSAT is NP-hard.
Deciding TAUT is easy.
↪ only empty CNF-formula is TAUT.

DNF looks good

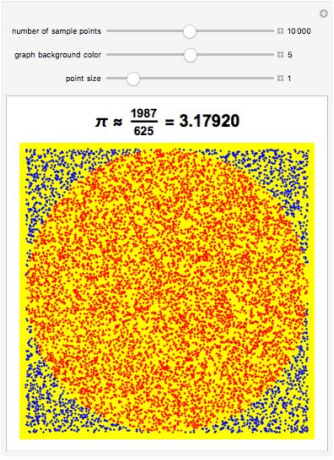
$$(\bar{x_1} \wedge x_2 \wedge x_{42}) \vee \ldots \vee (x_1 \wedge \bar{x_3} \wedge x_{37})$$

Deciding UNSAT is easy.
↪ only empty DNF-formula is UNSAT.
Deciding TAUT is hard.

Goal: Approximate $\#B$ for DNF formula B.

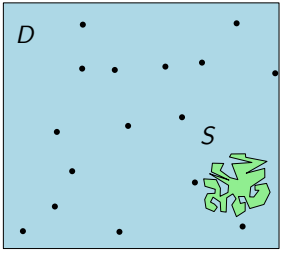
(Equivalently: Approximate $2^n - \#B$ for CNF formula B.)

Intuition: Approximating π



https://demonstrations.wolfram.com/ApproximatingPiByTheMonteCarloMethod/

Intuition: Approximate $|S|$ for $S \subseteq D$ by sampling from D



$|S| \approx |D| \cdot \frac{0}{16}$

Requirements

For this to work we must be able to

1. compute the size of D
2. sample uniformly from D
3. decide for $x \in D$ whether $x \in S$

soft requirement:

4. $\frac{|S|}{|D|}$ should not be too small

Approximate $|S|$ for $S \subseteq D$ by sampling from D

Algorithm approxSetSize**:**

hits $\leftarrow 0$
for $i = 1$ *to* N **do**
  sample $x \sim \mathcal{U}(D)$
  hits $\leftarrow$ hits $+ \mathbb{1}_{x \in S}$
return $\frac{\text{hits}}{N} \cdot |D|$

Chernoff

For $\varepsilon \in (0,1)$ and $X \sim \text{Bin}(N,p)$:

$$\Pr[|X - \mathbb{E}[X]| > \varepsilon \mathbb{E}[X]] < 2\exp(-\varepsilon^2 \mathbb{E}[X]/3).$$

Simple Theorem

Let D be a finite set and $S \subseteq D$ such that we can efficiently

1. compute $|D|$
2. sample uniformly from D
3. decide for $x \in D$ whether $x \in S$

Let $p = |S|/|D|$. Then approxSetSize with $N = \frac{3\log(2/\delta)}{\varepsilon^2 p}$ approximates $|S|$ with relative error ε and failure probability δ.

$\hookrightarrow$ Special Case $\varepsilon, \delta = \Theta(1)$: Need $N = \Omega(1/p)$ samples.

Proof: Apply Chernoff to hits $\sim \text{Bin}(N,p)$.

$$\begin{aligned}\Pr[\text{fail}] &= \Pr[|\text{result} - |S|| > \varepsilon|S|] = \Pr[|\tfrac{\text{hits}}{N} \cdot |D| - |S|| > \varepsilon|S|] = \Pr[|\text{hits} - \tfrac{|S|}{|D|}N| > \varepsilon\tfrac{|S|}{|D|}N] \\ &= \Pr[|\text{hits} - pN| > \varepsilon pN] = \Pr[|\text{hits} - \mathbb{E}[\text{hits}]| > \varepsilon\mathbb{E}[\text{hits}]] \leq 2\exp(-\varepsilon^2\mathbb{E}[\text{hits}]/3) = 2\exp(-\varepsilon^2 pN/3) = \delta.\end{aligned}$$

Approximate $|S|$ for $S \subseteq D$ by sampling from D

Algorithm approxSetSize**:**

hits $\leftarrow 0$
for $i = 1$ *to* N **do**
  sample $x \sim \mathcal{U}(D)$
  hits $\leftarrow$ hits $+ \mathbb{1}_{x \in S}$
return $\frac{\text{hits}}{N} \cdot |D|$

Chernoff

For $\varepsilon \in (0,1)$ and $X \sim \text{Bin}(N,p)$:

$$\Pr[|X - \mathbb{E}[X]| > \varepsilon \mathbb{E}[X]] < 2\exp(-\varepsilon^2 \mathbb{E}[X]/3).$$

Simple Theorem

Let D be a finite set and $S \subseteq D$ such that we can efficiently

1. compute $|D|$
2. sample uniformly from D
3. decide for $x \in D$ whether $x \in S$

Let $p = |S|/|D|$. Then approxSetSize with $N = \frac{3\log(2/\delta)}{\varepsilon^2 p}$ approximates $|S|$ with relative error ε and failure probability δ.

$\hookrightarrow$ Special Case $\varepsilon, \delta = \Theta(1)$: Need $N = \Omega(1/p)$ samples.

Application to $\#B$

- $S =$ satisfying assignments of B
- $D = \{0,1\}^n$
- $p = \frac{|S|}{|D|} = \frac{\#B}{2^n}$
- We may have $p = 1/2^n$
- $N = \Omega(2^n)$ required
- :-(

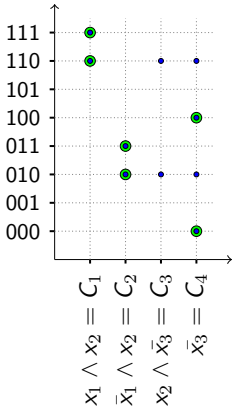
No Surprise

Of course this didn't work

Did not exploit that B is in DNF.

Approximating $\#B$ for B in DNF

Assume $B = C_1 \vee \ldots \vee C_m$ where C_i contains ℓ_i literals.



- $D_i := \{x \in \{0,1\}^n \mid C_i(x) = 1\}$ (satisfying assignments of C_i)
- $D := \{(i, x) \mid i \in [m], x \in D_i\}$ $(= D_1 \dot\cup \ldots \dot\cup D_m)$
- $S := \{(i, x) \mid i \in [m], x \in D_i, x \notin D_1 \cup \ldots D_{i-1}\}$

Claim: We can approximate $|S| = \#B$ by sampling from D

1. We can compute $|D_i| = 2^{n-\ell_i}$ and $|D| = \sum_{i=1}^{m} |D_i|$. ✓
2. We can efficiently sample $(I, X) \sim \mathcal{U}(D)$ ✓
   - sample I such that $\Pr[I = i] = \frac{|D_i|}{|D|}$
   - sample $X \sim \mathcal{U}(D_i)$
     ↪ set variables appearing in C_i as required, sample others from $\mathrm{Ber}(1/2)$.
   - Yields $\Pr[(I, X) = (i, x)] = \frac{|D_i|}{|D|} \cdot \frac{1}{|D_i|} = \frac{1}{|D|}$ for all $(i, x) \in D$.
3. We can efficiently decide "is $(i, x) \in S$?" ✓
   ↪ just plug x into all clauses $C_1, \ldots, C_i$ // time $\mathcal{O}(mn)$
4. $p = \frac{|S|}{|D|}$ satisfies $p \geq \frac{1}{m}$. ✓

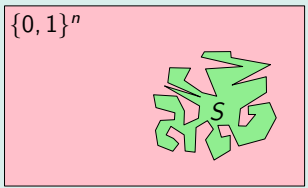
Takeaway

Theorem

If B is in DNF, then we can approximate $\#B$ in polynomial time (using $N = m \cdot \frac{3\log(2/\delta)}{\varepsilon^2}$ samples) with relative error ε and failure probability δ.

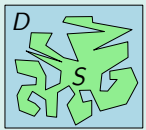
Intuition: Why did this work?

Naive strategy:



Problem: $|S|/|\{0,1\}^n|$ may be exponentially small

Improved strategy:



Advantage: $|S|/|D|$ is $\Omega(1/m)$.

Conclusion

Randomised Approximation is Powerful

For B in DNF:

- Computing $\#B$ *exactly* is $\#\boldsymbol{P}$-complete.
- no *deterministic approximation* algorithm for such problems is known
- we analysed an efficient *randomised approximation* algorithm

Anhang: Mögliche Prüfungsfragen

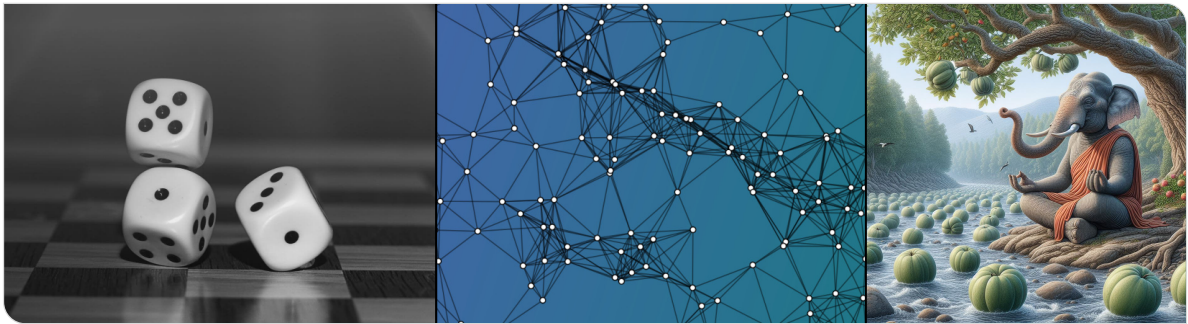
- Was ist ein randomisierter Approximationsalgorithmus (für ein Zählproblem)?
- Wir haben das Zählproblem $\#B$ für Boolesche Formeln betrachtet. Hatten wir im allgemeinen Fall Erfolg? Warum nicht?
- Welchen Spezialfall haben wir uns vorgenommen? Wieso tritt dort nicht das selbe Problem auf wie im allgemeinen Fall?
- Wir haben einen Algorithmus gesehen der für zwei Mengen $S \subseteq D$ die Größe von $|S|$ schätzt.
  - Unter welchen Annahmen ist dieser anwendbar?
  - Wie hat der Algorithmus funktioniert?
  - Wie hängt die Anzahl der nötigen samples von $|S|$ und $|D|$ ab?
- Um $\#B$ für DNF Formel B zu schätzen haben wir einen schlaueren Ansatz kennengerlernt.
  - Wie hat dieser funktioniert?
  - Wie vermeidet dieser das Problem das naiven Ansatzes?

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Streaming

Stefan Walzer | WS 2024/2025



Content

1. Definition: What is a Streaming Algorithm?

2. Morris' Algorithm for $F_1 = m$

3. The CVM Algorithm for $F_0 = |\{a_1 \ldots, a_m\}|$

4. Conclusion

What is a Streaming Algorithm?

- looong input data stream $(a_1, \ldots, a_m) \in [n]^m$ can only be read *once* from left to right
- goal: approximate some value $F = F(a_1, \ldots, a_m)$ with small relative error ε and failure probability δ.
  $\hookrightarrow$ streaming algorithms are approximation algorithms
- challenge: use less *space* than exact algorithm (in particular: cannot store $(a_1, \ldots, a_m)$).
  $\hookrightarrow$ don't care about running time

Formally, a streaming algorithm is given by three algorithms init, update and result used as follows:

$Z \leftarrow \text{init}()$
for $i = 1$ **to** m **do**
  $Z \leftarrow \text{update}(Z, a_i)$
return $\text{result}(Z)$

Its space complexity is the space required for Z.

Today's Motivating Examples

A Router approximately counts traffic over each connection.
  $\hookrightarrow$ maybe: detect anomalies related to DDoS

B Website approximately counts number of unique users visiting a resource.

Today's Formal Results

A Approximate $F_1(a_1, \ldots, a_m) = m$ in expected space $\frac{1}{\varepsilon^2 \delta} \log \log m$.

B Approximate $F_0(a_1, \ldots, a_m) = |\{a_1, \ldots, a_m\}|$ in expected space $\frac{1}{\varepsilon^2} \log(n) \cdot \log(m/\delta)$.

Content

1. Definition: What is a Streaming Algorithm?

2. Morris' Algorithm for $F_1 = m$

3. The CVM Algorithm for $F_0 = |\{a_1 \ldots, a_m\}|$

4. Conclusion

Attempt I: Naive Counting

Approximate Counting
- stream $(a_1, \ldots, a_m)$
- want $F_1 = m$

Karlsruhe Institute of Technology

Naive Counting

Algorithm init**:**
  $Z \leftarrow 0$
  return Z

Algorithm update(Z, a)**:**
  $Z \leftarrow Z + 1$
  return Z

Algorithm result(Z)**:**
  return Z

Observations on Naive counting

- No errors ($\varepsilon = \delta = 0$).
- Requires $\lceil \log(m+1) \rceil$ bits of memory.
- No *deterministic* algorithm can use less space
  - Would have to "reuse" a state Z.
  - Is then trapped in an infinite loop.
  - Result arbitrarily far off if m large enough.

Attempt II: Lossy Counting

Approximate Counting
- stream $(a_1, \ldots, a_m)$
- want $F_1 = m$

Karlsruhe Institute of Technology

Lossy Counting, parameter p

Algorithm init**:**
  $Z \leftarrow 0$
  return Z

Algorithm update(Z, a)**:**
  with probability p **do**
    $Z \leftarrow Z + 1$
  return Z

Algorithm result(Z)**:**
  return Z/p

Analysis (Exercise)

For any $p \in (0, 1]$ we have
- $\mathbb{E}[\text{result}] = m$
- $\Pr[|\text{result} - m| \leq \varepsilon m] \geq 1 - 2\exp(-\varepsilon^2 pm/3)$.
- $\mathbb{E}[\text{space}] \leq \log_2(1 + mp) + 1$.

Corollary

By choosing $p = \frac{3}{\varepsilon^2 m} \log(2/\delta)$ we get

$$\Pr[\text{fail}] \leq \delta \text{ and } \mathbb{E}[\text{space}] \leq \mathcal{O}(\log(\tfrac{1}{\varepsilon}) + \log\log(1/\delta))).$$

Serious Objection

Correctly choosing p requires already knowing m.

(or at least the order of magnitude of m)

Attempt III: Morris' Algorithm

Approximate Counting
- stream $(a_1, \ldots, a_m)$
- want $F_1 = m$

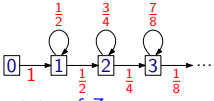
Karlsruhe Institute of Technology

Morris' Algorithm

Algorithm init**:**
  $Z \leftarrow 0$
  return Z

Algorithm update(Z, a)**:**
  with probability 2^{-Z} **do**
    $Z \leftarrow Z + 1$
  return Z

Algorithm result(Z)**:**
  return $2^Z - 1$



Lemma: Morris' Algorithm is an *Unbiased Estimator*

$\mathbb{E}[\text{result}] = m$.

Proof

Let Z_i for $i \in [m]$ denote the value of Z after i updates.
Consider the expected change to 2^Z in one step...

- ... conditioned on a current value $j \in \mathbb{N}$:

$$\mathbb{E}[2^{Z_{i+1}} - 2^{Z_i} \mid Z_i = j] = 2^{-j} \cdot (2^{j+1} - 2^j) + (1 - 2^{-j}) \cdot \underbrace{(2^j - 2^j)}_{=0} = 2 - 1 = 1.$$

- ... unconditionally:

$$\mathbb{E}[2^{Z_{i+1}} - 2^{Z_i}] \overset{\text{LTE}}{=} \sum_{j \geq 0} \Pr[Z_i = j] \cdot \underbrace{\mathbb{E}[2^{Z_{i+1}} - 2^{Z_i} \mid Z_i = j]}_{=1} = \sum_{j \geq 0} \Pr[Z_i = j] = 1.$$

Hence:

$$\mathbb{E}[\text{result}] = \mathbb{E}[2^{Z_m} - 1] = \mathbb{E}[2^{Z_m} - 2^{Z_0}] = \mathbb{E}[\sum_{i=1}^{m} 2^{Z_{i+1}} - 2^{Z_i}] = \sum_{i=1}^{m} \underbrace{\mathbb{E}[2^{Z_{i+1}} - 2^{Z_i}]}_{=1} = m.$$

Attempt III: Morris' Algorithm

Approximate Counting
- stream $(a_1, \ldots, a_m)$
- want $F_1 = m$

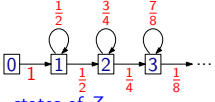
Karlsruhe Institute of Technology

Morris' Algorithm

Algorithm init:
  $Z \leftarrow 0$
  return Z

Algorithm update(Z, a):
  with probability 2^{-Z} **do**
    $Z \leftarrow Z + 1$
  return Z

Algorithm result(Z):
  return $2^Z - 1$



states of Z
transition probabilities

Lemma 1: Worryingly large Variance

$\mathrm{Var}(2^{Z_i}) = \frac{i^2-i}{2} = \Theta(i^2)$.

Lemma 2

$\mathbb{E}[2^{2Z_i}] = \frac{3i(i+1)}{2} + 1$.

Proof of Lemma 1 using Lemma 2.

$$\mathrm{Var}(2^{Z_i}) = \mathbb{E}[2^{2Z_i}] - \mathbb{E}[2^{Z_i}]^2 \overset{\text{Lem. 2}}{=} \frac{3i(i+1)}{2} + 1 - (i+1)^2$$
$$= \frac{3i(i+1) + 2 - 2i^2 - 4i - 2}{2} = \frac{i^2 - i}{2}. \quad \square$$

Attempt III: Morris' Algorithm

Approximate Counting
- stream $(a_1, \ldots, a_m)$
- want $F_1 = m$

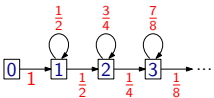
Karlsruhe Institute of Technology

Morris' Algorithm

Algorithm init:
  $Z \leftarrow 0$
  return Z

Algorithm update(Z, a):
  with probability 2^{-Z} **do**
    $Z \leftarrow Z + 1$
  return Z

Algorithm result(Z):
  return $2^Z - 1$



Lemma 1: Worryingly large Variance

$\mathrm{Var}(2^{Z_i}) = \frac{i^2-i}{2} = \Theta(i^2)$.

Lemma 2

$\mathbb{E}[2^{2Z_i}] = \frac{3i(i+1)}{2} + 1$.

Proof of Lemma 2.

For $i \in \{0, 1\}$ ✓. Let now $i \geq 1$. Note $\Pr[Z_{i+1} = 0] = \Pr[Z_i = 0] = 0$.

$$\begin{aligned}
\mathbb{E}[2^{2Z_{i+1}}] &= \sum_{j\geq 1} 2^{2j} \Pr[Z_{i+1} = j] = \sum_{j\geq 1} 2^{2j} (\Pr[Z_i = j-1] \cdot 2^{-j+1} + \Pr[Z_i = j] \cdot (1 - 2^{-j})) \\
&= \sum_{j\geq 1} 2^{j+1} \Pr[Z_i = j-1] + \sum_{j\geq 1} 2^{2j} \Pr[Z_i = j] - \sum_{j\geq 1} 2^j \Pr[Z_i = j] \\
&= 4 \sum_{j\geq 0} 2^j \Pr[Z_i = j] + \sum_{j\geq 0} 2^{2j} \Pr[Z_i = j] - \sum_{j\geq 0} 2^j \Pr[Z_i = j] \\
&= 4\mathbb{E}[2^{Z_i}] + \mathbb{E}[2^{2Z_i}] - \mathbb{E}[2^{Z_i}] = 3\mathbb{E}[2^{Z_i}] + \mathbb{E}[2^{2Z_i}] = 3(i+1) + \mathbb{E}[2^{2Z_i}] \\
&\overset{\text{Ind.}}{=} 3(i+1) + \frac{3i(i+1)}{2} + 1 = \frac{3(i+2)(i+1)}{2} + 1. \quad \square
\end{aligned}$$

Space

Expected Space

$$\mathbb{E}[\text{space}] \leq \mathbb{E}[\lceil \log_2(1+Z_m) \rceil] \leq 1 + \mathbb{E}[\log_2(1+Z_m)] = 1 + \mathbb{E}[\log_2(1+\log_2(2^{Z_m}))]$$
$$\overset{(\star)}{\leq} 1 + \log_2(1+\log_2(\mathbb{E}[2^{Z_m}])) = 1 + \log_2(1+\log_2(m+1)) = \Theta(\log\log m).$$

($\star$) uses Jensen's inequality that you'll prove as an exercise.

Interim Conclusion: Morris is not good enough *yet*

- $\mathbb{E}[\text{result}] = m$ ✓ unbiased estimator
- $\mathbb{E}[\text{space}] = \mathcal{O}(\log\log m)$ ✓ highly space efficient
- $\text{Var}(\text{result}) = \Theta(m^2)$ ✗
  - Standarddeviation $\Theta(m)$
    $\rightsquigarrow$ right order of magnitude, but not better.

Morris$^+$: Use many copies of Morris' Algorithm

Theorem

Consider a streaming algorithm that maintains a sequence $Z = (Z_1, \dots, Z_s)$ of independent Morris-counters and returns $\text{result}(Z) := \frac{\text{result}(Z_1)+\cdots+\text{result}(Z_s)}{s}$. For $s = \frac{1}{\varepsilon^2\delta}$ we obtain

- $\mathbb{E}[\text{result}(Z)] = m$ and $\mathbb{E}[\text{space}] = \mathcal{O}(\frac{1}{\varepsilon^2\delta} \log\log m)$
- $\Pr[|\text{result}(Z) - m| \leq \varepsilon m] = 1 - \mathcal{O}(\delta)$.

Reminder on Variance

If X, Y are independent random variables and $s > 0$ then

- $\text{Var}(sX) = s^2 \text{Var}(X)$
- $\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y)$

Proof of Concentration using Chebyshev

$$\text{Var}(\text{result}(Z)) = \text{Var}(\tfrac{1}{s}\sum_{i=1}^{s} \text{result}(Z_i)) = \tfrac{1}{s^2}\text{Var}(\sum_{i=1}^{s} \text{result}(Z_i))$$
$$= \tfrac{1}{s^2}\sum_{i=1}^{s} \text{Var}(\text{result}(Z_i)) = \tfrac{s}{s^2}\text{Var}(\text{result}(Z_1)) = \tfrac{1}{s}\Theta(m^2) = \Theta(m^2/s).$$

Chebyschev:

$\Pr[X - \mathbb{E}[X] > c] \leq \frac{\text{Var}(X)}{c^2}$.

$$\Pr[\text{fail}] = \Pr[|\text{result}(Z) - m| > \varepsilon m] = \Pr[|\text{result}(Z) - \mathbb{E}[\text{result}(Z)]| > \varepsilon m] \leq \frac{\text{Var}(\text{result}(Z))}{\varepsilon^2 m^2} = \Theta(1/(\varepsilon^2 s)) = \Theta(\delta). \quad \square$$

Morris*: Use a different base in Morris' Algorithm

Morris with base $1 < \rho \ll 2$

- In every update: increment Z with probability ρ^{-Z}.
- In the end: return $\rho^Z - 1$.

Modified Analysis

Show similarly to before:

- $\mathbb{E}[\text{result}] = m$
- $\text{Var}(\text{result}) = \Theta((\rho - 1)m^2)$

Choosing $\rho = 1 + \varepsilon^2\delta$ gives:

- $\Pr[|\text{result} - m| > \varepsilon m] = \mathcal{O}(\delta)$.
- $\mathbb{E}[\text{space}] = \mathcal{O}(\log\log m + \log \frac{1}{\delta\varepsilon})$.

Content

1. Definition: What is a Streaming Algorithm?

2. Morris' Algorithm for $F_1 = m$

3. The CVM Algorithm for $F_0 = |\{a_1 \ldots, a_m\}|$

4. Conclusion

History

Counting Distinct Elements

- stream $(a_1, \ldots, a_m) \in [n]^m$
- want $F_0 = |\{a_1, \ldots, a_m\}|$

Karlsruhe Institute of Technology

Remark: CVM is not well-known

Popular line of algorithms for F_0 by Philippe Flajolet et al:

- ~~1984: Flajolet-Martin~~ (deprecated)
  ↪ https://en.wikipedia.org/wiki/Flajolet-Martin_algorithm
- ~~2003: LogLog~~ (deprecated)
- 2007: HyperLogLog
  ↪ `https://en.wikipedia.org/wiki/HyperLogLog`

The CVM-Algorithm

- 2022: European Symposium on *Simplicity* in Algorithms 2022
  ↪ „Distinct Elements in Streams: An Algorithm for the (Text) Book“
- is a bit worse than HyperLogLog
- is easier to analyse than HyperLogLog

Next: We develop CVM in three steps.

Attempt I: Naively storing the set

Naive Storing

Algorithm init**:**
  $Z \leftarrow \varnothing$
  return Z

Algorithm update(Z, a)**:**
  $Z \leftarrow Z \cup \{a\}$
  return Z

Algorithm result(Z)**:**
  return $|Z|$

Observation

Naively storing the set requires $\Omega(F_0 \cdot \log n)$ bits.

Attempt II: Storing the set lossily

Counting Distinct Elements
- stream $(a_1, \ldots, a_m) \in [n]^m$
- want $F_0 = |\{a_1, \ldots, a_m\}|$

Karlsruhe Institute of Technology

LossyStore, parameter p

Algorithm init**:**
  $Z \leftarrow \varnothing$
  return Z

Algorithm update(Z, a)**:**
  $Z \leftarrow Z \setminus \{a\}$
  with probability p **do**
    $Z \leftarrow Z \cup \{a\}$
  return Z

Algorithm result(Z)**:**
  return $|Z|/p$;

Chernoff for $X \sim \text{Bin}(n, p)$

$\Pr[|X - \mathbb{E}[X]| > \varepsilon\mathbb{E}[X]] \leq 2\exp(-\varepsilon^2\mathbb{E}[X]/3)$.

Analysis

Let $Z_0, \ldots, Z_m$ be the states of Z over time. Invariant: Each $a \in \{a_1, \ldots, a_i\}$ is in Z_i independently with probability p. Hence $|Z_m| \sim \text{Bin}(F_0, p)$.

- $\mathbb{E}[\text{result}] = \mathbb{E}[|Z_m|/p] = \mathbb{E}[|Z_m|]/p = F_0 p/p = F_0$.
  $\hookrightarrow$ result is *unbiased estimator* of F_0.
- $\Pr[\text{fail}] = \Pr[|\text{result} - F_0| > \varepsilon F_0] = \Pr\big[\big||Z_m|/p - F_0\big| > \varepsilon F_0\big]$
  $= \Pr\big[\big||Z_m| - pF_0\big| > \varepsilon pF_0\big] = \Pr\big[\big||Z_m| - \mathbb{E}[|Z_m|]\big| > \varepsilon\mathbb{E}[|Z_m|]\big]$
  $\overset{\text{Chern.}}{\leq} 2\exp(-\varepsilon^2\mathbb{E}[|Z_m|]/3) = 2\exp(-\varepsilon^2 pF_0/3)$.
  $\hookrightarrow$ choose $p = p_\delta := \frac{3\log(2/\delta)}{\varepsilon^2 F_0}$ for $\Pr[\text{fail}] \leq \delta$.
- Expected space *in the end* for $p = p_\delta$ (⚠ $\neq$ peak space consumption)
  $\mathbb{E}[|Z_m| \cdot \mathcal{O}(\log n)] = F_0 p_\delta \cdot \mathcal{O}(\log n) = \mathcal{O}(\frac{\log(1/\delta)}{\varepsilon^2}\log n)$.

Serious Objection: Need to know F_0 to choose p

- for $p \gg p_\delta$: space is wasted
- for $p \ll p_\delta$: failure becomes likely

Attempt III: Adjust lossiness dynamically

Counting Distinct Elements
- stream $(a_1, \dots, a_m) \in [n]^m$
- want $F_0 = |\{a_1, \dots, a_m\}|$

Karlsruhe Institute of Technology

CVM, parameter T

Algorithm init**:**
- $Z \leftarrow \varnothing$
- $P \leftarrow 1$
- **return** (P, Z)

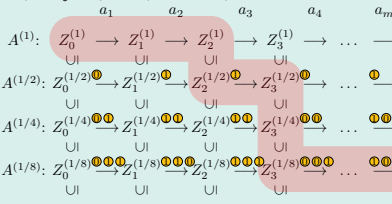
Algorithm update$((P, Z), a)$**:**
- $Z \leftarrow Z \setminus \{a\}$
- **with probability** P **do**
  - $Z \leftarrow Z \cup \{a\}$
- **while** $|Z| \geq T$ **do** // shrink
  - $Z' \leftarrow \varnothing$
  - **for** $a \in Z$ **do**
    - **with probability** $1/2$ **do**
      - $Z' \leftarrow Z' \cup \{a\}$
  - $(Z, P) \leftarrow (Z', P/2)$
- **return** (P, Z)

Algorithm result$((P, Z))$**:**
- **return** $|Z|/P$

CVM behaves like LossyStore with dynamic p

Consider $A^{(p)} := \text{LossyStore}(p)$ with states $Z_0^{(p)}, \dots, Z_m^{(p)}$ for $p \in \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots\}$.

Let $(P_0, Z_0^{(\text{CVM})}), \dots, (P_m, Z_m^{(\text{CVM})})$ be the state of CVM.



Intuition: The path of CVM:

$(x, y) \leftarrow (0, 0)$ // top left
for $i = 1$ **to** m **do** // m updates
- $x \leftarrow x + 1$ // go right
- **while** $|Z_x^{(2^{-y})}| \geq T$ **do**
  - $y \leftarrow y + 1$ // go down

final state is $Z_m^{(2^{-y})}$

Coupling between executions of $A^{(p)}$ and CVM:
- $A^{(p/2)}$ uses coin tosses of $A^{(p)}$ and one more.
  "$A^{(p/2)}$ keeps half of what $A^{(p)}$ keeps."
- CVM uses coin tosses of $A^{(P)}$ to process elements.
- When shrinking, CVM inspects past coin tosses done by $A^{(P/2)}$. (the next unused coin for all $a \in Z$)

Effects of the coupling:
- $Z_j^{(\text{CVM})} = Z_j^{(P_j)}$ for $j \in [m]$
- $\text{result}^{(\text{CVM})} = \text{result}^{(P_m)}$
- $\text{fail}^{(\text{CVM})} = \text{fail}^{(P_m)}$

Attempt III: Adjust lossiness dynamically

Counting Distinct Elements
- stream $(a_1, \ldots, a_m) \in [n]^m$
- want $F_0 = |\{a_1, \ldots, a_m\}|$

Karlsruhe Institute of Technology

CVM, parameter T

Algorithm init:
- $Z \leftarrow \varnothing$
- $P \leftarrow 1$
- **return** (P, Z)

Algorithm update($(P, Z), a$):
- $Z \leftarrow Z \setminus \{a\}$
- **with probability** P **do**
  - $Z \leftarrow Z \cup \{a\}$
- **while** $|Z| \geq T$ **do** // shrink
  - $Z' \leftarrow \varnothing$
  - **for** $a \in Z$ **do**
    - **with probability** $1/2$ **do**
      - $Z' \leftarrow Z' \cup \{a\}$
  - $(Z, P) \leftarrow (Z', P/2)$
- **return** (P, Z)

Algorithm result((P, Z)):
- **return** $|Z|/P$

Lemma: Failure Probability and Space

With $T = \frac{18 \log_2(2m/\delta)}{\varepsilon^2}$ we get $\Pr[\text{fail}^{\text{CVM}}] = \mathcal{O}(\delta)$ and $\text{space}^{\text{CVM}} = \mathcal{O}(\frac{\log(m/\delta)}{\varepsilon^2} \log n) + \lceil \log_2(\log_2(1/P_m)) \rceil$.

Analysis of CVM's failure probability (a bit sketchy)

- Recall: LossyStore$(p_\delta = \frac{3 \log(2/\delta)}{\varepsilon^2 F_0})$ has failure probability $\leq \delta$. Assume p_δ is power of 2.
- Then $\Pr[\text{fail}^{(p_\delta)}] \leq \delta$, $\Pr[\text{fail}^{(2p_\delta)}] \leq \delta^2$, $\Pr[\text{fail}^{(4p_\delta)}] \leq \delta^4$, ...
- Therefore $\Pr[\text{fail}^{(1)}] + \ldots + \Pr[\text{fail}^{(2p_\delta)}] + \Pr[\text{fail}^{(p_\delta)}] \leq \ldots + \delta^8 + \delta^4 + \delta^2 + \delta = \mathcal{O}(\delta)$.
-
$$\Pr[P_m < p_\delta] = \Pr[|Z_j^{(p_\delta)}| \geq T \text{ for some } j \in [m]] \leq m \cdot \Pr[|Z_m^{(p_\delta)}| \geq T]$$
$$= m \cdot \Pr_{Z \sim \text{Bin}(F_0, p_\delta)}[Z \geq T] \overset{\Delta}{\leq} m \cdot 2^{-T} \leq m \cdot 2^{-\log(m/\delta)} = \delta.$$
where Δ uses a Chernoff bound and $6\mathbb{E}[Z] = 6F_0 p_\delta = \frac{18 \log_2(2/\delta)}{\varepsilon^2} \leq T$.
- $\text{fail}^{\text{CVM}} \Leftrightarrow \text{fail}^{(P_m)} \Rightarrow \left(P_m < p_\delta \vee \text{fail}^{(1)} \vee \text{fail}^{(2)} \vee \ldots \vee \text{fail}^{(p_\delta)}\right)$

$$\text{Finally: } \Pr[\text{fail}^{\text{CVM}}] \leq \Pr[P_m < p_\delta \vee \text{fail}^{(1)} \vee \text{fail}^{(2)} \vee \ldots \vee \text{fail}^{(p_\delta)}] \overset{\text{UB}}{\leq} \delta + \mathcal{O}(\delta) = \mathcal{O}(\delta).$$

Conclusion

Streaming Algorithms

- Input read only once, from left to right.
- Goal: Use little space. (less than what is needed to store input stream)
- Motivation: Network actor wants to maintain statistic on traffic.

Morris$^+$ Algorithm for Counting the Stream Length

- approximation in space $\mathcal{O}(\frac{1}{\varepsilon^2\delta}\log\log m)$ // or $\mathcal{O}(\log\log m + \log\frac{1}{\delta\varepsilon})$ using Morris*?
  (ε = relative error, δ = failure probability)
- deterministic algorithms need space $\lceil\log(1+m)\rceil$

CVM Algorithm for Counting *Distinct* Elements

- approximation in space $\mathcal{O}(\frac{1}{\varepsilon^2}\log(n)\log(m/\delta))$

Anhang: Mögliche Prüfungsfragen I

- Definition Streamingalgorithmen:
  - Was ist die Aufgabe eines Streamingalgorithmus (in Bezug auf eine Größe $F = F(a_1, \ldots, a_m)$)?
  - Was ist die spezifische Herausforderung für Streamingalgorithmen?
- Streamingalgorithmen für $F_1 = m$:
  - Was könnte ein Anwendungsfall sein, in dem man F_1 schätzen möchte?
  - Wie viel Speicher braucht man wenn man einfach nur zählt? Kann ein deterministischer Algorithmus etwas Schlaueres machen?
  - Wie funktioniert der LossyCounting Algorithmus? Warum hilft dieser uns nicht weiter?
  - Wie funktioniert Morris' Algorithmus?
  - Beweise, dass Morris' Algorithmus erwartungstreu ist.*
  - Beweise, dass der Speicherbedarf von Morris doppelt logarithmisch in m ist.
  - Welche Schwäche hatte Morris' Algorithmus noch und wie haben wir diese behoben?

Anhang: Mögliche Prüfungsfragen II

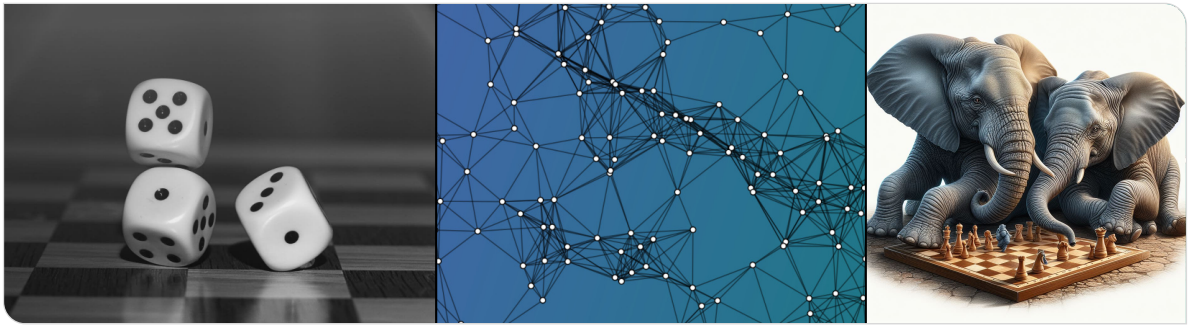
- Streamingalgorithmen für $F_0 = \{a_1, \ldots, a_m\}$:
  - Was könnte ein Anwendungsfall sein, in dem man F_0 schätzen möchte?
  - Wie viel Speicher braucht der naive deterministische Algorithmus? Was können wir mit CVM erreichen?
  - Als Zwischenschritt haben wir den Algorithmus LossyStore formuliert. Wie funktioniert dieser?
  - Wie funktioniert der CVM Algorithmus? Wie steht dieser mit dem LossyStore Algorithmus in Verbindung?
  - In der Analyse der Fehlerwahrscheinlichkeit von CVM haben wir zwei Arten von Problemen unterschieden. Welche?*

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Game Theory & Yao's Principle

Stefan Walzer | WS 2024/2025



Lecture Notes by Worsch

Some of this lecture's content is covered in Thomas Worsch's notes from 2019.

Content

1. Nash Equilibria in 2-Player Zero-Sum Games

- Games and Nash Equilibria
- Two Player Zero Sum Games
- Loomis' Theorem for Two-Player Zero Sum Games

2. Yao's Minimax Principle

3. Applications of Yao's Principle

- Evaluation of $\overline{\wedge}$-Trees
  - Proof Sketch of Tarsi's Theorem (nicht prüfungsrelevant)
- The Ski-Rental Problem

4. Conclusion

Content

1. Nash Equilibria in 2-Player Zero-Sum Games
- Games and Nash Equilibria
- Two Player Zero Sum Games
- Loomis' Theorem for Two-Player Zero Sum Games

2. Yao's Minimax Principle

3. Applications of Yao's Principle
- Evaluation of $\overline{\wedge}$-Trees
  - Proof Sketch of Tarsi's Theorem (nicht prüfungsrelevant)
- The Ski-Rental Problem

4. Conclusion

Content

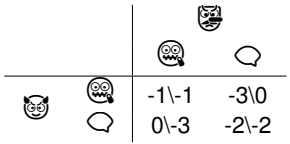
1. Nash Equilibria in 2-Player Zero-Sum Games
- Games and Nash Equilibria
- Two Player Zero Sum Games
- Loomis' Theorem for Two-Player Zero Sum Games

2. Yao's Minimax Principle

3. Applications of Yao's Principle
- Evaluation of $\overline{\wedge}$-Trees
  - Proof Sketch of Tarsi's Theorem (nicht prüfungsrelevant)
- The Ski-Rental Problem

4. Conclusion

Prisoner's Dilemma



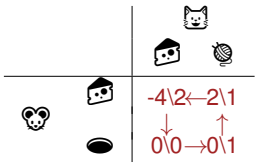
Setting

- *strategies* and available to both *players*
- table shows *payoffs* for players depending on chosen strategies
- here: always better to choose
  ↪ pair (,) is unique *equilibrium*

Definition: Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

A cat and mouse game



Someone always regrets their decision

		reaction
🧀	🧀	🐭 should have played 🕳
🕳	🧀	🐱 should have played 🧶
🕳	🧶	🐭 should have played 🧀
🧀	🧶	🐱 should have played 🧀

↪ No combination of *pure* strategies is an *equilibrium*.

Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

Nash Equilibria

What a Game *is*

- Finite sets S_1, S_2 of *pure strategies*.
- Utility functions $u_1, u_2 : S_1 \times S_2 \to \mathbb{R}$.

How a Game is played

- Players pick a strategy simultaneously
  $\hookrightarrow$ gives pair $(s_1, s_2) \in S_1 \times S_2$.
- player 1 gets payoff $u_1(s_1, s_2)$ and
  player 2 gets payoff $u_2(s_1, s_2)$.

Existence of Mixed-Strategy Nash Equilibria

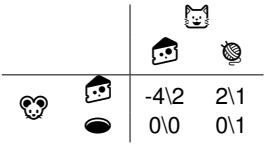
There exist distributions $\mathcal{S}_1^*$ on S_1 and $\mathcal{S}_2^*$ on S_2, called *mixed strategies* such that $(\mathcal{S}_1^*, \mathcal{S}_2^*)$ is an equilibrium:

player 1 cannot increase expected payoff: $\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[u_1(s_1, s_2)] = \max_{s_1 \in S_1} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[u_1(s_1, s_2)]$.

player 2 cannot increase expected payoff: $\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[u_2(s_1, s_2)] = \max_{s_2 \in S_2} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[u_2(s_1, s_2)]$.

Remark: Theorem holds for $n \geq 3$ players as well.

Nash Equilibrium in Cat & Mouse Game



Equilibrium

$\mathcal{S}_{\text{mouse}} = \{\text{cheese} : \frac{1}{2}, \text{hole} : \frac{1}{2}\}$

$\mathcal{S}_{\text{cat}} = \{\text{cheese} : \frac{1}{3}, \text{yarn} : \frac{2}{3}\}$

Verification of Equilibrium Property: Calculating Expected Payoffs

for mouse:

- playing cheese gives expected payoff $\frac{1}{3} \cdot (-4) + \frac{2}{3} \cdot 2 = 0$
- playing hole gives expected payoff $\frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 0 = 0$
- playing $\mathcal{S}_{\text{mouse}}$ is a mix of both
  $\hookrightarrow$ also expected payoff 0.

for cat:

- playing cheese gives expected payoff $\frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 0 = 1$
- playing yarn gives expected payoff $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 = 1$
- playing $\mathcal{S}_{\text{cat}}$ is a mix of both
  $\hookrightarrow$ also expected payoff 1.

Content

1. Nash Equilibria in 2-Player Zero-Sum Games
- Games and Nash Equilibria
- Two Player Zero Sum Games
- Loomis' Theorem for Two-Player Zero Sum Games

2. Yao's Minimax Principle

3. Applications of Yao's Principle
- Evaluation of $\overline{\wedge}$-Trees
  - Proof Sketch of Tarsi's Theorem (nicht prüfungsrelevant)
- The Ski-Rental Problem

4. Conclusion

Two Player Zero Sum Games and their Matrix Formulation

- Finite sets of pure strategies
  - S_1 for player 1
  - S_2 for player 2
- utility function $u : S_1 \times S_2 \to \mathbb{R}$
  - player 1 gets $u(s_1, s_2)$
  - player 2 gets $-u(s_1, s_2)$

- Implicit sets of pure strategies
  - $S_1 = [n]$ for the *row player*
  - $S_2 = [m]$ for the *column players*
- matrix $M \in \mathbb{R}^{n \times m}$
  - row player gets M_{s_1, s_2}
  - column player gets $-M_{s_1, s_2}$

		0	-1	1
		1	0	-1
		-1	1	0

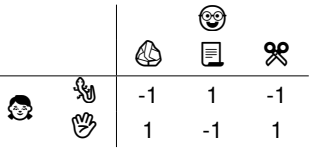
Unique equilibrium of

$\mathcal{S}_1 = \mathcal{S}_2 = \{ : \frac{1}{3}, : \frac{1}{3}, : \frac{1}{3}\}$

Two Player Zero Sum Games and their Matrix Formulation

- Finite sets of pure strategies
  - S_1 for player 1
  - S_2 for player 2
- utility function $u : S_1 \times S_2 \to \mathbb{R}$
  - player 1 gets $u(s_1, s_2)$
  - player 2 gets $-u(s_1, s_2)$

- Implicit sets of pure strategies
  - $S_1 = [n]$ for the *row player*
  - $S_2 = [m]$ for the *column players*
- matrix $M \in \mathbb{R}^{n \times m}$
  - row player gets M_{s_1, s_2}
  - column player gets $-M_{s_1, s_2}$



Equilibria of 🪨 📄 ✂ 🦎 🖖

Work it out yourself!

Nash Equilibria for Two-Player Zero-Sum Games

Nash's Theorem (1950), Special Case

For any $M \in \mathbb{R}^{n \times m}$ there exist distributions $\mathcal{S}_1^*$ on $[n]$ and $\mathcal{S}_2^*$ on $[m]$ such that

$$\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[M_{s_1,s_2}].$$

Intuition

When the players play according to $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$, then no player can benefit by deviating from his strategy.

Corollary: Loomis (1946) Von Neumann (1928)

For any $M \in \mathbb{R}^{n \times m}$ we have

$$\max_{\mathcal{S}_1} \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[M_{s_1,s_2}] = \min_{\mathcal{S}_2} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}]$$

Intuition

No first-mover disadvantage if

- first player choses mixed strategy
- second player answers with pure strategy

Proof of Corollary ("$\geq$")

$$\max_{\mathcal{S}_1} \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[M_{s_1,s_2}] \geq \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[M_{s_1,s_2}] \overset{\text{Nash}}{=} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] \geq \min_{\mathcal{S}_2} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}]$$

Nash Equilibria for Two-Player Zero-Sum Games

Nash's Theorem (1950), Special Case

For any $M \in \mathbb{R}^{n \times m}$ there exist distributions $\mathcal{S}_1^*$ on $[n]$ and $\mathcal{S}_2^*$ on $[m]$ such that

$$\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[M_{s_1,s_2}].$$

Intuition

When the players play according to $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$, then no player can benefit by deviating from his strategy.

Corollary: Loomis (1946) Von Neumann (1928)

For any $M \in \mathbb{R}^{n \times m}$ we have

$$\max_{\mathcal{S}_1} \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[M_{s_1,s_2}] = \min_{\mathcal{S}_2} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}]$$

Intuition

No first-mover disadvantage if

- first player choses mixed strategy
- second player answers with pure strategy

Proof of Corollary ("$\leq$")

$$\max_{\mathcal{S}_1} \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[M_{s_1,s_2}] = \max_{\mathcal{S}_1} \min_{\mathcal{S}_2} \mathbb{E}_{s_1 \sim \mathcal{S}_1, s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}] \leq \min_{\mathcal{S}_2} \max_{\mathcal{S}_1} \mathbb{E}_{s_1 \sim \mathcal{S}_1, s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}] = \min_{\mathcal{S}_2} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}]$$

Content

1. Nash Equilibria in 2-Player Zero-Sum Games
   - Games and Nash Equilibria
   - Two Player Zero Sum Games
   - Loomis' Theorem for Two-Player Zero Sum Games

2. Yao's Minimax Principle

3. Applications of Yao's Principle
   - Evaluation of $\overline{\wedge}$-Trees
     - Proof Sketch of Tarsi's Theorem (nicht prüfungsrelevant)
   - The Ski-Rental Problem

4. Conclusion

Algorithm Design as a 2-Player Zero-Sum Game

Setting

- P: a computational problem
- **Inputs**: *finite* set of inputs
- **Algos**: *finite* set of deterministic algorithms
- $C(A, I) \in \mathbb{R}$ cost of $A \in$ **Algos** on $I \in$ **Inputs**.

A Two-Player Zero-Sum Game

- Designer chooses (randomised) algorithm, i.e. a distribution on **Algos**.
  $\hookrightarrow$ Goal: Minimise (expected) cost.
- Adversary chooses (randomised) input, i.e. a distribution on **Inputs**.
  $\hookrightarrow$ Goal: Maximise (expected) cost.

Example: Sorting

- $P =$ "sort n numbers comparison-based"[a]
- **Inputs** $= S_n$ //permutations of $[n]$
- **Algos** $=$ e.g. suitable set of decision trees
- $C(A, I) =$ # of comparisons of A for input I

[a] n finite, though possibly $n \to \infty$ later.

Sorting (x, y, z)

		Adversary			
		$(1,2,3)$	$(3,1,2)$	$(2,3,1)$	$\dots$
Algorithm Designer	$x < y$ then $y < z$ then* $z < x$	2	3	3	
	$y < z$ then $z < x$ then* $x < y$	3	2	3	
	$\dots$				

* Only if needed.

Randomised Complexity and Yao's Principle

Definition: Randomised Complexity of a Problem

$$\mathcal{C} := \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] \qquad \text{designer moves first}$$

$$\overset{\text{Loomis}}{=} \max_{\mathcal{I} \text{ dist. on } \textbf{Inputs}} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}}[C(A, I)] \qquad \text{adversary moves first}$$

Yao's Principle: (Upper and) Lower Bounds on $\mathcal{C}$

Let $\mathcal{A}_0$ be a distribution on **Algos** and $\mathcal{I}_0$ a distribution on **Inputs**. Then

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)] \overset{\text{(old news)}}{\geq} \mathcal{C} \overset{\text{"Yao's Principle"}}{\geq} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)].$$

Tightness: Loomis implies that "$=$" is possible.

$\hookrightarrow$ Can attain (tight) lower bounds on $\mathcal{C}$ by thinking about deterministic algorithm only!

Content

1. Nash Equilibria in 2-Player Zero-Sum Games
   - Games and Nash Equilibria
   - Two Player Zero Sum Games
   - Loomis' Theorem for Two-Player Zero Sum Games

2. Yao's Minimax Principle

3. Applications of Yao's Principle
   - Evaluation of $\overline{\wedge}$-Trees
     - Proof Sketch of Tarsi's Theorem (nicht prüfungsrelevant)
   - The Ski-Rental Problem

4. Conclusion

Computational Problem: $\overline{\wedge}$-Tree-Evaluation

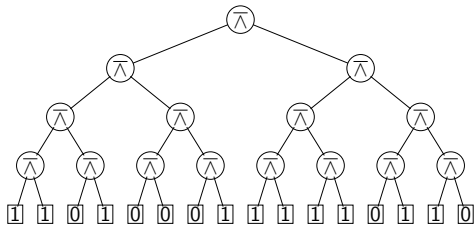
Problem: Evaluate $\overline{\wedge}$-Tree of depth d

- **Inputs** $= \{0,1\}^n$ for $n = 2^d$. Specify bits at leafs.
- **Algos** $=$ Algorithms computing value at root.
- $C(A, I) = \#$ bits of I that A examines
  $\hookrightarrow$ query complexity of A on I

Goal

Bound randomised query complexity

$$\mathcal{C} = \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$



Computational Problem: $\overline{\wedge}$-Tree-Evaluation

Problem: Evaluate $\overline{\wedge}$-Tree of depth d

- **Inputs** $= \{0,1\}^n$ for $n = 2^d$. Specify bits at leafs.
- **Algos** $=$ Algorithms computing value at root.
- $C(A, I) = \#$ bits of I that A examines
  $\hookrightarrow$ query complexity of A on I

Goal

Bound randomised query complexity

$$\mathcal{C} = \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$

Example and possible formalisation of **Algos** (that we won't use)

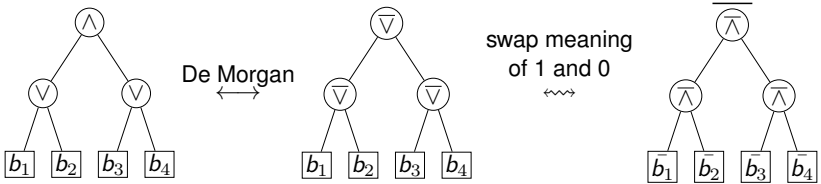
Each $A \in$ **Algos** corresponds to a *decision tree*. In the example:

- $C(A, (1,0,1,0)) = 4$
- $C(A, (0,1,0,1)) = 2$

Each leaf queried at most once per path
$\Rightarrow$ depth $\leq n \Rightarrow |\textbf{Algos}| < \infty$

What we already know

$\wedge$-$\vee$-trees are $\overline{\vee}$-trees are $\overline{\wedge}$-trees



What we already know

$\wedge$-$\vee$-trees are $\overline{\vee}$-trees are $\overline{\wedge}$-trees

Deterministic Query Complexity is n (Übungsblatt 2, Aufgabe 3)

For all $A \in$ **Algos** there exists $I \in$ **Inputs** such that $C(A, I) = n$.

Randomised Query Complexity is $\mathcal{O}(n^{\log_4(3)}) \approx \mathcal{O}(n^{0.792})$ (Lecture "The Power of Randomness")

Let $\mathcal{A}$ be the randomised algorithm that evaluates one of the two depth $d-1$ subtrees at random (recursively) and, if that yields 1, also evaluates the other subtree (recursively).

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \mathcal{O}(3^{d/2}) = \mathcal{O}(n^{\log_4(3)}).$$

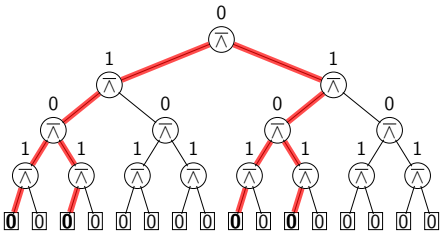
Goal: Show lower bound of $\Omega(\varphi^d) \approx \Omega(n^{0.694})$ using Yao's Principle (φ is the golden ratio).

Remark: actual complexity is $\Theta(n^{\log_4(3)})$, but that's more difficult.

Warm Up: A simple lower bound

Observation

For any even $d \in \mathbb{N}$ and $A \in$ **Algos** we have $C(A, (0, \ldots, 0)) \geq 2^{d/2}$.



Proof

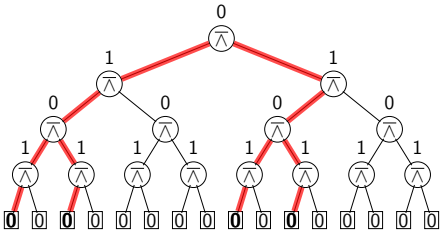
- in the end A knows that the root is 0.
- knowing a 0 requires knowing that both children are 1.
- Knowing a 1 requires knowing of one child that it is 0.

$\hookrightarrow$ A knows of $\geq 2^{d/2}$ leafs that they are 0 and must have checked them.

Warm Up: A simple lower bound

Observation

For any even $d \in \mathbb{N}$ and $A \in$ **Algos** we have $C(A, (0, \ldots, 0)) \geq 2^{d/2}$.

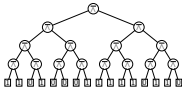


Corollary: Randomised Complexity is $\Omega(\sqrt{n})$

$$
\begin{aligned}
\mathcal{C} &= \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] \\
&\geq \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, (0, \ldots, 0))] \\
&= \min_{A \in \textbf{Algos}} [C(A, (0, \ldots, 0))] \\
&\geq 2^{d/2} = 2^{\log_2(n)/2} = n^{1/2}.
\end{aligned}
$$

Note Yao's spirit: Lower bound on *randomised complexity* from result on *deterministic algorithms*.

A stronger lower bound



Karlsruhe Institute of Technology

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

Lemma

Let $\varphi = \frac{\sqrt{5}+1}{2}$ be the golden ratio and $p_0 = \varphi - 1$. Then

$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1 + p_0)^d = \varphi^d.$$

Corollary: $\mathcal{C} = \Omega(\varphi^d) \approx \Omega(n^{0.694})$

$$\mathcal{C} \overset{\text{Yao}}{\geq} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] \overset{\text{Tarsi}}{=} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)]$$

$$\overset{\text{Lemma}}{=} \varphi^d = \varphi^{\log_2 n} = n^{\log_2 \varphi} \approx n^{0.694}.$$

Independent Bernoulli Inputs

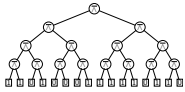
Let $\mathcal{I}_p = \text{Ber}(p)^n$ be the distribution where leafs are assigned independently values with distribution $\text{Ber}(p)$.

Deterministic Algorithm

Algorithm simpleEval(T)**:**

- **if** $T = \text{leaf}(b)$ **then**
  - **return** b
- **else**
  - $(T_\ell, T_r) \leftarrow T$
  - **if** simpleEval(T_ℓ) $= 0$ **then**
    - **return** 1
  - **else**
    - **return** $\neg$simpleEval(T_r)

Proof of Lemma: Cost of simpleEval on $\mathcal{I}_{p_0}$



Karlsruhe Institute of Technology

Lemma

Let $\varphi = \frac{\sqrt{5}+1}{2}$ be the golden ratio and $p_0 = \varphi - 1$. Then

$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1+p_0)^d = \varphi^d.$$

Proof.

- $p_0 = \frac{\sqrt{5}-1}{2}$ is the solution to $p = 1 - p^2$.
- If $a, b \sim \text{Ber}(p_0)$ then $a \overline{\wedge} b \sim \text{Ber}(1-p_0^2) = \text{Ber}(p_0)$.
- For $I \sim \mathcal{I}_{p_0}$ the probability that an *internal* tree node evaluates to 1 is p_0.
- Let $c_d := \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)]$ for trees of depth d. Then
  - $c_0 = 1$ // tree of depth 0 is just the leaf
  - $c_d = c_{d-1} + p_0 \cdot c_{d-1} = (1+p_0)c_{d-1} \overset{\text{Ind.}}{=} (1+p_0)(1+p_0)^{d-1} = (1+p_0)^d$
    // Always one recursive call, with probability p a second one.

□

Deterministic Algorithm

Algorithm simpleEval(T):

```
if T = leaf(b) then
    return b
else
    (T_ℓ, T_r) ← T
    if simpleEval(T_ℓ) = 0 then
        return 1
    else
        return ¬simpleEval(T_r)
```

Content

1. Nash Equilibria in 2-Player Zero-Sum Games
   - Games and Nash Equilibria
   - Two Player Zero Sum Games
   - Loomis' Theorem for Two-Player Zero Sum Games

2. Yao's Minimax Principle

3. Applications of Yao's Principle
   - Evaluation of $\overline{\wedge}$-Trees
     - Proof Sketch of Tarsi's Theorem (nicht prüfungsrelevant)
   - The Ski-Rental Problem

4. Conclusion

Tarsi's Theorem

Theorem (Tarsi 1984)

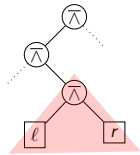
For any $p \in [0, 1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

Proof idea:

- Take optimal Algorithm A.
- Transform A into simpleEval step by step.
- Show: Expected query complexity never increases.

Lemma: Evaluating Superleafs like simpleEval



Definition: Superleafs

A *superleaf* consists of two sibling leafs and their parent.

Lemma

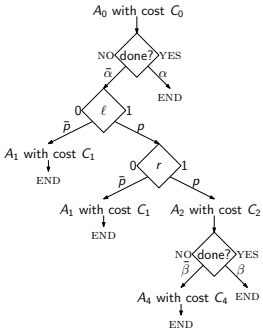
For any $p \in [0,1]$ and any $A \in$ **Algos** there exists $A' \in$ **Algos** such that

- $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$
- A' behaves on any superleaf $T = (\ell, r)$ like simpleEval:
  - i never visits r before ℓ
  - ii never visits r if $\ell = 0$
  - iii immediately visits r after visiting ℓ if $\ell = 1$

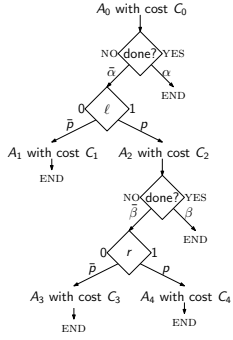
Proof Idea

- We fix every superleaf one by one. Let T be superleaf that needs fixing.
- Property i: Switch roles of ℓ and r if needed. Does not change the expected cost.
- Property ii: r does not contribute to result. Not visiting r *reduces* expected cost.
- Property iii: More difficult. See next slide.

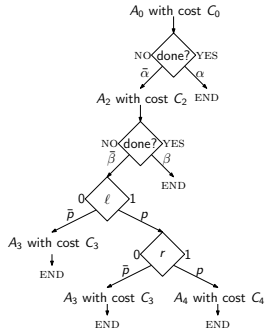
Modified Algorithm B



Original Algorithm A



Modified Algorithm D



$$C_A := \mathbb{E}[C(A,I)] = \mathbb{E}[C_0 + \bar{\alpha} \cdot (1 + \bar{p}C_1 + p \cdot (C_2 + \bar{\beta}(1 + \bar{p}C_3 + pC_4)))]$$
$$C_B := \mathbb{E}[C(B,I)] = \mathbb{E}[C_0 + \bar{\alpha} \cdot (1 + \bar{p}C_1 + p \cdot (1 + \bar{p}C_1 + p(C_2 + \bar{\beta}C_4)))]$$
$$C_D := \mathbb{E}[C(D,I)] = \mathbb{E}[C_0 + \bar{\alpha} \cdot (C_2 + \bar{\beta}(1 + \bar{p}C_3 + p(1 + \bar{p}C_3 + pC_4)))]$$

$$(C_B - C_A) + p \cdot (C_D - C_A) = \ldots = 0$$
$$\Rightarrow C_B - C_A \leq 0 \vee C_D - C_A \leq 0$$
$\Rightarrow B$ or D (or both) are at least as good as A and both visit superleaf (ℓ, r) as desired.

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in$ **Algos** be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.
By Lemma: There exists $A' \in$ **Algos** that behaves like simpleEval on superleafs such that

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Let L' be the number of superleafs visited by A' and L the number of superleafs visited by simpleEval.
Superleafs evaluate to 1 with probability $1 - p^2$ independently and are in a complete binary tree of depth $d - 1$.

Apply induction for $d' = d - 1$ and $p' = 1 - p^2$.

$$\mathbb{E}_{I \sim \mathcal{I}_p}[L] \overset{\text{Ind.}}{\leq} \mathbb{E}_{I \sim \mathcal{I}_p}[L'].$$

The expected cost for evaluating a superleaf is $1 + p$.
Hence

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] = (1 + p)\mathbb{E}[L']$$
$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)] = (1 + p)\mathbb{E}[L]$$

Finally we obtain:

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(\text{simpleEval}, I)] = (1 + p)\mathbb{E}[L] \leq (1 + p)\mathbb{E}[L']$$
$$= \mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Hence, simpleEval is optimal for $\mathcal{I}_p$. □

Content

1. Nash Equilibria in 2-Player Zero-Sum Games
   - Games and Nash Equilibria
   - Two Player Zero Sum Games
   - Loomis' Theorem for Two-Player Zero Sum Games

2. Yao's Minimax Principle

3. Applications of Yao's Principle
   - Evaluation of $\overline{\wedge}$-Trees
     - Proof Sketch of Tarsi's Theorem (nicht prüfungsrelevant)
   - The Ski-Rental Problem

4. Conclusion

Ski Rental – A Prototypical Online Problem

Setting: You are on a ski trip

Trip lasts for unknown number of days $I \in \mathbb{N}$ ("as long as there is snow").

Every day, if no skis bought yet:

- RENT skis for one day for cost 1 *or*
- BUY skis for cost $B \in \mathbb{N}$.

Goal: Minimise Competitive Ratio

The *competitive ratio* of distribution $\mathcal{A}$ on **Algos** is

$$C_{\mathcal{A}} = \sup_{I \in \textbf{Inputs}} \frac{\mathbb{E}_{A \sim \mathcal{A}}[C(A, I)]}{\mathrm{OPT}(I)}.$$

Framing using Online Algorithms

- **Inputs** $= \mathbb{N}$: number of days (not known in advance)
- **Algos** $= \mathbb{N}$: specify day for choosing BUY
- cost for $A \in$ **Algos** on $I \in$ **Inputs**:

$$C(A, I) = \begin{cases} I & \text{if } I < A \\ A - 1 + B & \text{otherwise.} \end{cases}$$

- cost of optimum *offline* solution

$$\mathrm{OPT}(I) = \begin{cases} I & \text{if } I < B \\ B & \text{otherwise.} \end{cases}$$

Break-Even is the best deterministic algorithm

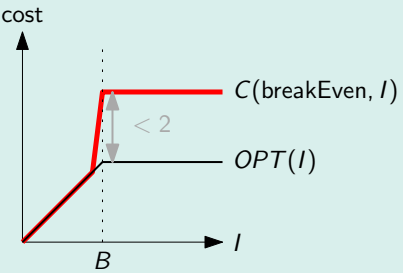
Observation

The algorithm breakEven $:= B$ has competitive ratio $\frac{2B-1}{B} \approx 2$.
All other $A \in$ **Algos** have competitive ratio ≥ 2.

Recall

B is the cost to BUY.

Proof



The worst ratio for breakEven is attained for input $I = B$.

$$C_{\text{breakEven}} = \sup_{I \in \mathbb{N}} \frac{C(\text{breakEven}, I)}{\text{OPT}(I)} = \frac{C(\text{breakEven}, B)}{\text{OPT}(B)}$$
$$= \frac{B - 1 + B}{B} = \frac{2B - 1}{B}.$$

Break-Even is the best deterministic algorithm

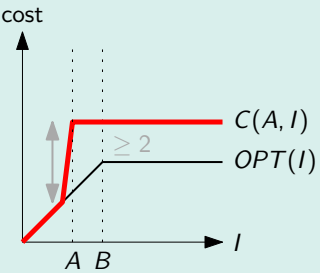
Observation

The algorithm breakEven := B has competitive ratio $\frac{2B-1}{B} \approx 2$.
All other $A \in$ **Algos** have competitive ratio ≥ 2.

Recall

B is the cost to BUY.

Proof



The worst ratio for $A \in$ **Algos** with $A < B$ is attained for input $I = A$.

$$C_A = \sup_{I \in \mathbb{N}} \frac{C(A,I)}{\mathrm{OPT}(I)} = \frac{C(A,A)}{\mathrm{OPT}(A)} = \frac{A-1+B}{A} = 1 + \frac{B-1}{A} \geq 1+1 = 2.$$

Break-Even is the best deterministic algorithm

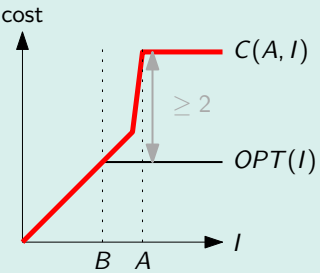
Observation

The algorithm breakEven $:= B$ has competitive ratio $\frac{2B-1}{B} \approx 2$.
All other $A \in$ **Algos** have competitive ratio ≥ 2.

Recall

B is the cost to BUY.

Proof



The worst ratio for $A \in$ **Algos** with $A > B$ is attained for input $I = A$.

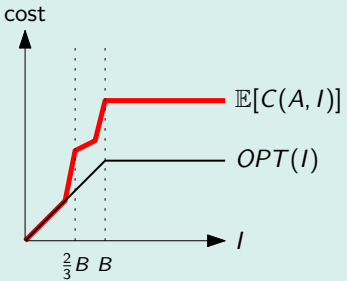
$$C_A = \sup_{I \in \mathbb{N}} \frac{C(A,I)}{\mathrm{OPT}(I)} = \frac{C(A,A)}{\mathrm{OPT}(A)} = \frac{A-1+B}{B} = 1 + \frac{A-1}{B} \geq 1+1 = 2.$$

A randomised algorithm can beat break-even

Observation (assuming wlog that B is a multiple of 3)

The randomised algorithm $\mathcal{A} = \mathcal{U}(\{\frac{2}{3}B, B\})$ has competitive ratio $\approx 1 + \frac{5}{6}$.

Proof



The competitive ratio of $\mathcal{A}$ "spikes" for inputs $\frac{2}{3}B$ and B. It is decreasing in between and constant after B.

$$\mathbb{E}_{A\sim\mathcal{A}}[C(A, \tfrac{2}{3}B)] = \underbrace{\tfrac{2}{3}B - 1}_{\text{rent}} + \underbrace{\tfrac{1}{2}(1+B)}_{\text{rent or buy}} < \tfrac{7}{6}B, \qquad \mathrm{OPT}(\tfrac{2}{3}B) = \tfrac{2}{3}B,$$

$$\mathbb{E}_{A\sim\mathcal{A}}[C(A, B)] = \underbrace{B}_{\text{buy}} + \underbrace{\tfrac{2}{3}B - 1}_{\text{rent}} + \underbrace{\tfrac{1}{2}(\tfrac{1}{3}B)}_{\text{maybe rent}} < \tfrac{11}{6}B, \qquad \mathrm{OPT}(B) = B.$$

Hence $C_{\mathcal{A}} = \sup_{I\in\mathbb{N}} \dfrac{\mathbb{E}_{A\sim\mathcal{A}}[C(A, I)]}{\mathrm{OPT}(I)} \leq \max\left\{\dfrac{7/6}{2/3}, \dfrac{11/6}{1}\right\} = \max\left\{\dfrac{7}{4}, \dfrac{11}{6}\right\} = \dfrac{11}{6}.$

What's next?

Goal: Lower bound

No randomised algorithm has competitive ratio better than $\frac{e-1}{e} \approx 1.582$.

Yao's Principle for Online Algorithms

Theorem (see Online Optimization Lecture, Corollary 3.8, Prof. Yann Disser, Darmstadt, 2023)

For any distribution $\mathcal{A}_0$ on **Algos** and any distribution $\mathcal{I}_0$ on **Inputs** we have

$$C_{\mathcal{A}_0} \stackrel{\text{def}}{=} \sup_{I \in \textbf{Inputs}} \frac{\mathbb{E}_{A\sim\mathcal{A}_0}[C(A,I)]}{\mathrm{OPT}(I)} \geq \frac{\mathbb{E}_{I\sim\mathcal{I}_0, A\sim\mathcal{A}_0}[C(A,I)]}{\mathbb{E}_{I\sim\mathcal{I}_0}[\mathrm{OPT}(I)]} \geq \frac{\inf_{A\in\textbf{Algos}} \mathbb{E}_{I\sim\mathcal{I}_0}[C(A,I)]}{\mathbb{E}_{I\sim\mathcal{I}_0}[\mathrm{OPT}(I)]}.$$

Remark

- Yao's principle exists for other settings as well.
- Proof of "$\geq$" relatively easy to prove.
- Tightness typically follows from duality of optimisation problems or fixed point theorems.
  (though I'm not sure how it works here)

A hard distribution for Ski-Rental: Intuition

$$\mathcal{I}_0 := \mathrm{Geom}_1(\tfrac{1}{B}).$$

Why $\mathcal{I}_0$?

- distribution is **memoryless**, i.e. $\Pr_{I\sim\mathcal{I}_0}[I = i + t \mid I > i] = \Pr_{I\sim\mathcal{I}_0}[I = t]$.
  Assume no skis bought on day i: Minimising expected *future* cost is the same problem as on day 1.
  $\hookrightarrow$ wlog: either buy right away or not at all.
- **expectation** tuned such that

$$\mathbb{E}_{I\sim\mathcal{I}_0}[C(\text{never buy}, I)] = \mathbb{E}_{I\sim\mathcal{I}_0}[C(\text{immediately buy}, I)] = B.$$

  $\hookrightarrow$ all strategies equally good

A hard distribution for Ski-Rental: Analysis

Lemma

Let $\mathcal{I}_0 := \text{Geom}(\frac{1}{B})$ *and* $q := 1 - \frac{1}{B} =$ "$\Pr[❄]$". *Then*

i. $\mathbb{E}_{I\sim\mathcal{I}_0}[C(A, I)] = B$ *for all* $A \in \mathbb{N}$.

ii. $\mathbb{E}_{I\sim\mathcal{I}_0}[\text{OPT}(I)] = B(1 - (1 - \frac{1}{B})^B)$.

Tail sum formula:

For random variable X with values in $\mathbb{N}$:

$$\mathbb{E}[X] = \sum_{j\geq 1} \Pr[X \geq j].$$

Proof of (i)

$$\begin{aligned}\mathbb{E}_{I\sim\mathcal{I}_0}[C(A, I)] &= \underbrace{\Pr[I \geq A] \cdot B}_{\text{buy}} + \sum_{i=1}^{A-1} \underbrace{\Pr[I \geq i] \cdot 1}_{\text{rent}} \\ &= q^{A-1} \cdot B + \sum_{i=1}^{A-1} q^{i-1} = q^{A-1} \cdot B + \sum_{i=0}^{A-2} q^i = q^{A-1} \cdot B + \frac{1 - q^{A-1}}{1 - q} \\ &= q^{A-1} \cdot B + (1 - q^{A-1})B = B.\end{aligned}$$

A hard distribution for Ski-Rental: Analysis

Lemma

Let $\mathcal{I}_0 := \text{Geom}(\frac{1}{B})$ *and* $q := 1 - \frac{1}{B} =$ "$\Pr[\text{❄}]$". *Then*

i. $\mathbb{E}_{I\sim\mathcal{I}_0}[C(A, I)] = B$ *for all* $A \in \mathbb{N}$.

ii. $\mathbb{E}_{I\sim\mathcal{I}_0}[\text{OPT}(I)] = B(1 - (1 - \frac{1}{B})^B)$.

Tail sum formula:

For random variable X with values in $\mathbb{N}$:

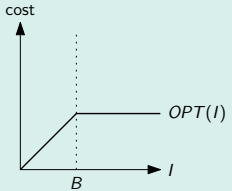
$$\mathbb{E}[X] = \sum_{j\geq 1} \Pr[X \geq j].$$

Proof of (ii)

$$\mathbb{E}_{I\sim\mathcal{I}_0}[\text{OPT}(I)] \overset{\text{TSF}}{=} \sum_{j\geq 1} \Pr[\text{OPT}(I) \geq j] = \sum_{j=1}^{B} \Pr[\text{OPT}(I) \geq j]$$

$$= \sum_{j=1}^{B} \Pr[I \geq j] = \sum_{j=1}^{B} q^{j-1} = \sum_{j=0}^{B-1} q^j$$

$$= \frac{1 - q^B}{1 - q} = B(1 - (1 - \tfrac{1}{B})^B).$$



Note: $\text{OPT}(I) = I$ for $I \in [B]$.

A hard distribution for Ski-Rental: Analysis

Lemma

Let $\mathcal{I}_0 := \text{Geom}(\frac{1}{B})$ *and* $q := 1 - \frac{1}{B} =$ "$\Pr[$ ❄ $]$"*. Then*

i. $\mathbb{E}_{I\sim\mathcal{I}_0}[C(A, I)] = B$ *for all* $A \in \mathbb{N}$.

ii. $\mathbb{E}_{I\sim\mathcal{I}_0}[\text{OPT}(I)] = B(1 - (1 - \frac{1}{B})^B)$.

Tail sum formula:

For random variable X with values in $\mathbb{N}$:

$$\mathbb{E}[X] = \sum_{j\geq 1} \Pr[X \geq j].$$

Lower bound for Ski-Rental

By Yao's theorem any randomised algorithm $\mathcal{A}$ for ski-rental has competitive ratio at least

$$C_{\mathcal{A}} \overset{\text{Yao}}{\geq} \frac{\inf_{A\in\textbf{Algos}} \mathbb{E}_{I\sim\mathcal{I}_0}[C(A, I)]}{\mathbb{E}_{I\sim\mathcal{I}_0}[\text{OPT}(I)]} = \frac{B}{B(1 - (1 - \frac{1}{B})^B)} = \frac{1}{1 - (1 - \frac{1}{B})^B}.$$

For large B the lower bound converges to $\lim_{B\to\infty} \frac{1}{1 - (1 - \frac{1}{B})^B} = \frac{1}{1 - 1/e} = \frac{e}{e - 1} \approx 1.582.$

Upper bound for Ski-Rental

Remark: The lower bound is tight (Karlin et al. 1994)

There exists a distribution $\mathcal{A}$ on $[B]$ such that $c_{\mathcal{A}} \leq \frac{e}{e-1}$.

Applications

Very basic online question:

Should I pay a small possibly recurring cost or a large one time cost?

Occurs in:

- Cache management.
- Networking.
- Scheduling.
- ...

Conclusion

Algorithm Design as a Two-Player Game

- “we” choose algorithm to minimise cost
- “adversary” chooses input to maximise cost
- Nash/Loomis: It does not matter who moves first *if* mixed strategy is allowed for first player.

Yao’s Principle

Lower bound on worst-case expected cost of *any randomised algorithm* $\mathcal{A}_0$ by analying *any deterministic algorithm* on specific input distribution $\mathcal{I}_0$.

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)] \geq \mathcal{C} \geq \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)].$$

Can narrow down randomised complexity $\mathcal{C}$ of underlying problem from both sides.

Anhang: Mögliche Prüfungsfragen I

Spieltheorie:

- Was ist ein Zwei-Spieler-Spiel im Sinne der Spieltheorie?
- Was ist ein Nash-Equilibrium?
- Gibt es immer ein Nash-Equilibrium?
- Was ist ein Nullsummenspiel?
- Was besagt der Satz von Nash (für Zwei-Spieler Nullsummenspiele)?
- Was besagt der Satz von Loomis?
- Beweise den Satz von Loomis! (anspruchsvolle Aufgabe)

Yaos Prinzip:

- Worin besteht die Verbindung zwischen Spieltheorie und dem Entwurf von Algorithmen?
- Wie ist die randomisierte Komplexität (bzgl. einer Kostenfunktion C) normalerweise definiert? Welche andere Sichtweise ergibt sich darauf durch den Satz von Loomis?
- Formuliere Yaos Prinzip! Wofür ist es nützlich?

Anhang: Mögliche Prüfungsfragen II

Anwendung auf $\overline{\wedge}$-Bäume:

- Welches Ziel haben wir uns bei der Auswertung von $\overline{\wedge}$-Bäumen gesetzt? (Anfragekomplexität minimieren)
- Welche Worst-Case Kosten lassen sich mit einem deterministischen Algorithmus erreichen?
- Können randomisierte Algorithmen das besser? Wie?
- Man kann sich recht leicht überlegen, dass die randomisierte Komplexität $\Omega(\sqrt{n})$ beträgt. Wie ging das?
- Wir haben auch eine schärfere Analyse gesehen. Welche Komponenten hatte diese? Insbesondere: Wie kommt dabei Yao Prinzip zur Anwendung?
- Was besagt der Satz von Tarsi?

Ski-Rental-Problem:

- Formuliere das Ski-Rental Problem.
- Wie nennt man diese Art von Problem? (*Online* Problem)
- Spielt das nur im Wintersport eine Rolle? (nur Stichworte)
- Wie ist der kompetitive Faktor definiert?

Anhang: Mögliche Prüfungsfragen III

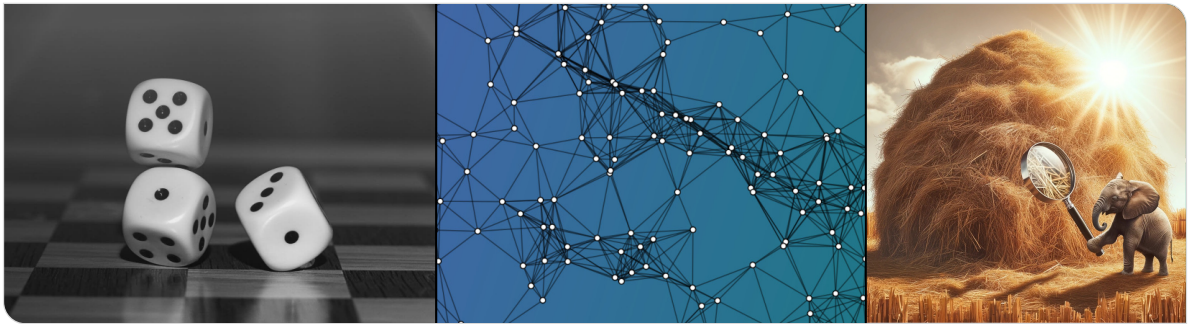
- Was ist der beste deterministische Algorithmus? Wie kann man das einsehen?
- Gibt es einen randomisierten Algorithmus der Break-Even schlagen kann? (nur die Idee)
- Formuliere Yaos Prinzip für Online Algorithmen.
- Welche Eingabeverteilung haben wir für die untere Schranke für Ski-Rental zugrunde gelegt? Was ist die Intuition?
- Welche Kosten ergeben sich für Online und Offline Algorithmen für diese Eingabeverteilung? Was lässt sich entsprechend über den kompetitiven Faktor sagen?

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Probabilistic Method

Stefan Walzer | WS 2024/2025



The Name of the Game

The Probabilistic Method (pioneered by Paul Erdős)

Show that something exists by proving that it has a positive probability of arising from a random process.

- Used to proved statements that don't involve randomness at all.
- Probabilistic arguments replace combinatorial arguments.

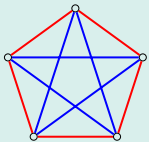
First Example: Ramsey Numbers

Definition: Ramsey Number

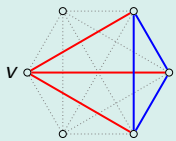
$R(k,k) := \min\{n \in \mathbb{N} \mid$ any red-blue colouring of the edges of K_n contains a monochromatic k-clique$\}$. [a]

[a] The general definition of $R(r,b)$ asks for red r-clique or blue b-clique.

$R(3,3) > 5$



$R(3,3) \leq 6$



- v has 3 incident edges of the same colour
- wlog that colour is red
- if there is no red triangle then w_1, w_2, w_3 form a blue triangle.

Hence: $R(3,3) = 6$.

First Example: Ramsey Numbers

Definition: Ramsey Number

$R(k,k) := \min\{n \in \mathbb{N} \mid$ any red-blue colouring of the edges of K_n contains a monochromatic k-clique$\}$.

Theorem: $R(k,k) > 2^{k/2}$ for $k \geq 6$. // actually $k \geq 3$ suffices

Proof.

- *To show*: Edges of K_n with $n \leq 2^{k/2}$ can be coloured while avoiding a monochromatic k-clique.
- *Plan*: Show that *uniformly random colouring* avoids monochromatic k-clique with positive probability.
- There are $\binom{n}{k}$ k-cliques. Each is monochromatic with probability $2^{-\binom{k}{2}+1}$.
- The number M of monochromatic k-cliques satisfies:

$$\mathbb{E}[M] = \binom{n}{k} \cdot 2^{-\binom{k}{2}+1} \leq \frac{n^k}{k!} \cdot 2^{-k^2/2+k/2+1} \leq \frac{(2^{k/2})^k}{(k/2)^{k/2}} \cdot 2^{-k^2/2} 2^{k/2} 2 = 2\left(\frac{4}{k}\right)^{k/2} < 1.$$

Since $\mathbb{E}[M] < 1$ it is possible that $M = 0$. In particular a colouring with no monochromatic k-cliques exists. □

Expectation Argument

We have implicitly used:

$\Pr[X \leq \mathbb{E}[X]] > 0$ and $\Pr[X \geq \mathbb{E}[X]] > 0$.

Probabilistic Method with Expectation Argument

Show that an object x with $f(x) \overset{\leq}{\geq} b$ exists by proving that a random object X satisfies $\mathbb{E}[f(X)] \overset{\leq}{\geq} b$.

Simple Use Case

Any graph $G = (V, E)$ admits a cut of weight at least $|E|/2$.

Proof.

- Assign each $v \in V$ to V_1 or V_2 uniformly at random.
- Each edge crosses the cut (V_1, V_2) with probability $1/2$.
- $\mathbb{E}[\text{weight of } (V_1, V_2)] = \mathbb{E}\Big[\sum_{e \in E}[e \text{ crosses } (V_1, V_2)]\Big] = \sum_{e \in E} \Pr[e \text{ crosses } (V_1, V_2)] = |E| \cdot \frac{1}{2}$. □

Example: Independent Sets

Theorem

Let $G = (V, E)$ with $n = |V|$, $m = |E|$ and $m \geq \frac{n}{2}$.
Then there exists an independent set of size $\frac{n^2}{4m}$. // $\Theta(\frac{n}{\text{average degree}})$

Proof.

- sampleAndReject computes an independent set $V^+ \setminus V^-$.
- $\mathbb{E}[|V^+|] = n \cdot \frac{n}{2m} = \frac{n^2}{2m}$.
- $\mathbb{E}[|V^-|] \leq \sum_{\{u,v\} \in E} \Pr[u \in V^+, v \in V^+] = \sum_{\{u,v\} \in E} \left(\frac{n}{2m}\right)^2 = \frac{n^2}{4m}$.
- $\mathbb{E}[|V^+ \setminus V^-|] = \mathbb{E}[|V^+|] - \mathbb{E}[|V^-|] \geq \frac{n^2}{2m} - \frac{n^2}{4m} = \frac{n^2}{4m}$. □

Algorithm sampleAndReject**:**

```
V⁺ ← ∅
for v ∈ V do
    with probability n/(2m) do
        V⁺ ← V⁺ ∪ {v}
V⁻ ← ∅
for {u, v} ∈ E do
    if u ∈ V⁺ and v ∈ V⁺ then
        with probability 1/2 do
            V⁻ ← V⁻ ∪ {u}
        otherwise
            V⁻ ← V⁻ ∪ {v}
return V⁺ \ V⁻
```

Remark: sampleAndReject seems suitable for a parallel / distributed setting.

Lovász Local Lemma

Context

Given: Family $\mathcal{E} = \{E_1, \ldots, E_n\}$ of "bad" events with $\Pr[E_i] \leq p < 1$.

Want: Show $\Pr[\bar{E}_1 \cap \ldots \cap \bar{E}_n] = \Pr[\text{none of } \mathcal{E}] > 0$.

Observation: Easy if $\mathcal{E}$ is independent

If $\mathcal{E}$ is an independent family then $\Pr[\text{none of } \mathcal{E}] = \prod_{i=1}^{n} \Pr[\bar{E}_i] \geq (1-p)^{|\mathcal{E}|} > 0$.

Observation: Expectation arguments only gets us so far

If $np < 1$ then $\mathbb{E}[\#\text{events from } \mathcal{E} \text{ occuring}] \leq np < 1$, hence $\Pr[\text{none of } \mathcal{E}] > 0$.
If $np = 1$ then $\Pr[\text{none of } \mathcal{E}] = 0$ is possible, e.g. $X \sim \mathcal{U}([n])$ and $E_i := \{X = i\}$.

Lovász Local Lemma (László Lovász and Paul Erdős, 1973)

If each $E \in \mathcal{E}$ has $\Pr[E] < p$ and depends on at most d events[a] from $\mathcal{E}$ and $4pd \leq 1$ then $\Pr[\text{none of } \mathcal{E}] > 0$.

[a] Little challenge: State what this means formally.

Example for Lovász Local Lemma (by Wikipedia User *Kevinatilusa*)

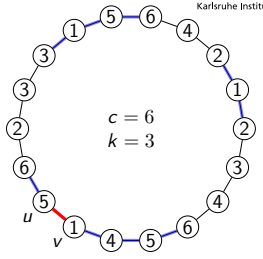
Lovász Local Lemma (László Lovász and Paul Erdős, 1973)

If each $E \in \mathcal{E}$ has $\Pr[E] < p$ and depends on at most d events from $\mathcal{E}$ and $4pd \leq 1$ then $\Pr[\text{none of } \mathcal{E}] > 0$.

Setting

Consider a necklace of ck beads with k beads of each of c colours.
An *independent rainbow* is a set of beads

- containing one bead of each colour // rainbow
- and not containing a pair of adjacent beads. // independent



Claim: If $k \geq 16$ then an independent rainbow always exists. // $k \geq 11$ also suffices

Consider any necklace. Let R contain a random bead of each color. // Goal: $\Pr[R \text{ independent}] > 0$.
One bad event per pair of adjacent beads:

$$E_{\{u,v\}} := \{u \in R \wedge v \in R\}, \quad \Pr[E] \leq \tfrac{1}{k^2} =: p.$$

$E_{\{u,v\}}$ depends on $E_{\{u',v'\}}$ only if u' or v' share the colour of u or v.
$2k$ relevant beads, hence $4k-2$ relevant pairs.

$$\Rightarrow d = 4k-2, \qquad 4pd \leq 4\tfrac{1}{k^2}(4k-2) < \tfrac{16}{k} \leq 1.$$

$$\Pr[R \text{ independent}] = \Pr[\text{none of } (E_{\{u,v\}})_{u,v}] \overset{\text{LLL}}{>} 0.$$

Proof of Lovász Local Lemma

Lovász Local Lemma (László Lovász and Paul Erdős, 1973)

If each $E \in \mathcal{E}$ has $\Pr[E] < p$ and depends on at most d events from $\mathcal{E}$ and $4pd \leq 1$ then $\Pr[\text{none of } \mathcal{E}] > 0$.

Claim: $\forall S \subseteq \mathcal{E} : \forall E^* \in \mathcal{E} \setminus S : \Pr[E^* \mid \text{none of } S] \leq 2p$.

Proof of LLL using the Claim.

$\Pr[\text{none of } \mathcal{E}] = \prod_{i=1}^{n} \Pr[\bar{E}_i \mid \text{none of } \{E_1, \ldots, E_{i-1}\}] \geq (1-2p)^n \overset{4pd \leq 1}{>} 2^{-n} > 0.$ □

Proof of Lovász Local Lemma

Lovász Local Lemma (László Lovász and Paul Erdős, 1973)

If each $E \in \mathcal{E}$ has $\Pr[E] < p$ and depends on at most d events from $\mathcal{E}$ and $4pd \leq 1$ then $\Pr[\text{none of } \mathcal{E}] > 0$.

Claim: $\forall S \subseteq \mathcal{E} : \forall E^* \in \mathcal{E} \setminus S : \Pr[E^* \mid \text{none of } S] \leq 2p$.

Proof of the Claim by Induction on $|S|$.

- Base case: If $|S| = 0$ then $\Pr[E^* \mid \text{none of } \varnothing] = \Pr[E^*] \leq p \leq 2p$. ✓ Let now $|S| > 0$.
- Partition $S = I \dot\cup D$ such that E^* is independent of I and $1 \leq |D| \leq d$. // $\leq d$ possible by assumption, > 0 is our choice.
- $\Pr[\text{none of } D \mid \text{none of } I] = 1 - \Pr[\bigcup_{E \in D} E \mid \text{none of } I] \overset{\text{UB}}{\geq} 1 - \sum_{E \in D} \underbrace{\Pr[E \mid \text{none of } I]}_{\leq 2p \text{ (Induction, using } |I| < |S|)} \geq 1 - 2dp \overset{4pd \leq 1}{\geq} \frac{1}{2}$. (☆).

$$\Pr[E^* \mid \text{none of } S] = \frac{\Pr[E^* \wedge \text{none of } S]}{\Pr[\text{none of } S]} \leq \frac{\Pr[E^* \wedge \text{none of } I]}{\Pr[\text{none of } D \mid \text{none of } I]\Pr[\text{none of } I]}$$

$$= \frac{\Pr[E^*]\Pr[\text{none of } I]}{\Pr[\text{none of } D \mid \text{none of } I]\Pr[\text{none of } I]} = \frac{p}{\Pr[\text{none of } D \mid \text{none of } I]} \overset{(☆)}{\leq} \frac{p}{1/2} = 2p. \qquad \square$$

Summary

What the Probabilistic Method is all About

- Goal: Prove the existence of objects with certain properties.
- Use probabilistic language as a tool.

Vanilla Variant:

Goal: Show that $P \subseteq \Omega$ is not empty.

1. Define a random object $X \in \Omega$.
2. Show: $\Pr[X \in P] > 0$.
3. Conclude: $\exists x \in \Omega : x \in P$.

Variant with Expectation Argument

Goal: Show that $f : \Omega \to \mathbb{R}$ has maximum at least q.

1. Define a random object $X \in \Omega$.
2. Show: $\mathbb{E}[f(X)] \geq q$.
3. Conclude: $\exists x \in \Omega : f(x) \geq q$.

Variant with Lovász Local Lemma

Goal: Show that $P \subseteq \Omega$ is not empty.

1. Define random object X.
2. Define family $\mathcal{E}$ of bad events such that $\bigcap_{E \in \mathcal{E}} \bar{E} \Rightarrow X \in P$.
4. Show that $E \in \mathcal{E}$ satisfies $\Pr[E] \leq p$.
5. Show $E \in \mathcal{E}$ depends on at most d other events from $\mathcal{E}$.
6. Show $4dp \leq 1$.
7. Conclude with LLL: $\exists x : x \in P$.

Anhang: Mögliche Prüfungsfragen I

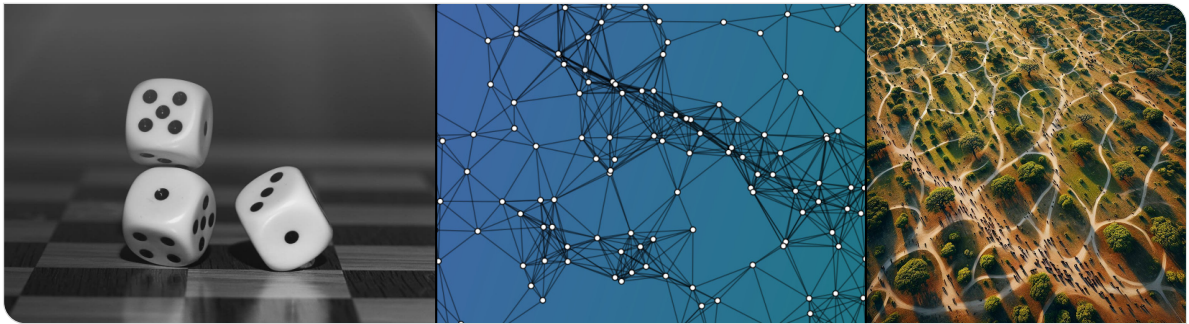
- Was ist das Ziel der probabilistischen Methode?
- Bezüglich der grundlegenden Methode:
  - Welche “Kreativleistung” muss man erbringen und was muss man dann ausrechnen?
  - Verdeutliche die Methode an einem Beispiel.
- Bezüglich der Variante mit Erwartungswertargument:
  - Welche “Kreativleistung” muss man erbringen und was muss man dann ausrechnen?
  - Verdeutliche die Methode an einem Beispiel.
  - Wir haben gezeigt, dass jeder Graph einen Schnitt von Gewicht $|E|/2$ besitzt. Wie?
  - Wir haben gezeigt, dass jeder Graph eine unabhängige Menge der Größe $\frac{n^2}{4m}$ besitzt. Wie?
- Bezüglich Lovász Local Lemma:
  - Formuliere die Aussage des Lemmas.
  - Was ist der Bezug zur probabilistischen Methode?
  - Wir haben gezeigt, dass gefärbte Graphen unabhängige Regenbogenmengen gewisser Größe besitzen. Wie sind wir vorgegangen?

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Random Graphs

Stefan Walzer | WS 2024/2025



Content

1. Motivation

2. Erdős-Renyi Random Graphs
   - Degree Distribution
   - Degree Statistics
   - Tree-like local structure
   - Emergence of the Giant Component

3. Random Geometric Graphs

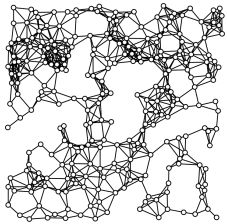
4. Scale-Free Networks (Teaser)

Motivation 1: Average Case Analysis

Theory-Practice Gap

Minimum Vertex Cover is APX-hard $\overset{???}{\longleftrightarrow}$ small vertex covers can often be computed efficiently in practice

$\rightsquigarrow$ relevant graph classes (e.g. social networks) are not worst-case.



Bridging the Gap

1. Define a distribution $\mathcal{G}$ on graphs.
   - $\mathcal{G}$ should be realistic, i.e. model real world instances
   - $\mathcal{G}$ should have simple mathematical structure
2. Consider randomised complexity of handling $G \sim \mathcal{G}$.

Goals

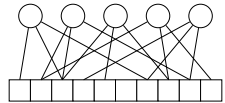
- model real world instances
- identify useful properties of these instances
- build algorithms exploiting these properties

Motivation 2: Data Structure Design

Stay tuned

Random graphs occur naturally in

- cuckoo hash tables
- retrieval data structures
- perfect hash functions



Content

1. Motivation

2. Erdős-Renyi Random Graphs
   - Degree Distribution
   - Degree Statistics
   - Tree-like local structure
   - Emergence of the Giant Component

3. Random Geometric Graphs

4. Scale-Free Networks (Teaser)

The Erdős-Renyi Model and Related Distributions

Original Erdős-Renyi Model $G(n, m)$: "Uniformly random graph with n nodes and m edges"

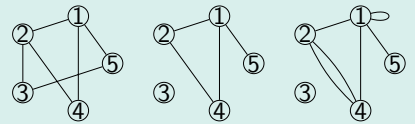
Gilbert Model $G(n, p)$: "Every edge with probability p"

Uniform Endpoint Model $G^{\text{UE}}(n, m)$: "randomly attach the $2m$ endpoints of edges"

Definition

Let $n \in \mathbb{N}$, $0 \leq m \leq \binom{n}{2}$. We use $G(n, m)$ to refer to a graph sampled uniformly from the set of all graphs with vertex set $[n]$ and m edges.

Example: $n = 5, m = 6$



probability $1/\binom{\binom{n}{2}}{m}$ 0 0

The Erdős-Renyi Model and Related Distributions

Original Erdős-Renyi Model $G(n, m)$: "Uniformly random graph with n nodes and m edges"

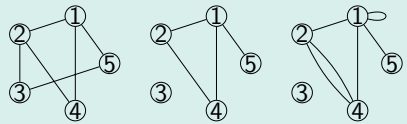
Gilbert Model $G(n, p)$: "Every edge with probability p"

Uniform Endpoint Model $G^{\text{UE}}(n, m)$: "randomly attach the $2m$ endpoints of edges"

Definition

Let $n \in \mathbb{N}$ and $p \in (0, 1)$. We use $G(n, p)$ to refer to a graph with vertex set $[n]$ that contains each of the $\binom{n}{2}$ possible edges with probability p, independently from other edges.

Example: $n = 5$



probability $p^6(1-p)^4$ $p^4(1-p)^6$ 0

The Erdős-Renyi Model and Related Distributions

Original Erdős-Renyi Model $G(n, m)$: "Uniformly random graph with n nodes and m edges"

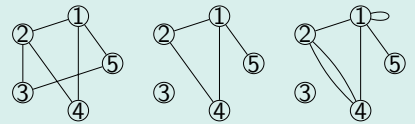
Gilbert Model $G(n, p)$: "Every edge with probability p"

Uniform Endpoint Model $G^{\text{UE}}(n, m)$: "randomly attach the $2m$ endpoints of edges"

Definition

Let $n, m \in \mathbb{N}$ and $v_1, \ldots, v_{2m} \sim \mathcal{U}([n])$. We use $G^{\text{UE}}(n, m)$ to refer to a multi-graph with vertex set $[n]$ and a multiset of edges that contains a copy of $\{v_{2i-1}, v_{2i}\}$ for each $i \in [m]$.

Example: $n = 5, m = 6$



probability $6! \cdot 2^6 \cdot 5^{-12}$ 0 $6! \cdot 2^4 \cdot 5^{-12}$

The Erdős-Renyi Model and Related Distributions

Original Erdős-Renyi Model $G(n, m)$: "Uniformly random graph with n nodes and m edges"

Gilbert Model $G(n, p)$: "Every edge with probability p"

Uniform Endpoint Model $G^{\text{UE}}(n, m)$: "randomly attach the $2m$ endpoints of edges"

Remarks

- for $p = m/\binom{n}{2}$ the three distributions are similar in many ways
- the original Erdős-Renyi model is often inconvenient to work with
- the uniform endpoint model is non-standard (we'll need it in later chapters)

Plan for the Next Few Slides: Sparse Graphs

Focus on Expected Degree $\lambda \in \mathcal{O}(1)$

- for $G(n, m)$ choose $m = \frac{\lambda n}{2}$ $\Rightarrow$ average vertex degree $\frac{2m}{n} = \lambda$
- for $G(n, p)$ choose $p = \frac{\lambda}{n-1}$ $\Rightarrow$ expected vertex degree $(n-1) \cdot p = \lambda$
- for $G^{\text{UE}}(n, m)$ choose $m = \frac{\lambda n}{2}$ $\Rightarrow$ average vertex degree $\frac{2m}{n} = \lambda$ // loops contribute 2 to a vertex degree

Goals

- Build intuition for properties of Erdős-Renyi graphs.
- Get a feeling for how to work with them.
- For simplicity: Focus on the Gilbert model only.

Selected Properties of Erdős-Renyi Graphs

On the next few slides we consider:

Vertex Degrees

For large n, the degree of a given vertex is approximately Poisson distributed.

Local Structure

The neighbourhood around a vertex resembles a Galton-Watson tree.

Degree Statistics

The number of vertices of each degree is highly concentrated around its expectation.

Largest Connected Component

Size of the largest component is highly predictable.

Content

1. Motivation

2. Erdős-Renyi Random Graphs
   - Degree Distribution
   - Degree Statistics
   - Tree-like local structure
   - Emergence of the Giant Component

3. Random Geometric Graphs

4. Scale-Free Networks (Teaser)

Degree Distribution of Erdős-Reyni Graphs

Exercise: Degrees are approximately Poisson distributed

For each $n \in \mathbb{N}$ consider $G(n, \lambda/n)$ and the degree $X_n \sim \text{Bin}(n-1, \lambda/n)$ of vertex 1. Moreover, let $X \sim \text{Pois}(\lambda)$. Then

$$X_n \xrightarrow{d} X \text{ for } n \to \infty.$$

The same holds for $G(n, \lfloor \lambda n/2 \rfloor)$ and $G^{\text{UE}}(n, \lambda n/2)$.

Content

1. Motivation

2. Erdős-Renyi Random Graphs
   - Degree Distribution
   - Degree Statistics
   - Tree-like local structure
   - Emergence of the Giant Component

3. Random Geometric Graphs

4. Scale-Free Networks (Teaser)

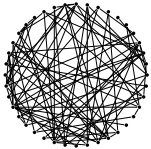
The Number N_d of Vertices of Degree N_d

Notation

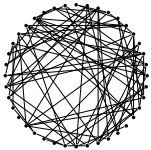
- Let $d \in \mathbb{N}$, $\lambda > 0$. We consider $G(n, \lambda/n)$. // Gilbert model
- Let $N_d := |\{v \in [n] \mid \deg(v) = d\}|$

Is N_d highly concentrated?

- Note: $(\deg(v))_{v \in [n]}$ are correlated.
- Otherwise N_d would have a binomial distribution and we could use Chernoff bounds.



d	0	1	2	3	4	5	6	7	8	9
N_d	0	2	8	6	7	7	3	2	3	1



d	0	1	2	3	4	5	6	7	8	9
N_d	1	2	5	8	9	11	3	1	0	0

Concentration of N_d

Lemma (Near Independence of Degrees)

Let $u \neq v$ be two vertices of $G(n, \lambda/n)$. Then $\Pr[\deg(u) = d, \deg(v) = d] = \Pr[\deg(u) = d] \Pr[\deg(v) = d] \pm \Theta(1/n)$.

Proof

Let $\deg'(u) = \deg(u) - [\{u, v\} \in E]$ be the degree of u when ignoring $\{u, v\}$ if present. Then

$$\Pr[\deg(u) \neq \deg'(u)] = \Pr[\{u, v\} \in E] = \lambda/n = \Theta(1/n).$$

The same holds for $\deg'(v) = \deg(v) - [\{u, v\} \in E]$. We conclude:

$$\begin{aligned}\Pr[\deg(v_1) = d, \deg(v_2) = d] &= \Pr[\deg'(v_1) = d, \deg'(v_2) = d] \pm \Theta(1/n) \\ &= \Pr[\deg'(v_1) = d] \Pr[\deg'(v_2) = d] \pm \Theta(1/n) = \Pr[\deg(v_1) = d] \Pr[\deg(v_2) = d] \pm \Theta(1/n).\end{aligned}$$

Concentration of N_d

Lemma (Near Independence of Degrees)

Let $u \neq v$ be two vertices of $G(n, \lambda/n)$. Then $\Pr[\deg(u) = d, \deg(v) = d] = \Pr[\deg(u) = d] \Pr[\deg(v) = d] \pm \Theta(1/n)$.

Theorem

$\Pr[|N_d - np_d| \geq n^{2/3}] = \mathcal{O}(n^{-1/3})$ where $p_d = \Pr[\deg(1) = d] \approx e^{-\lambda} \frac{\lambda^d}{d!}$.

Proof

$\mathbb{E}[N_d] = np_d$ $\qquad$ $\mathbb{E}[N_d^2] = n^2 p_d^2 \pm \mathcal{O}(n)$ $\qquad$ $\mathrm{Var}(N_d) = \mathcal{O}(n)$.

$$\mathbb{E}[N_d] = \mathbb{E}[\sum_{v \in [n]} [\deg(v) = d]] = n \cdot \Pr[\deg(1) = d] = n \cdot p_d.$$

Concentration of N_d

Lemma (Near Independence of Degrees)

Let $u \neq v$ be two vertices of $G(n, \lambda/n)$. Then $\Pr[\deg(u) = d, \deg(v) = d] = \Pr[\deg(u) = d] \Pr[\deg(v) = d] \pm \Theta(1/n)$.

Theorem

$\Pr[|N_d - np_d| \geq n^{2/3}] = \mathcal{O}(n^{-1/3})$ where $p_d = \Pr[\deg(1) = d] \approx e^{-\lambda} \frac{\lambda^d}{d!}$.

Proof

$\mathbb{E}[N_d] = np_d \qquad \mathbb{E}[N_d^2] = n^2 p_d^2 \pm \mathcal{O}(n) \qquad \mathrm{Var}(N_d) = \mathcal{O}(n).$

$$\mathbb{E}[N_d^2] = \mathbb{E}\big[(\sum_{v \in [n]} [\deg(v) = d])^2\big] = \mathbb{E}\big[\sum_{u \in [n]} \sum_{v \in [n]} [\deg(u) = d, \deg(v) = d]\big]$$

$$= \sum_{u \in [n]} \sum_{v \in [n]} \Pr[\deg(u) = d, \deg(v) = d] = \sum_{u \in [n]} \Pr[\deg(u) = d] + \sum_{u \in [n]} \sum_{v \neq u} \Pr[\deg(u) = d, \deg(v) = d]$$

$$= n \cdot p_d + n \cdot (n-1) \cdot (p_d^2 \pm \mathcal{O}(1/n)) = n^2 p_d^2 \pm \mathcal{O}(n).$$

Concentration of N_d

Lemma (Near Independence of Degrees)

Let $u \neq v$ be two vertices of $G(n, \lambda/n)$. Then $\Pr[\deg(u) = d, \deg(v) = d] = \Pr[\deg(u) = d] \Pr[\deg(v) = d] \pm \Theta(1/n)$.

Theorem

$\Pr[|N_d - np_d| \geq n^{2/3}] = \mathcal{O}(n^{-1/3})$ where $p_d = \Pr[\deg(1) = d] \approx e^{-\lambda} \frac{\lambda^d}{d!}$.

Proof

$\mathbb{E}[N_d] = np_d$ $\qquad$ $\mathbb{E}[N_d^2] = n^2 p_d^2 \pm \mathcal{O}(n)$ $\qquad$ $\mathrm{Var}(N_d) = \mathcal{O}(n)$.

$$\mathrm{Var}(N_d) = \mathbb{E}[N_d^2] - \mathbb{E}[N_d]^2 \leq n^2 p_d^2 + \mathcal{O}(n) - (np_d)^2 = \mathcal{O}(n).$$

Concentration of N_d

Lemma (Near Independence of Degrees)

Let $u \neq v$ be two vertices of $G(n, \lambda/n)$. Then $\Pr[\deg(u) = d, \deg(v) = d] = \Pr[\deg(u) = d] \Pr[\deg(v) = d] \pm \Theta(1/n)$.

Theorem

$\Pr[|N_d - np_d| \geq n^{2/3}] = \mathcal{O}(n^{-1/3})$ where $p_d = \Pr[\deg(1) = d] \approx e^{-\lambda} \frac{\lambda^d}{d!}$.

Proof

$\mathbb{E}[N_d] = np_d$ $\qquad$ $\mathbb{E}[N_d^2] = n^2 p_d^2 \pm \mathcal{O}(n)$ $\qquad$ $\mathrm{Var}(N_d) = \mathcal{O}(n)$.

$$\text{Hence: } \Pr[|N_d - np_d| \geq n^{2/3}] = \Pr[|N_d - \mathbb{E}[N_d]| \geq n^{2/3}] \overset{\text{Cheb.}}{\leq} \frac{\mathrm{Var}(N_d)}{n^{4/3}} = \mathcal{O}(n^{-1/3}).$$

Content

1. Motivation

2. Erdős-Renyi Random Graphs
   - Degree Distribution
   - Degree Statistics
   - Tree-like local structure
   - Emergence of the Giant Component

3. Random Geometric Graphs

4. Scale-Free Networks (Teaser)

Erdős-Renyi Graphs have Few Cycles

Theorem: There are few short cycles in Erdős-Renyi graphs

Let C_k be the number of cycles of length k in $G(n, \lambda/n)$ where $k, \lambda = \Theta(1)$. Then $\mathbb{E}[C_k] \leq \frac{\lambda^k}{2k} = \Theta(1)$.

Proof.

The number of potential cycles is $\underbrace{n(n-1)\cdot\ldots\cdot(n-k+1)}_{\text{sequences } (v_1,\ldots,v_k)} \cdot \underbrace{\frac{1}{k}\cdot\frac{1}{2}}_{\substack{\text{startpoint and direction} \\ \text{irrelevant}}}$

The probability that $(v_1, \ldots, v_k, v_1)$ is a cycle is $(\lambda/n)^k$. Hence:

$$\mathbb{E}[C_k] \leq \frac{n^k}{2k}\left(\frac{\lambda}{n}\right)^k = \frac{\lambda^k}{2k}.$$

□

The Galton-Watson Branching Process

Definition

Let $\mathcal{D}$ be a distribution on $\mathbb{N}_0$ and $X_{i,j} \sim \mathcal{D}$ for $i, j \in \mathbb{N}$. Define $Z_0 = 1$ and $Z_i = \sum_{j=1}^{Z_{i-1}} X_{i,j}$ for $i \geq 1$.

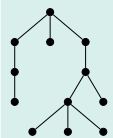
Intuition

- Start with a population of size $Z_1 = 1$.
- Each individual has a random number of decendents.
- Key question: What is the probability of extinction, i.e. for $\lim_{i \to \infty} Z_i = 0$?

Exercise: Galton-Watson Process with $\mathcal{D} = \text{Pois}(\lambda)$

If $\lambda \leq 1$ then the process goes extinct with probability 1.
If $\lambda > 1$ then the process survives with probability $s_\lambda > 0$.

Galton-Watson *Tree*



$X_{i,j}$	1	2	3	4	...
1	3	1	0	2	...
2	1	0	1	3	...
3	1	2	2	0	...
4	0	3	0	0	...
5	0	0	0	2	...
⋮	⋮	⋮	⋮	⋮	

Local Structure of Erdős-Renyi Graphs

Theorem: The Neighbourhood of v looks like a Galton Watson Tree

Let $R = \mathcal{O}(1)$. Let H be an ordered tree of depth R given by a sequence $c_1, \ldots, c_k$ specifiying the number of children of nodes in all layers except the last, in level order.
Let $\mathrm{GWT}(\lambda)|_R$ be the first R layers of a $\mathrm{Pois}(\lambda)$-Galton-Watson tree.
Let $G(n, \lambda/n)|_{v,R}$ be the subgraph of $G(n, \lambda/n)$ induced by vertices with distance $\leq R$ from v.

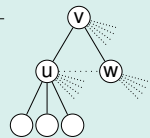
$$\Pr[\mathrm{GWT}(\lambda)|_R = H] \overset{\text{(i)}}{=} \prod_{i=1}^{k} \Pr_{X \sim \mathrm{Pois}(\lambda)}[X = c_i] = \prod_{i=1}^{k} e^{-\lambda} \frac{\lambda^{c_i}}{c_i!} \overset{\text{(ii)}}{\approx} \Pr[G(n, \lambda/n)|_{v,R} = H].$$

Example for H

$(c_1, c_2, c_3) = (2, 3, 0)$

Proof of (ii) by Example: The following has to "go right" for $G(n, \lambda/n)|_{v,R} = H$

random variable	desired outcome	probability
$\deg(v) \sim \mathrm{Bin}(n-1, \lambda/n) \approx \mathrm{Pois}(\lambda)$	2	$\approx e^{-\lambda} \frac{\lambda^2}{2!}$
$[\{u, w\} \in E]$	0	$1 - \frac{\lambda}{n} \approx 1$
$\deg(u) - 1 \sim \mathrm{Bin}(n-3, \lambda/n) \approx \mathrm{Pois}(\lambda)$	3	$\approx e^{-\lambda} \frac{\lambda^3}{3!}$
$\deg(w) - 1 \sim \mathrm{Bin}(n-3, \lambda/n) \approx \mathrm{Pois}(\lambda)$	0	$\approx e^{-\lambda} \frac{\lambda^0}{0!}$



Local Structure of Erdős-Renyi Graphs

Theorem: The Neighbourhood of v looks like a Galton Watson Tree

Let $R = \mathcal{O}(1)$. Let H be an ordered tree of depth R given by a sequence $c_1, \ldots, c_k$ specifiying the number of children of nodes in all layers except the last, in level order.
Let $\mathrm{GWT}(\lambda)|_R$ be the first R layers of a $\mathrm{Pois}(\lambda)$-Galton-Watson tree.
Let $G(n, \lambda/n)|_{v,R}$ be the subgraph of $G(n, \lambda/n)$ induced by vertices with distance $\leq R$ from v.

$$\Pr[\mathrm{GWT}(\lambda)|_R = H] \overset{\text{(i)}}{=} \prod_{i=1}^{k} \Pr_{X \sim \mathrm{Pois}(\lambda)}[X = c_i] = \prod_{i=1}^{k} e^{-\lambda} \frac{\lambda^{c_i}}{c_i!} \overset{\text{(ii)}}{\approx} \Pr[G(n, \lambda/n)|_{v,R} = H].$$

Example for H

$(c_1, c_2, c_3) = (2, 3, 0)$

Corollaries

- $G(n, \lambda/n)|_{v,R} \xrightarrow{d} \mathrm{GWT}(\lambda)|_R$ // convergence in distribution for $n \to \infty$
- The number N_H of "copies" of H in $G(n, \lambda/n)$ satisfies $\mathbb{E}[N_H] \approx n \cdot \prod_{i=1}^{k} e^{-\lambda} \frac{\lambda^{c_i}}{c_i!}$.
  Concentration of N_H can be proved much like we proved concentration of N_d earlier.

Content

1. Motivation

2. Erdős-Renyi Random Graphs
   - Degree Distribution
   - Degree Statistics
   - Tree-like local structure
   - Emergence of the Giant Component

3. Random Geometric Graphs

4. Scale-Free Networks (Teaser)

How does $G(n, \lambda/n)$ look like for different λ?

Karlsruhe Institute of Technology

Theorem: Sudden Emergence of the Giant Component (Erdős, Renyi 1960)

Consider $G(n, \lambda/n)$. The following holds with probability approaching 1 for $n \to \infty$.

i. If $\lambda < 1$ then $G(n, \lambda/n)$ only has components of size $\mathcal{O}(\log n)$.
Each component is a tree or pseudotree.pseudotree means: connected and as many edges as vertices
$\hookrightarrow$ Intuition: GWT(λ) dies out with probability 1.

ii. If $\lambda > 1$ then $G(n, \lambda/n)$ has one "giant" component of size $\approx s(\lambda) \cdot n$.
$\hookrightarrow$ Intuition: $s(\lambda)$ is the probability that GWT(λ) is infinite.

iii. If $\lambda = 1$ then the largest component of $G(n, \lambda/n)$ has size $\Theta(n^{2/3})$.
$\hookrightarrow$ Intuition: ?

Content

1. Motivation

2. Erdős-Renyi Random Graphs
   - Degree Distribution
   - Degree Statistics
   - Tree-like local structure
   - Emergence of the Giant Component

3. Random Geometric Graphs

4. Scale-Free Networks (Teaser)

Locality: A Property of Networks in Practice

Observation: Locality in Practice

Take social networks. A friend of my friend is more likely to be my friend than a random person.

Definition: Locality[1]

$L = \Pr[\{u, w\} \in E \mid \{v, u\} \in E \wedge \{v, w\} \in E]$ where v, u, w are distinct (random) vertices.

Similar numbers are sometimes called *clustering coefficient*.

Observation: No Locality in Erdős-Renyi Random Graphs

In $G(n, \lambda/n)$ we have $L = \frac{\lambda}{n} = \mathcal{O}(\frac{1}{n})$.

Next: Random *Geometric* Graphs with $L = \Omega(1)$.

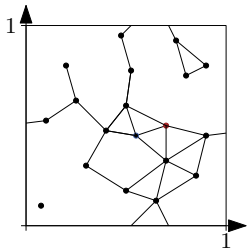
Random Geometric Graphs

Definition: Random Geometric Graph (RGG)

An RGG is obtained by distributing vertices in a metric space and connecting any two vertices with a probability depending on their distance.

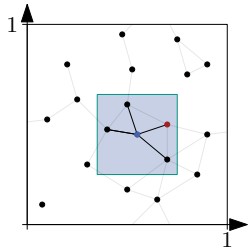
Simple Example: $G^{\mathbb{T}^2}(n, r)$

- number of vertices: n
- space: 2-dimensional torus $\mathbb{T}^2 = [0, 1)^2$
  // standard unit square is more common but less simple
- metric: L_∞ // L_2 is more common but less simple
  $\hookrightarrow \text{dist}((x_1, y_1), (x_2, y_2)) = \max(\text{dist}(x_1, x_2), \text{dist}(y_1, y_2))$.
- vertex distribution: for $v \in [n]$: $P_v \sim \mathcal{U}(\mathbb{T}^2)$
- edge "probability" is 0 or 1: $\{u, v\} \in E \Leftrightarrow \text{dist}(P_u, P_v) \leq r$
  // not random when P_u and P_v are given



Degree Distribution of $G^{\mathbb{T}^2}(n, r)$

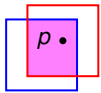
- Consider arbitrary $v \in [n]$.
- By symmetry of $\mathbb{T}^2$ each outcome of P_v behaves the same.
- $\Pr[\{u, v\} \in E] = \Pr[P_u \text{ is in the } 2r \times 2r \text{ square centered at } P_v] = 4r^2$.
- Hence $\deg(v) \sim \text{Bin}(n-1, 4r^2)$ and $\mathbb{E}[\deg(v)] = 4r^2(n-1)$.



Locality in $G^{\mathbb{T}^2}(n, r)$

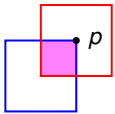
Observation

Let $p, q \sim \mathcal{U}([0,1]^2)$ and S_p the unit square around p.
Then $\Pr[q \in S_p] = \int_{-0.5}^{0.5}\int_{-0.5}^{0.5}(1-|x|)(1-|y|)dx\,dy = \frac{9}{16}$.

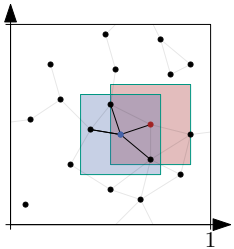


general case

best case



worst case



Corollary

By "rescaling" the observation we get $L = \Pr[\underbrace{\{u,w\} \in E}_{\substack{P_w \text{ in square} \\ \text{around } P_u}} \mid \underbrace{\{v,u\} \in E \wedge \{v,w\} \in E}_{P_u, P_w \text{ in square around } P_v}] = \frac{9}{16} = \Omega(1)$.

Poissonisation

Poissonised Variant $G^{\mathbb{T}^2}_{\text{Pois}}(n,r)$ of $G^{\mathbb{T}^2}(n,r)$

Replace the point set with a Poisson point process on $\mathbb{T}^2$ with intensity n.
$\hookrightarrow$ i.e. region of size λ contains $\text{Pois}(\lambda n)$-many points, independent for disjoint regions
Equivalently: $G^{\mathbb{T}^2}_{\text{Pois}}(n,r) = G^{\mathbb{T}^2}(N,r)$ where $N \sim \text{Pois}(n)$.

Advanatages

- No long-distance correlations. ✓
- $\text{Pois}(4r^2)$-distributed degrees. ✓

Disadvantages

- Less natural in practice. ✗
- Number of vertices $N \sim \text{Pois}(n)$ not fixed. ✗

De-Poissonisation (an analogous result holds for de-Poissonising balls-into-bins)

Let P be a graph property. If P is very unlikely for $G^{\mathbb{T}^2}_{\text{Pois}}(n,r)$ then P is unlikely for $G^{\mathbb{T}^2}(n,r)$:

$$\Pr[G^{\mathbb{T}^2}(n,r) \in P] = \Pr[G^{\mathbb{T}^2}_{\text{Pois}}(n,r) \in P \mid N = n] \leq \frac{\Pr[G^{\mathbb{T}^2}_{\text{Pois}}(n,r) \in P]}{\Pr[N=n]} = \Theta(n^{1/2}) \Pr[G^{\mathbb{T}^2}_{\text{Pois}}(n,r) \in P].$$

Content

1. Motivation

2. Erdős-Renyi Random Graphs
   - Degree Distribution
   - Degree Statistics
   - Tree-like local structure
   - Emergence of the Giant Component

3. Random Geometric Graphs

4. Scale-Free Networks (Teaser)

Scale-Free Networks

Semi-Formal Definition

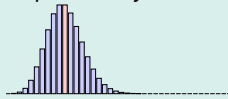
A scale-free network is a graph with a degree distribution that follows a power law (in an asymptotic sense)

Practical Consequence

There are vertices of very high degree (*hubs*).

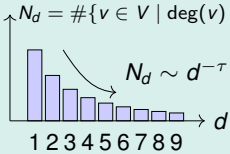
Contrast: Erdős-Renyi

exponentially decreasing tail.



for $\lambda = 10$.

Power Laws



$\tau \leq 1$: not a distribution

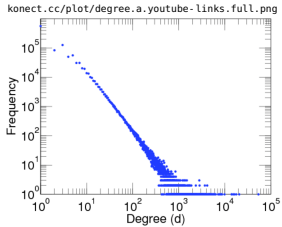
$1 < \tau \leq 2$: distribution, but $\mathbb{E}[\deg(v)] = \infty$

$2 < \tau \leq 3$: $\mathbb{E}[\deg(v)] < \infty$, but $\mathrm{Var}(\deg(v)) = \infty$

$3 < \tau \leq 4$: variance $< \infty$, but higher moments are ∞

$\tau \in (2, 3]$ is especially popular

"Youtube"



Scale-Free Networks

Semi-Formal Definition

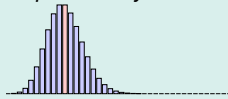
A scale-free network is a graph with a degree distribution that follows a power law (in an asymptotic sense)

Practical Consequence

There are vertices of very high degree (*hubs*).

Contrast: Erdős-Renyi

exponentially decreasing tail.



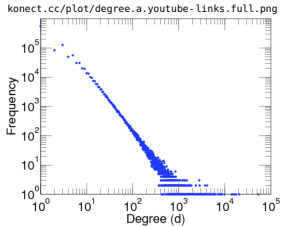
$\text{Pois}(\lambda)$ for $\lambda = 10$.

The Name “Scale-Free”

From *Barabási: “Linked: The New Science of Networks”, 2002.*

In a random network [...] the vast majority of nodes have the same number of links [...]. Therefore, a random network has a characteristic scale in its node connectivity [...]. In contrast, the absence of a peak in a power-law degree distribution implies that [...] we see a continuous hierarchy of nodes, spanning from rare hubs to the numerous tiny nodes. There is no intrinsic scale in these networks. This is the reason my research group started to describe networks with power-law degree distribution as scale-free.

“Youtube”



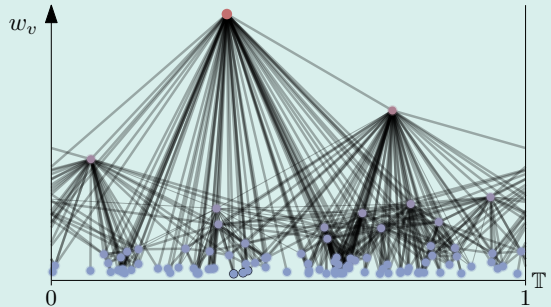
A Scale-Free Random Geometric Graph

Reminder: Random Geometric Graph (RGG)

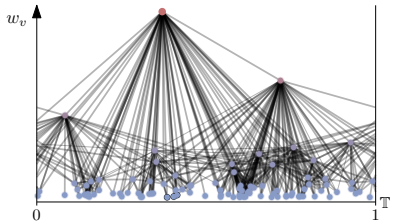
Distribute vertices in a metric space and connect any two vertices with a probability depending on their distance.

Definition: Geometric Inhomogeneous Random Graph (GIRG)

- number of vertices: n
- metric space $\mathbb{T}$ // more generally: $\mathbb{T}^d$ for $d \in \mathbb{N}$
- for each v: position $x_v \sim \mathcal{U}(\mathbb{T})$
- for each v: weight $w_v \sim \mathrm{Par}(\tau - 1, 1)$
  the Pareto distribution is a power law distribution with exponent τ
- $\{u, v\} \in E \Leftrightarrow \mathrm{dist}(x_u, x_v) \leq \frac{\lambda}{n} w_u w_v$
  $\Leftrightarrow \frac{n}{\lambda w_v} \leq \frac{w_u}{\mathrm{dist}(x_u, x_v)}$.



How GIRGs are Useful



GIRGs are Scale-Free

$\mathbb{E}[\deg(v) \mid w_v] = \Theta(w_v)$ and $\deg(v)$ follows a power law if w_v does.

GIRGs are a Good Model for Real World Networks (Bläsius, Fischbeck, 2022)

- consider two graph parameters: locality and heterogeneity ($\approx \log \mathrm{Var}(\deg(v))$).
- in many contexts, a real network behaves like a GIRG with the same parameters

On the External Validity of Average-Case Analyses of Graph Algorithms, ESA 2022.

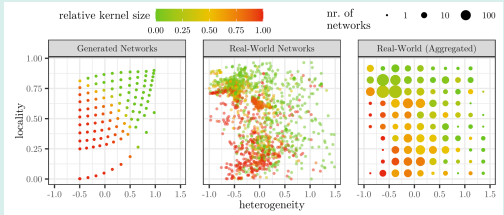


Figure 7 The relative kernel size of the vertex cover domination rule.

Hyperbolic Geometric Graphs

Poincaré Model of Hyperbolic Geometry

Illustration by M.C. Escher, Circle Limit III, 1959.

(All creatures are congruent in hyperbolic space.)

Result (Bläsius, Friedrich, Katzmann, 2021)

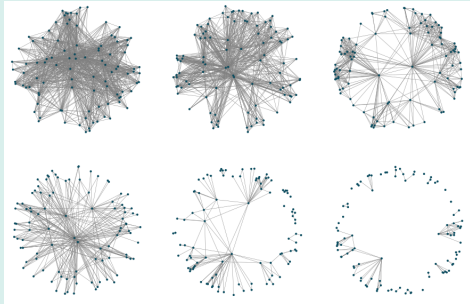
Vertex Cover can be Approximated on HGGs.

Efficiently Approximating Vertex Cover on Scale-Free Networks with Underlying Hyperbolic Geometry, ESA 2021.

Hyperbolic Random Graph (HGGs)

Sample points with bias towards the centre.
Connect points if distance is beneath a threshold.

https://commons.wikimedia.org/wiki/File:Hyperbolic_graphs.png



Can yield power law distribution for node degrees.

Preferential Attachment and the Barabasi-Albert Model

How a Graph is Grown Over Time

- There is a parameter $m \in \mathbb{N}$.
- start with any graph on $\geq m$ nodes.
- add new nodes one by one
  - new node is connected to m existing nodes
  - existing nodes are selected with probability proportional to their degree

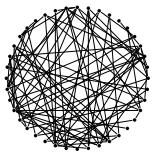
Why the Model is Interesting

- node degrees approach a power law distribution with exponent 3
- model may explain *why* scale-free networks emerge in practice

Summary

Erdős-Renyi Random Graphs

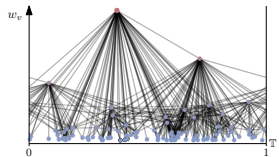
- simplest type of random graphs
- “Erdős-Renyi” refers to various related models
- arise in certain data structures (stay tuned)
- look locally like Poisson Galton-Watson Trees
- no locality or high-degree vertices



Random Graphs for Average Case Analsis

Mimic properties of real world networks:

- locality // a friend of my friend is often my friend
  - arises naturally in random geometric graphs
- “scale-freeness” $\approx$ existence of hubs
  - assign weights to vertices (in GIRGs)
  - use hyperbolic geometry
  - use preferential attachment



Anhang: Mögliche Prüfungsfragen I

- Was ist mit Theory-Practice Gap im Kontext von Graphalgorithmen gemeint?
- Wie kann die Theorie der Praxis entgegenkommen?
- Was ist das klassische Modell von Erdős und Renyi?
  - Welche Varianten des Erdős-Renyi Modells haben wir betrachtet?
  - Was gilt für die Verteilung von $\deg(v)$, wenn wir $\mathbb{E}[\deg(v)] = \lambda$ einstellen?
  - Was lässt sich über $N_d = |\{v \in [n] \mid \deg(v) = d\}|$ sagen?
  - Wir haben uns die R-Nachbarschaft $G(n, \lambda/n)|_{v,R}$ eines Knotens v angeschaut.
    - Was gilt für die Verteilung von $G(n, \lambda/n)|_{v,R}$ und warum?
    - Was ist ein Galton-Watson Baum?
    - Was lässt sich über die Aussterbewahrscheinlichkeit eines Poisson-Galton-Watson Baumes sagen?
  - Was versteht man unter „Sudden Emergence of the Giant Component“. Formuliere die Aussage formal.
  - Wir haben eine Größe L betrachtet, und Lokalität genannt. Wie ist sie definiert?
  - Welche Lokalität haben Erdős-Renyi Graphen?
- Nenne Eigenschaften, die Netzwerke in der Praxis von Erdős-Renyi Graphen unterscheiden.
  - Gib ein Beispiel für einen geometrischen Zufallsgraphen. Was ist die Lokalität in diesem Modell?

Anhang: Mögliche Prüfungsfragen II

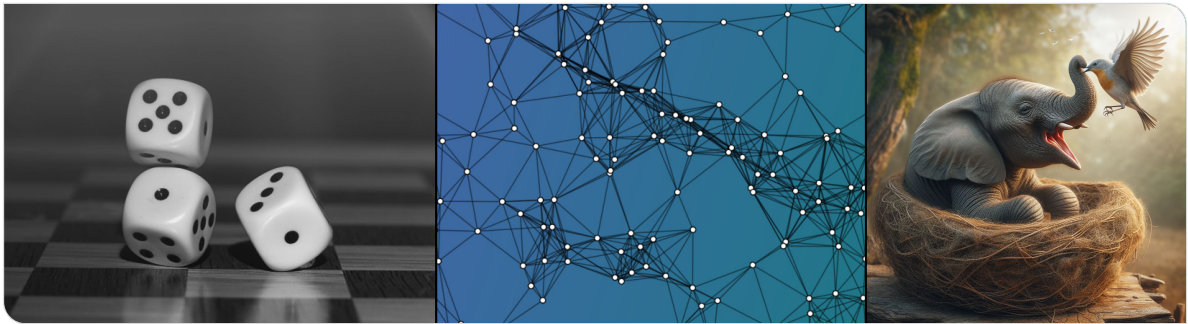
- Inwiefern könnte ein Poissonisiertes Modell bequemer sein?
- Wann ist ein Netzwerk “Scale-Free”?
  - Gib ein Beispiel für ein Netzwerk aus der Praxis dem man diese Eigenschaft zuschreibt.
  - Beschreibe mindestens zwei Arten, wie man Netzwerke dieser Art generieren kann.

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Cuckoo Hashing

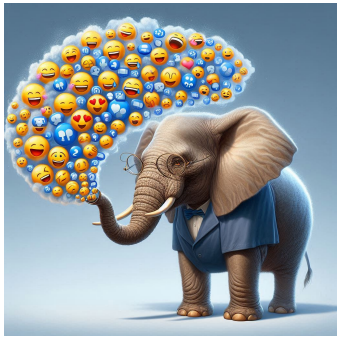
Stefan Walzer | WS 2024/2025



Content

1. Cuckoo Hashing
   - Algorithm
   - Analysis

Evaluation

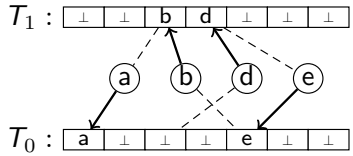


Note: No separate evaluation of the exercise sessions.

Cuckoo Hashing

Setup

$S \subseteq D$	key set of size n
T_0, T_1	two tables of size m
$h_0, h_1 \sim \mathcal{U}([m]^D)$	two hash functions (SUHA)
$\frac{n}{m} = 1 - \beta$	for some $\beta > 0$
	($\triangle$ load factor $\alpha = \frac{n}{2m}$)



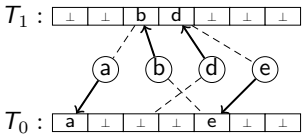
Algorithm lookup(x):
  return $x \in \{T_0[h_0(x)], T_1[h_1(x)]\}$

Algorithm delete(x):
  if $T_0[h_0(x)] = x$ **then**
    $T_0[h_0(x)] \leftarrow \perp$
  else if $T_1[h_1(x)] = x$ **then**
    $T_1[h_1(x)] \leftarrow \perp$

Algorithm insert(x):
  for $i = 0$ **to** LIMIT **do**
    $b \leftarrow i \bmod 2$
    swap(x, $T_b[h_b(x)]$)
    if $x = \perp$ **then**
      return SUCCESS
  return FAILURE

Cuckoo Hashing Theorem

Algorithm insert(x)**:**
- **for** $i = 0$ **to** LIMIT **do**
  - $b \leftarrow i \bmod 2$
  - swap(x, $T_b[h_b(x)]$)
  - **if** $x = \perp$ **then**
    - **return** SUCCESS
- **return** FAILURE



Theorem (Analysis with LIMIT $= \infty$)

Assume we insert all $x \in S$ and then another key y. Let E be the event that this succeeds and

$$T = \begin{cases} \text{insertion time of } y & \text{if } E \text{ occurs} \\ 0 & \textit{otherwise} \end{cases}$$

Then (i) $\Pr[E] = 1 - \mathcal{O}(1/m)$ and (ii) $\mathbb{E}[T] = \mathcal{O}(1)$.

Theorem (full analysis, not here)

If we
- set LIMIT $= \Omega(\log n)$ appropriately
- rebuild the table with fresh hash functions when LIMIT is reached

we obtain a hash table where lookup and delete take $\mathcal{O}(1)$ time and insert takes *expected* $\mathcal{O}(1)$ time.

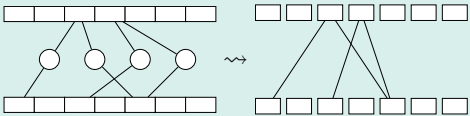
Proof of i: Success probability is $1 - \mathcal{O}(1/m)$

The Cuckoo Graph

Consider the bipartite *cuckoo graph*

$$G = ([m], [m], \{(h_0(x), h_1(x)) \mid x \in S\})$$

the key x corresponds to the edge $(h_0(x), h_1(x))$ and each table position to a vertex.



// Duplicate edges possible, don't worry about it.

Connection to Erdős-Renyi Graphs

G is a bipartite Erdős-Renyi variant

- much like $G^{\text{UE}}(2m, n)$ // uniform endpoint model
- a bit like $G(2m, n)$ // original Erdős-Renyi
- a bit like $G(2m, n/(2m^2))$ // Gilbert

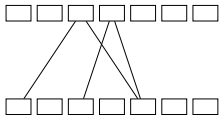
Confusing: n is a number of edges and $2m$ a number of vertices.

Exercise

Design a variant of cuckoo hashing such that the "Sudden Emergence" result for the $G^{\text{UE}}(m, n)$ model implies success for load factor $\alpha < \frac{1}{2}$.

Next: Completely self-contained analysis without reference to Erdős-Renyi.

Proof of i: Success probability is $1 - \mathcal{O}(1/m)$



Keys and buckets in the infinite loop

Assume $\bar{E}$ occurs, i.e. an insertion fails due to an infinite loop. Let $G^* = (V^*, E^*)$ be the subgraph of G with

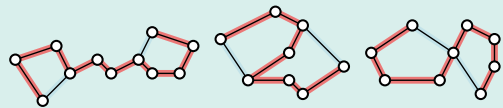
- V^*: table positions touched in the infinite loop
- E^*: keys touched in the infinite loop.

Properties of G^*:

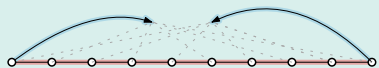
- connected
- $|E^*| = |V^*| + 1$ // can you see why?
- $\deg_{E^*}(v) \geq 2$ for $v \in V^*$.

Possibilities for G^*

There are three options:

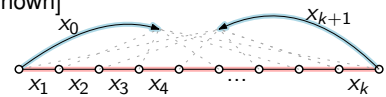


In all three cases: Simple path through $|V^*|$ and two extra edges connecting inwards:



Proof of i: Success probability is $1 - \mathcal{O}(1/m)$

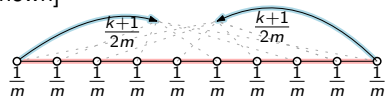
$$\begin{aligned}
\Pr[\bar{E}] &= \Pr[\exists \text{path as shown}] \\
&= \Pr[\exists k \in \mathbb{N} : \exists x_0, \ldots, x_{k+1} \in S : x_0, \ldots, x_{k+1} \text{ form a path as shown}] \\
&\overset{\text{union bound}}{\leq} \sum_{k=1}^{n} \sum_{x_0,\ldots,x_{k+1} \in S} \Pr[x_0, \ldots, x_{k+1} \text{ form a path as shown}] \\
&\leq \sum_{k=1}^{n} \underbrace{n^{k+2}}_{\text{a}} \cdot \underbrace{2}_{\text{b}} \cdot \underbrace{\tfrac{1}{m^{k+1}}}_{\text{c}} \cdot \underbrace{\left(\tfrac{k+1}{2m}\right)^2}_{\text{d}} \\
&\leq \tfrac{1}{2} \sum_{k=1}^{n} m^{k+2-k-1-2} (1-\beta)^{k+2} (k+1)^2 \\
&\leq \tfrac{1}{2m} \sum_{k=1}^{\infty} (1-\beta)^{k+2} (k+1)^2 = \tfrac{1}{m} \cdot \mathcal{O}\left(\tfrac{1}{\beta^3}\right) = \mathcal{O}\left(\tfrac{1}{m}\right) \quad \square
\end{aligned}$$



- a Choose sequence of $k+2$ keys.
- b Choose to start in top or bottom table.
- c Neighbouring keys share a hash.
- d Two bordering keys connect back inward.

Proof of i: Success probability is $1 - \mathcal{O}(1/m)$

$$\begin{aligned}
\Pr[\bar{E}] &= \Pr[\exists \text{path as shown}] \\
&= \Pr[\exists k \in \mathbb{N} : \exists x_0, \ldots, x_{k+1} \in S : x_0, \ldots, x_{k+1} \text{ form a path as shown}] \\
&\overset{\text{union bound}}{\leq} \sum_{k=1}^{n} \sum_{x_0,\ldots,x_{k+1} \in S} \Pr[x_0, \ldots, x_{k+1} \text{ form a path as shown}] \\
&\leq \sum_{k=1}^{n} \underbrace{n^{k+2}}_{\text{a}} \cdot \underbrace{2}_{\text{b}} \cdot \underbrace{\tfrac{1}{m^{k+1}}}_{\text{c}} \cdot \underbrace{\left(\tfrac{k+1}{2m}\right)^2}_{\text{d}} \\
&\leq \tfrac{1}{2} \sum_{k=1}^{n} m^{k+2-k-1-2} (1-\beta)^{k+2} (k+1)^2 \\
&\leq \tfrac{1}{2m} \sum_{k=1}^{\infty} (1-\beta)^{k+2} (k+1)^2 = \tfrac{1}{m} \cdot \mathcal{O}\left(\tfrac{1}{\beta^3}\right) = \mathcal{O}\left(\tfrac{1}{m}\right) \quad \square
\end{aligned}$$



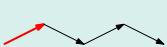
- a Choose sequence of $k + 2$ keys.
- b Choose to start in top or bottom table.
- c Neighbouring keys share a hash.
- d Two bordering keys connect back inward.

Proof of ii: Expected insertion time is $\mathcal{O}(1)$

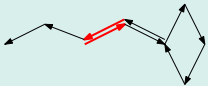
Lemma

If the insertion of y takes $t \in \mathbb{N}$ steps then the cuckoo graph G contained (previously) a path of length $\lceil (t-2)/3 \rceil$ starting from $h_0(y)$ or from $h_1(y)$.

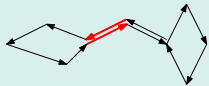
Proof.



no turning back
$\rightsquigarrow$ path of length $t-1$
starting from $h_0(y)$



turn back once
$\rightsquigarrow$ path of length $\lceil (t-2)/3 \rceil$
starting from $h_0(y)$ or $h_1(y)$



turn back twice
impossible: insertion would fail

□

Proof of ii: Expected insertion time is $\mathcal{O}(1)$ (continued)

$$\mathbb{E}[T] = \sum_{t \geq 1} \Pr[T \geq t] \qquad \text{tail sum formula}$$

$$\leq \sum_{t \geq 1} \Pr[\exists \text{path of length } \lceil (t-2)/3 \rceil \text{ starting from } h_0(y) \text{ or } h_1(y)] \qquad \text{by Lemma}$$

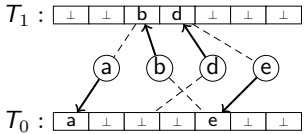
$$\leq 2 \cdot \sum_{t \geq 1} \Pr[\exists \text{path of length } \lceil (t-2)/3 \rceil \text{ starting from } h_0(y)] \qquad \text{union bound + symmetry}$$

$$\leq 2\left(2 + 3 \cdot \sum_{t \geq 1} \Pr[\exists \text{path of length } t \text{ starting from } h_0(y)]\right) \qquad \sum_{i \geq 1} f(\lceil t/3 \rceil) = 3 \cdot f(1) + 3 \cdot f(2) + \ldots$$

$$\leq 4 + 6 \cdot \sum_{t \geq 1} \sum_{x_1, \ldots, x_t \in S} \Pr[x_1, \ldots, x_t \text{ form path starting from } h_0(y)] \qquad \text{union bound}$$

$$\leq 4 + 6 \cdot \sum_{t \geq 1} n^t m^{-t} = 6 \sum_{t \geq 0} (1-\beta)^t = 6/\beta = \mathcal{O}(1). \quad \square$$

Conclusion



Cuckoo Hashing

- hash table with *worst case* constant access times
- analysis considers path in graphs similar to the Erdős-Renyi model
- many variations and spin-offs (not discussed here)

Anhang: Mögliche Prüfungsfragen I

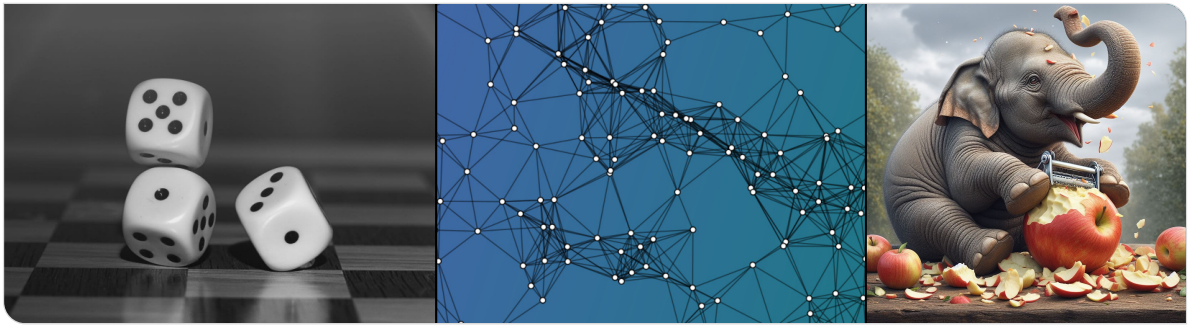
- Was ist und was kann Cuckoo Hashing?
  - Was ist die Grundidee? Wie funktionieren die Operationen?
  - Worauf ist bei der Wahl der Tabellengröße / beim Load Factor zu achten?
  - Was kann man über die Laufzeit der Operationen sagen?
  - Welche Vorteile und Nachteile ergeben sich im Vergleich zu anderen Techniken wie linearem Sondieren?
- Analyse:
  - Eine Einfügung, die fehlschlägt, entspricht gewissen Strukturen im Cuckoo-Graphen. Welchen?
  - Wie haben wir gezeigt, dass solche Strukturen unwahrscheinlich sind?
  - Wie haben wir die erwartete Einfügezeit abgeschätzt?

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – The Peeling Algorithm

Stefan Walzer | WS 2024/2025



Content

1. Cuckoo hashing with more than two hash functions

2. The Peeling Algorithm

3. The Peeling Theorem

4. Conclusion

Content

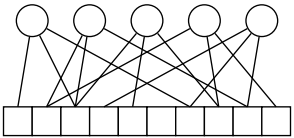
1. Cuckoo hashing with more than two hash functions

2. The Peeling Algorithm

3. The Peeling Theorem

4. Conclusion

Cuckoo Hashing with one table and k hash functions



$n \in \mathbb{N}$ keys
$m \in \mathbb{N}$ table size
$\alpha = \frac{n}{m}$ load factor
$h_1, \ldots, h_k \sim \mathcal{U}([m]^D)$ hash functions
↪ Could also use a separate table per hash function.

randomWalkInsert(x)

while $x \neq \perp$ **do** // TODO: limit
  sample $i \sim \mathcal{U}([k])$
  swap($x, T[h_i(x)]$)

(some improvements possible)

Theorem (without proof)

For each $k \in \mathbb{N}$ there is a **threshold** c_k^* such that:

- if $\alpha < c_k^*$ all keys can be placed with probability $1 - \mathcal{O}(\frac{1}{m})$.
- if $\alpha > c_k^*$ **not** all keys can be placed with probability $1 - \mathcal{O}(\frac{1}{m})$.

$c_2^* = \frac{1}{2}, \quad c_3^* \approx 0.92, \quad c_4^* \approx 0.98, \ldots$

Theorem (Bell, Frieze, 2024)

If $k \geq 4$ and $\alpha < c_k^*$ then, conditioned on a high probability event[a], the expected insertion time is $\mathcal{O}(1)$.

[a] Without this conditioning, randomWalkInsert might be trapped in an infinite loop.

Static Hash Tables

Static Hash Table

`construct`(S): builds table T with key set S no insertions or deletions after construction

`lookup`(x): checks if x is in T or not

Constructing cuckoo hash tables:

- solved by Khosla 2013: “Balls into Bins Made Faster”
- matching algorithm resembling preflow push
- expected running time $\mathcal{O}(n)$, finds placement whenever one exists
- not in this lecture

Greedily constructing cuckoo hash tables

- Peeling: simple algorithm but sophisticated analysis
- interesting applications beyond hash tables (see “retrieval” in next lecture)

Content

1. Cuckoo hashing with more than two hash functions

2. The Peeling Algorithm

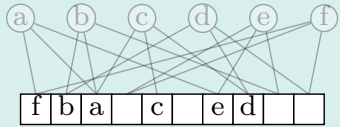
3. The Peeling Theorem

4. Conclusion

The Peeling Algorithm

constructByPeeling($S \subseteq D$, $h_1, h_2, h_3 \in [m]^D$)

$T \leftarrow [\bot, \ldots, \bot]$ // empty table of size m
while $\exists i \in [m] : \exists$ *exactly one* $x \in S : i \in \{h_1(x), h_2(x), h_3(x)\}$ **do**
  // x is only unplaced key that may be placed in i
  $T[i] \leftarrow x$
  $S \leftarrow S \setminus \{x\}$
if $S = \varnothing$ **then**
  return T
else
  return NOT-PEELABLE



Exercise

- Success of constructByPeeling does not depend on choices for i made by while.
- constructByPeeling can be implemented in linear time.

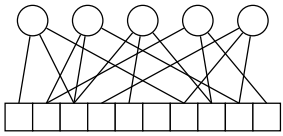
Peelability and the Cuckoo Graph

Cuckoo Graph and Peelability

- The **Cuckoo Graph** is the bipartite graph

$$G_{S,h_1,h_2,h_3} = (S, [m], \{(x, h_i(x)) \mid x \in S, i \in [3]\})$$

- Call G_{S,h_1,h_2,h_3} **peelable** if constructByPeeling(S, h_1, h_2, h_3) succeeds.
- If $h_1, h_2, h_3 \sim \mathcal{U}([m]^D)$ then the distribution of G_{S,h_1,h_2,h_3} does not depend on S. We then simply write $G_{m,\alpha m}$.
  - m □-nodes and $\lfloor \alpha m \rfloor$ -○-nodes
  - think: α is constant and $m \to \infty$.



Peeling simplified (not computing placement)

while ∃ □*-node of degree* 1 **do**
  remove it and its incident ○

G is peelable if and only if
this algorithm removes all ○-nodes.

Content

1. Cuckoo hashing with more than two hash functions

2. The Peeling Algorithm

3. The Peeling Theorem

4. Conclusion

Peeling Theorem

Peeling Threshold

Let $c_3^\Delta = \min_{y \in [0,1]} \frac{y}{3(1-e^{-y})^2} \approx 0.81$.

Theorem (today's goal)

Let $\alpha < c_3^\Delta$. Then $\Pr[G_{m,\alpha m} \text{ is peelable}] = 1 - o(1)$.

Remark: More is known.

- For "$\alpha < c_3^\Delta$" we get "peelable" with probability $1 - \mathcal{O}(1/m)$.
- For "$\alpha > c_3^\Delta$" we get "not peelable" with probability $1 - \mathcal{O}(1/m)$.
- Corresponding thresholds c_k^Δ for $k \geq 3$ hash functions are also known.

Exercise: What about $k = 2$?

Peeling does not reliably work for $k = 2$ for any $\alpha > 0$.

Peeling Theorem: Proof outline

Theorem (today's goal)

Let $\alpha < c_3^\Delta$. Then $\Pr[G_{m,\alpha m} \text{ is peelable}] = 1 - o(1)$.

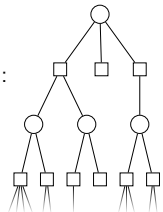
Proof Idea

The random (possibly) infinite tree T_α can be peeled for $\alpha < c_3^\Delta$ and T_α is locally like $G_{m,\alpha m}$.

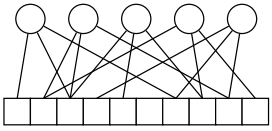
Steps

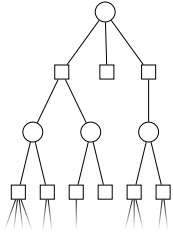
I. What is T_α?
II. What does peeling mean in this setting?
III. What role does c_3^Δ play?
IV. What does it mean for T_α to be locally like $G_{m,\alpha m}$?
V. What is the probability that a fixed key of $G_{m,\alpha m}$ is peeled?
VI. What is the probability that *all* keys of $G_{m,\alpha m}$ are peeled?

T_α :



What is T_α?





Observations for the finite Graph $G_{m,\alpha m}$

- each ◯ has 3 □ as neighbours (rare exception: $h_1(x), h_2(x), h_3(x)$ not distinct)
- each □ has random number X of ◯ as neighbours with $X \sim \text{Bin}(3n, \frac{1}{m}) = \text{Bin}(3\lfloor \alpha m \rfloor, \frac{1}{m})$. In an exercise we have seen that

$$X \xrightarrow{d} Y \text{ for } Y \sim \text{Pois}(3\alpha) \text{ and } m \to \infty$$

// i.e. $\forall i : \Pr[X = i] \xrightarrow{m\to\infty} \Pr[Y = i]$.

Definition of the (possibly) infinite random tree T_α

- root is ◯ and has three □ as children
- each □ has random number of ◯ children, sampled $\text{Pois}(3\alpha)$ (independently for each □).
- each non-root ◯ has two □ as children.

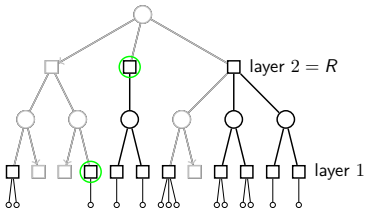
Remark: T_α is finite with positive probability > 0, e.g. when the first three $\text{Pois}(3\alpha)$ random variables come out as 0. But T_α is also infinite with positive probability.

ii What does peeling mean in this setting?

Peeling Algorithm

while $\exists$ *childless* □*-node* **do**
  remove it and its incident ○

↪ not well defined outcome on T_α!
↪ but well defined on T_α^R!



Peel only the first $R \in \mathbb{N}$ layers

- Let T_α^R be the first $2R + 1$ levels of T_α.
- *R layers* of □-nodes, labeled bottom to top.
- Run peeling on T_α^R (later $R \to \infty$).

↪ Why not consider the first $2R$ levels? (without $+1$)

Only care whether root is removed (root represents arbitrary node in $G_{m,\alpha m}$)

We may then simplify the peeling algorithm.

- replace "□-node of degree 1" condition with stronger "childless □-node".
  - prevents peeling of □-nodes with one child and no parent
  - no matter: such nodes are disconnected from the root anyway
- whether node is peeled only depends on subtree
  ↪ one bottom up pass suffices for peeling

ii What does peeling mean in this setting? (2)

Observation

Let $q_R = \Pr[\text{root survives when peeling } T_\alpha^R]$.
The values q_R are decreasing in R.

Peeling Algorithm

while $\exists$ *childless* □*-node* **do**
  remove it and its incident ○

Proof.

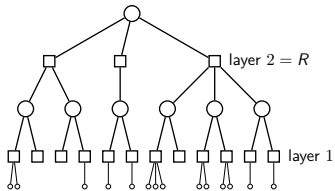
Assume when peeling T_α^R the sequence $\vec{x} = (x_1, \ldots, x_k)$ is a valid sequence of □-node choices. Then $\vec{x}$ is also valid when peeling T_α^{R+1}.

$$\text{peeling } T_\alpha^R \text{ removes the root} \Rightarrow \text{peeling } T_\alpha^{R+1} \text{ removes the root}$$

$$\text{root survives when peeling } T_\alpha^{R+1} \Rightarrow \text{peeling } T_\alpha^R \text{ removes the root}$$

$$q_{R+1} \leq q_R$$

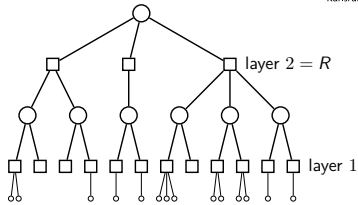
□



ii What does peeling mean in this setting? (3)

Peeling T_α^R bottom up

for $i = 1$ **to** R **do** // □-layers bottom to top
  for *each* □*-node v in layer i* **do**
    if *v has no children* **then**
      remove v and its parent ○



Survival probabilities $p_i :=$ Pr[□-node in layer i is *not* peeled]

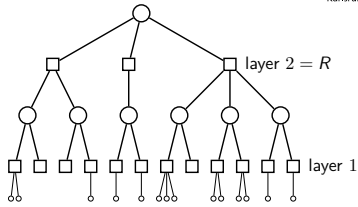
$$p_1 = \Pr[\square\text{-node has at least 1 child}]$$
$$= \Pr_{Y\sim\mathrm{Pois}(3\alpha)}[Y > 0] = 1 - e^{-3\alpha}.$$
$$p_i = \Pr[\text{layer } i \ \square\text{-node } v \text{ has at least 1 } \textit{surviving} \text{ child}]$$
$$= \Pr_{X\sim\mathrm{Pois}(3\alpha p_{i-1}^2)}[X > 0] = 1 - e^{-3\alpha p_{i-1}^2}.$$

$Y :=$ number of (initial) children of v
$X :=$ number of surviving children of v
each child ○-node survives if both its □-children from layer $i-1$ survive $\rightsquigarrow$ probability p_{i-1}^2.
$\Rightarrow Y \sim \mathrm{Pois}(3\alpha)$ and $X \sim \mathrm{Bin}(Y, p_{i-1}^2)$.
$\Rightarrow X \sim \mathrm{Pois}(3\alpha p_{i-1}^2)$. $\rightsquigarrow$ exercise!

ii What does peeling mean in this setting? (3)

Peeling T_α^R bottom up

for $i = 1$ **to** R **do** // □-layers bottom to top
  for *each* □*-node* v *in layer* i **do**
    if v *has no children* **then**
      remove v and its parent ○



Survival probabilities $p_i :=$ Pr[□-node in layer i is *not* peeled]

$$p_1 = \Pr[\text{□-node has at least 1 child}]$$
$$= \Pr_{Y\sim\text{Pois}(3\alpha)}[Y > 0] = 1 - e^{-3\alpha}.$$
$$p_i = \Pr[\text{layer } i \text{ □-node } v \text{ has at least 1 } \textit{surviving} \text{ child}]$$
$$= \Pr_{X\sim\text{Pois}(3\alpha p_{i-1}^2)}[X > 0] = 1 - e^{-3\alpha p_{i-1}^2}.$$

□-survival probabilities. With $p_0 := 1$ we have

$$p_i = \begin{cases} 1 & \text{if } i = 0 \\ 1 - e^{-3\alpha p_{i-1}^2} & \text{if } i = 1, 2, \ldots \end{cases}$$

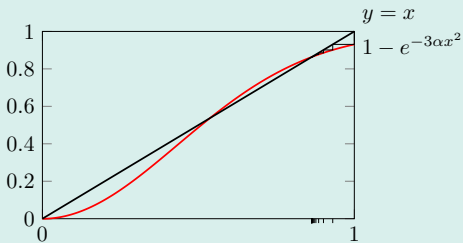
Moreover: $q_R := \Pr[\text{root survives}] = p_R^3$.

iii What role does $c_3^\Delta \approx 0.81$ play?

$$p_i = \begin{cases} 1 & \text{if } i = 0 \\ 1 - e^{-3\alpha p_{i-1}^2} & \text{if } i = 1, 2, \ldots \end{cases}$$

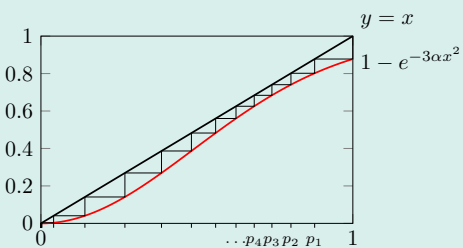
$\hookrightarrow$ consider $f(x) = 1 - e^{-3\alpha x^2}$

Case 1: $\exists x > 0 : f(x) = x$.



$\Rightarrow \lim_{i \to \infty} p_i = x^* = \max\{x \in [0,1] \mid f(x) = x\}$.

Case 2: $\forall x \in (0, 1] : f(x) < x$



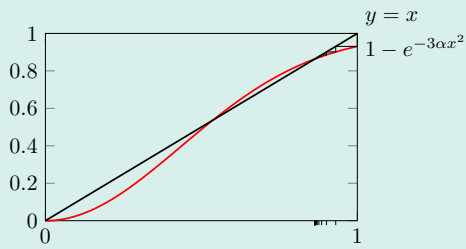
$\Rightarrow \lim_{i \to \infty} p_i = 0$.

iii What role does $c_3^\Delta \approx 0.81$ play?

$$p_i = \begin{cases} 1 & \text{if } i = 0 \\ 1 - e^{-3\alpha p_{i-1}^2} & \text{if } i = 1, 2, \ldots \end{cases}$$

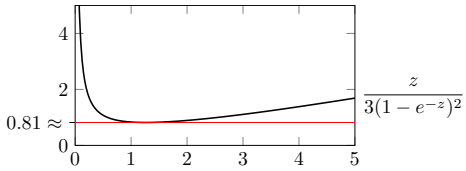
$\hookrightarrow$ consider $f(x) = 1 - e^{-3\alpha x^2}$

Case 1: $\exists x > 0 : f(x) = x$.



$$\Rightarrow \lim_{i\to\infty} p_i = x^* = \max\{x \in [0,1] \mid f(x) = x\}.$$

$$\text{Case 1} \Leftrightarrow \exists x > 0 : x = 1 - e^{-3\alpha x^2}$$
$$\Leftrightarrow \exists x > 0 : x^2 = (1 - e^{-3\alpha x^2})^2$$
$$\Leftrightarrow \exists z > 0 : \frac{z}{3\alpha} = (1 - e^{-z})^2 \;//\; z = 3\alpha x^2$$
$$\Leftrightarrow \exists z > 0 : \alpha = \frac{z}{3(1 - e^{-z})^2}$$
$$\Leftrightarrow \alpha \geq \min_{z>0} \frac{z}{3(1 - e^{-z})^2} =: c_3^\Delta \approx 0.81$$



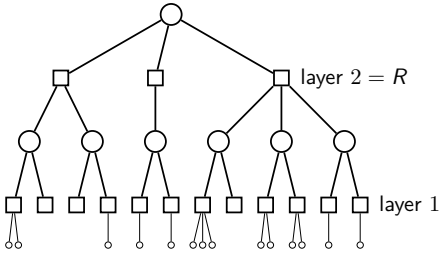
iii Interim Conclusion: What we learned about peeling T_α

Lemma

For $\alpha < c_3^\Delta \approx 0.81$ we have

- $\lim_{i\to\infty} p_i = 0$.
- $\lim_{R\to\infty} q_R = \lim_{R\to\infty} p_R^3 = 0$.

"Root rarely survives for large R."



iv What does it mean for T_α to be locally like $G_{m,\alpha m}$?

Neighbourhoods in T_α and G

Let $R \in \mathbb{N}$. We consider

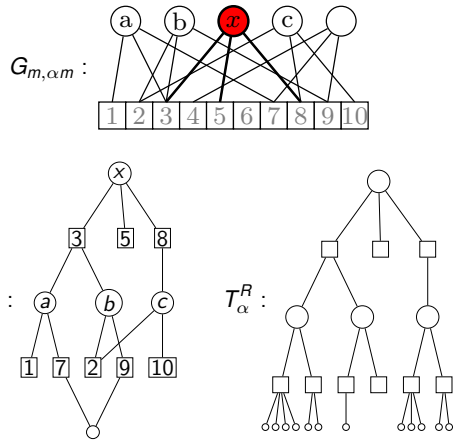
- T_α^R as before and
- for any fixed $x \in S$ the subgraph $G_{m,\alpha m}^{x,R}$ of $G_{m,\alpha m}$ induced by all nodes with distance at most $2R$ from x.

Lemma

For any $R \in \mathbb{N}$ and $m \to \infty$ we have

$$G_{m,\alpha m}^{x,R} \xrightarrow{d} T_\alpha^R$$

i.e. $\forall T : \lim_{m\to\infty} \Pr[G_{m,\alpha m}^{x,R} = T] = \Pr[T_\alpha^R = T]$.



iv Distribution of T_α^R

Lemma

Let T_y be a possible outcome of T_α^R given by a finite sequence $y = (y_1, \ldots, y_k) \in \mathbb{N}_0^k$ specifying the number of children of $\square$-nodes in level order. Then

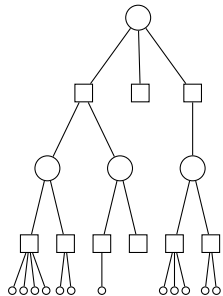
$$\Pr[T_\alpha^R = T_y] = \prod_{i=1}^{k} \Pr_{Y \sim \mathrm{Pois}(3\alpha)}[Y = y_i].$$

Remark on the meaning of "="

We consider rooted unlabeled graphs where neighbours of vertices are ordered.

e.g. for $y = (2, 0, 1, 4, 2, 1, 0, 3, 2)$:

$T_y =$



iv No cycles in $G^{x,R}_{m,\alpha m}$

Karlsruhe Institute of Technology

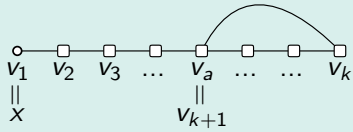
Lemma

Assume $R = \mathcal{O}(1)$. The probability that $G^{x,R}_{m,\alpha m}$ contains a cycle is $\mathcal{O}(1/m)$.

Proof.

If $G^{x,R}_{m,\alpha m}$ contains a cycle then we have

- a sequence $(v_1 = x, v_2, \ldots, v_k, v_{k+1} = v_a)$ of nodes with $a \in [k]$
- of length $k \leq 4R$ (consider BFS tree for x and additional edge in it)
- for each $i \in \{1, \ldots, k\}$ an index $j_i \in \{1, 2, 3\}$ of the hash function connecting v_i and v_{i+1}. (If $a = k - 1$ then $j_k \neq j_{k-1}$.)



$$\Pr[\exists \text{cycle in } G^{x,R}_{m,\alpha m}] \leq \Pr[\exists 2 \leq k \leq 4R : \exists v_2, \ldots, v_k : \exists a \in [k] : \exists j_1, \ldots, j_k \in [3] : \forall i \in [k] : h_{j_i} \text{ connects } v_i \text{ to } v_{i+1}]$$

$$\leq \sum_{k=2}^{4R} \sum_{v_2,\ldots,v_k} \sum_{a=1}^{k} \sum_{j_1,\ldots,j_k} \prod_{i=1}^{k} \Pr[h_{j_i} \text{ connects } v_i \text{ to } v_{i+1}] \leq \sum_{k=2}^{4R} (\max\{m, n\})^{k-1} \cdot k \cdot 3^k \left(\tfrac{1}{m}\right)^k = \tfrac{1}{m} \sum_{k=2}^{4R} k \cdot 3^k = \mathcal{O}(1/m). \quad \square$$

iv Distribution of $G^{x,R}_{m,\alpha m}$

Lemma

Let T_y be a possible outcome of T^R_α as before. Then

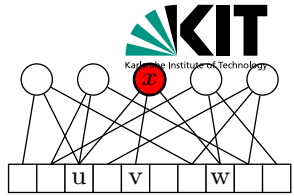
$$\Pr_{h_1,h_2,h_3\sim\mathcal{U}([m]^D)}[G^{x,R}_{m,\alpha m} = T_y] \overset{m\to\infty}{\longrightarrow} \prod_{i=1}^{k} \Pr_{Y\sim\mathrm{Pois}(3\alpha)}[Y = y_i].$$

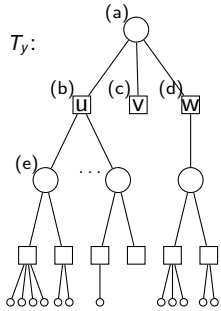
"Proof by example", using T_y shown on the right.

The following things have to "go right" for $G^{x,R}_{m,\alpha m} = T_y$.

a $h_1(x), h_2(x), h_3(x)$ pairwise distinct: probability $\overset{m\to\infty}{\longrightarrow} 1$
↪ non-distinct would give cycle of length 2. Unlikely by lemma.

Note: $3\lfloor\alpha m\rfloor - 3$ remaining hash values $\sim \mathcal{U}([m])$.





iv Distribution of $G^{x,R}_{m,\alpha m}$

Lemma

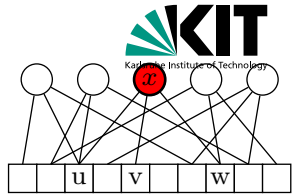
Let T_y be a possible outcome of T^R_α as before. Then

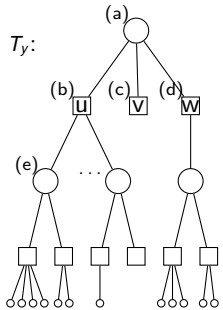
$$\Pr_{h_1,h_2,h_3\sim\mathcal{U}([m]^D)}[G^{x,R}_{m,\alpha m} = T_y] \xrightarrow{m\to\infty} \prod_{i=1}^{k} \Pr_{Y\sim\text{Pois}(3\alpha)}[Y = y_i].$$

"Proof by example", using T_y shown on the right.

b Exactly $y_1 = 2$ of the remaining hash values are u.
↪ $\Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor-3,\frac{1}{m})}[Y = 2] \xrightarrow{m\to\infty} \Pr_{Y\sim\text{Pois}(3\alpha)}[Y = 2]$. → exercise
Moreover: The two hash values must belong to 2 distinct keys. Probability $\xrightarrow{m\to\infty}$ 1.
↪ non-distinct would give cycle of length 2.

Note: The $3\lfloor\alpha m\rfloor - 5$ remaining hash values are $\sim \mathcal{U}([m] \setminus \{u\})$. → exercise





iv Distribution of $G^{x,R}_{m,\alpha m}$

Lemma

Let T_y be a possible outcome of T^R_α as before. Then

$$\Pr_{h_1,h_2,h_3\sim\mathcal{U}([m]^D)}[G^{x,R}_{m,\alpha m} = T_y] \xrightarrow{m\to\infty} \prod_{i=1}^{k} \Pr_{Y\sim\text{Pois}(3\alpha)}[Y = y_i].$$

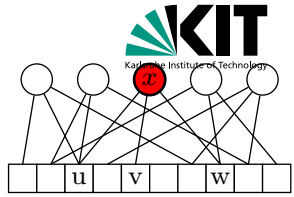
"Proof by example", using T_y shown on the right.

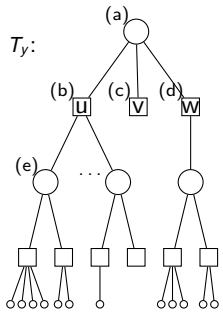
c None of the remaining hash values are v.
$\hookrightarrow \Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor-5,\frac{1}{m-1})}[Y=0] \xrightarrow{m\to\infty} \Pr_{Y\sim\text{Pois}(3\alpha)}[Y=0]$.

Note: The $3\lfloor\alpha m\rfloor - 5$ remaining hash values are $\sim \mathcal{U}([m]\setminus\{u,v\})$.

d One of the remaining hash values is w.
$\hookrightarrow \Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor-5,\frac{1}{m-2})}[Y=1] \xrightarrow{m\to\infty} \Pr_{Y\sim\text{Pois}(3\alpha)}[Y=1]$.

...





iv Distribution of $G^{x,R}_{m,\alpha m}$

Lemma

Let T_y be a possible outcome of T^R_α as before. Then

$$\Pr_{h_1,h_2,h_3\sim\mathcal{U}([m]^D)}[G^{x,R}_{m,\alpha m} = T_y] \overset{m\to\infty}{\longrightarrow} \prod_{i=1}^{k} \Pr_{Y\sim\mathrm{Pois}(3\alpha)}[Y = y_i].$$

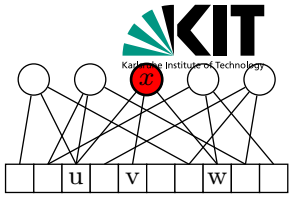
Proof sketch in general (some details ommitted)

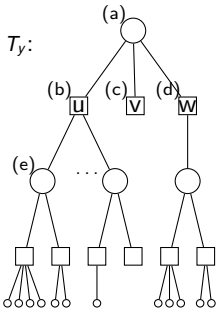
- General case at i-th □-node. Want: probability that $G^{x,R}_{m,\alpha m}$ continues to match T_y. Note: T_y is fixed, so i and the number c_i of previously revealed hash values is bounded.

$$\Pr_{Y\sim Bin(3\lfloor\alpha m\rfloor - c_i, \frac{1}{m-i+1})}[Y = y_i] \overset{m\to\infty}{\longrightarrow} \Pr_{Y\sim\mathrm{Pois}(3\alpha)}[Y = y_i].$$

  Moreover, those y_i hash values must belong to distinct fresh keys. Probability $\overset{m\to\infty}{\longrightarrow} 1$ ↪ otherwise we'd have a cycle.

- General case for ◯-node. The two children must be fresh: probability $\overset{m\to\infty}{\longrightarrow} 1$ ↪ otherwise there would be a cycle.





Probability that a specific key survives peeling

Lemma

Let $\alpha < c_3^{\Delta}$. Let x be any ◯-node in $G_{m,\alpha m}$ as before (chosen before sampling the hash functions). Let

$$\mu_m := \Pr_{h_1,h_2,h_3 \sim \mathcal{U}([m]^D)}[x \text{ is removed when peeling } G_{m,\alpha m}].$$

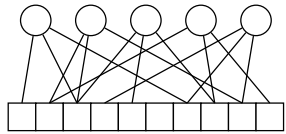
Then $\lim_{m\to\infty} \mu_m = 1$.

$\mu_m := \Pr[x \text{ is removed when peeling } G_{m,\alpha m}] \xrightarrow{m\to\infty} 1$

Let $\delta > 0$ be arbitrary. We will show $\lim_{m\to\infty} \mu_m \geq 1 - 2\delta$.
Let $R \in \mathbb{N}$ be such that $q_R < \delta$. — possible because $\lim_{R\to\infty} q_R = 0$
$\mathcal{Y}^R := \{\text{all possibilities for } T^R_\alpha\}$
$\mathcal{Y}^R_{\text{peel}} := \{T \in \mathcal{Y}^R \mid \text{peeling } T \text{ removes the root}\}$ — note: $\Pr[T^R_\alpha \notin \mathcal{Y}^R_{\text{peel}}] = q_R \leq \delta$.
Let $\mathcal{Y}^R_{\text{fin}} \subseteq \mathcal{Y}^R$ be a *finite* set such that $\Pr[T^R_\alpha \notin \mathcal{Y}^R_{\text{fin}}] \leq \delta$ — uses that $\mathcal{Y}^R$ is countable and $\sum_{T\in\mathcal{Y}^R} \Pr[T^R_\alpha = T] = 1$.

$$
\begin{aligned}
\lim_{m\to\infty} \mu_m &\geq \lim_{m\to\infty} \Pr[G^{x,R}_{m,\alpha m} \in \mathcal{Y}^R_{\text{peel}}] && \text{peeling only in } R\text{-neighbourhood of } x \text{ is “weaker”}\\
&\geq \lim_{m\to\infty} \Pr[G^{x,R}_{m,\alpha m} \in \mathcal{Y}^R_{\text{peel}} \cap \mathcal{Y}^R_{\text{fin}}]\\
&= \lim_{m\to\infty} \textstyle\sum_{T\in\mathcal{Y}^R_{\text{peel}}\cap\mathcal{Y}^R_{\text{fin}}} \Pr[G^{x,R}_{m,\alpha m} = T]\\
&= \textstyle\sum_{T\in\mathcal{Y}^R_{\text{peel}}\cap\mathcal{Y}^R_{\text{fin}}} \lim_{m\to\infty} \Pr[G^{x,R}_{m,\alpha m} = T] && \textit{finite} \text{ sums commute with limit}\\
&= \textstyle\sum_{T\in\mathcal{Y}^R_{\text{peel}}\cap\mathcal{Y}^R_{\text{fin}}} \Pr[T^R_\alpha = T] && \text{previous lemmas}\\
&= \Pr[T^R_\alpha \in \mathcal{Y}^R_{\text{peel}} \cap \mathcal{Y}^R_{\text{fin}}] = 1 - \Pr[T^R_\alpha \notin \mathcal{Y}^R_{\text{peel}} \cap \mathcal{Y}^R_{\text{fin}}]\\
&= 1 - \Pr[T^R_\alpha \notin \mathcal{Y}^R_{\text{peel}} \vee T^R_\alpha \notin \mathcal{Y}^R_{\text{fin}}] && \text{De Morgan's laws: } \overline{A\cap B} = \overline{A}\cup\overline{B}\\
&\geq 1 - \Pr[T^R_\alpha \notin \mathcal{Y}^R_{\text{peel}}] - \Pr[T^R_\alpha \notin \mathcal{Y}^R_{\text{fin}}] \geq 1 - 2\delta. && \text{union bound: } \Pr[E_1 \vee E_2] \leq \Pr[E_1] + \Pr[E_2] \quad \square
\end{aligned}
$$

vi Proof of the Peeling Theorem



Theorem

Let $\alpha < c_3^{\Delta}$. Then $\Pr[G_{m,\alpha m} \text{ is peelable}] = 1 - o(1)$.

Proof

Let $n = \lfloor \alpha m \rfloor$ and $0 \leq s \leq n$ the number of ◯ nodes surviving peeling.

last lemma:	each ◯ survives with probability $o(1)$.
linearity of expectation	$\mathbb{E}[s] = n \cdot o(1) = o(n)$.
Exercise:	$\Pr[s \in \{1, \ldots, \delta n\}] = \mathcal{O}(1/m)$ if $\delta > 0$ is a small enough constant.
Markov:	$\Pr[s > \delta n] \leq \frac{\mathbb{E}[s]}{\delta n} = \frac{o(n)}{\delta n} = o(1)$.
finally:	$\Pr[s > 0] = \Pr[s \in \{1, \ldots, \delta n\}] + \Pr[s > \delta n] = \mathcal{O}(1/m) + o(1) = o(1)$. □

Conclusion

Peeling Process

- greedy algorithm for placing keys in cuckoo table
- works up to a load factor of $c_3^{\Delta} \approx 0.81$

We saw glimpses of important techniques

- *Local interactions in large graphs.* Also used in statistical physics.
- *Galton-Watson Processes / Trees.* Random processes related to T_α.
- *Local weak convergence.* How the finite graph $G_{m,\alpha m}$ is locally like T_α.

But wait, there's more!

- Further applications of peeling
  - retrieval data structures (next lecture)
  - perfect hash functions (next lecture)
  - set sketches
  - linear error correcting codes

Anhang: Mögliche Prüfungsfragen I

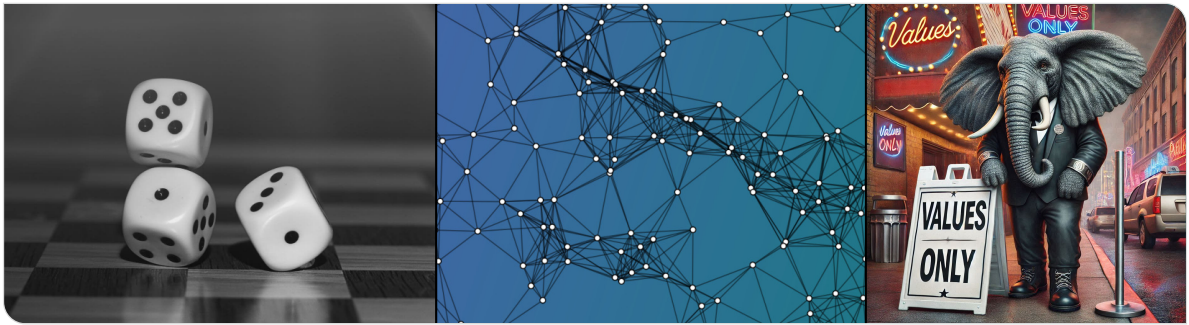
- Cuckoo Hashing und der Schälalgorithmus
  - (Wie) kann man Cuckoo Hashing mit mehr als 2 Hashfunktionen aufziehen?
  - Welcher Vorteil ergibt sich im Vergleich zu 2 Hashfunktionen?
  - Wie funktioniert der Schälalgorithmus zur Platzierung von Schlüsseln in einer Cuckoo Hashtabelle?
  - Schälen lässt sich als einfacher Prozess auf Graphen auffassen. Wie?
  - Was besagt das Hauptresultat, das wir zum Schälprozess bewiesen haben?
- Beweis des Schälsatzes. *Mir ist klar, dass der Beweis äußerst kompliziert ist.*
  - Im Beweis haben zwei Graphen eine Rolle gespielt ein endlicher und ein (potentiell) unendlicher. Wie waren diese Graphen definiert?
  - Welcher Zusammenhang besteht zwischen der Verteilung von T_α und der Verteilung von $G_{m,\alpha m}$?

Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Retrieval Data Structures

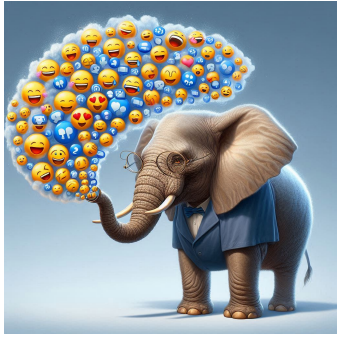
Stefan Walzer | WS 2024/2025



Evaluationsergebnisse

Zusammenfassung der Befragung ($n = 8$)

- Inhalt eher schwer… (3.8 / 5)
- … aber anschaulich (1.5 / 5)
- Aufwand eher hoch… (3.4 / 5)
- … aber sehr lehrreich (1.3 / 5)
- Aufwand Vorbereitung/Nachbereitung
  - 0h-1h ($\times$ 2)
  - 1h-2h ($\times$ 4)
  - 2h-3h ($\times$ 2)
- sehr geeignete Materialien (1.4 / 5)
- insgesamt noch sehr gut (Note 1.4)



Weitere Rückmeldungen oder Verbesserungsvorschläge?

Verbleibende Veranstaltungen & Prüfungstermine

Do	23.1.	reguläre Vorlesung Peeling
Di	28.1.	reguläre Übung
Do	30.1.	Vorlesung Retrieval mit Übung
Do	6.2.	Q&A, weitere Übungsaufgaben, könnte ausfallen
Di	11.2.	~~Übung fällt definitiv aus~~
Do	13.2.	*Gastvorlesung Hans-Peter Lehmann: Perfect Hashing*
	15.2.	*offizielles Ende der Vorlesungszeit*
Mi	26.2.	Prüfungstermine
Do	27.2.	Prüfungstermine
Fr	28.2.	Prüfungstermine
Mi	26.3.	Prüfungstermine
Do	27.3.	Prüfungstermine
Fr	28.3.	Prüfungstermine

Andere Prüfungstermine auf Anfrage vermutlich möglich.

Prüfung (mündlich, 20 Minuten)

- Anmeldung über das Sekretariat:
  - Anja Blancani (blancani@kit.edu)
  - cc an mich (stefan.walzer@kit.edu)
  - Bitte angeben:
    - Vollständiger Name
    - Matrikelnummer
    - Studienfach
    - Version der Prüfungsordnung
- Abmeldung auch über das Sekretariat
- Ort: Stefans Büro (50.34, Raum 209).
- Zeitslots jeweils:
  - 10:00, 10:25, 10:50, 11:15
  - 14:00, 14:25, 14:50, 15:15
- Inhalte von Vorlesung *und Übung*

Content

1. The Retrieval Problem
- Definition
- Motivation

2. Retrieval based on Fingerprinting

3. Cuckoo-Style Retrieval
- Using Peeling
- In General Using Linear Algebra
- Teaser: Ribbon Retrieval

4. Summary

The Retrieval Problem

The retrieval data type (for universe D, range $[k]$)

construct(f):
- input: function $f : S \to [k]$ // $f \subseteq D \times [k]$ where $S \subseteq D$ has size $n = |S|$
- output: data structure R.

eval$_R(x)$:
- input: $x \in D$
- output: some value in $[k]$
- requirement: **eval**$_R(x) = f(x)$ for all $x \in S$

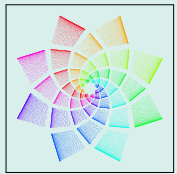
The price to pay

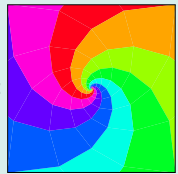
- R cannot be used to decide "is $x \in S$?"
- **eval**$_R(x)$ is *unspecified* if $x \notin S$.

Goals

- space requirement of R is $\mathcal{O}(n \log k)$ bits
  - possibly even $n\lceil \log_2(k) \rceil + o(n)$
  - ⚠ naively storing f needs $\Omega(n(\log(k) + \log(|D|)))$
- ideally running time of **eval**$_R$ is $\mathcal{O}(1)$
- ideally running time of **construct** is $\mathcal{O}(n)$

Intuition

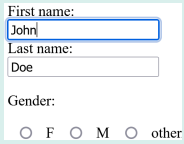
f : 

R : 

- R is a *continuation* of f
- information about the domain S is lost.

Motivation for Retrieval

Task: Predict gender based on first name



- want $\geq 90\%$ accuracy
- client side only
- lightweight

Have large data base:

Annotated list of 10000 most common first names.

$$f : \{\text{Dave} \mapsto M, \text{Joanna} \mapsto F, \text{Christina} \mapsto F, \ldots\}$$

$\approx$ 10 bytes per name, too large to send to client.

Solution using retrieval

- send $R = \textbf{construct}(f)$ to client
  $\hookrightarrow \approx 1$ bit per name
- prefill gender with $\textbf{eval}_R(\text{firstName})$

Weaknesses:

May guess incorrectly if

- name is ambiguous (“Kim”, “Chris”)
- user is *other* / prefers not to say
- name not listed in f (e.g. “Crhistina”, “Inghean”)
  $\hookrightarrow$ would be better to *refrain from guessing*

Static Filters / AMQs

Exercise: Filters from Retrieval

Good retrieval data structures yield good static filters.

Content

1. The Retrieval Problem
   - Definition
   - Motivation

2. Retrieval based on Fingerprinting

3. Cuckoo-Style Retrieval
   - Using Peeling
   - In General Using Linear Algebra
   - Teaser: Ribbon Retrieval

4. Summary

Fingerprint-Based Retrieval

x	$\text{fp}(x)$	$f(x)$	$\text{part}(x)$
Inception	I	👍	3
Life of Brian	L	👎	3
Avatar	A	😍	2
Titanic	T	👎	3
The Matrix	T	😍	1
Gladiator	G	😍	1
Forrest Gump	F	👍	1
Interstellar	I	👎	2
The Godfather	T	👍	2
Jurassic Park	J	👍	3
The Lion King	T	😍	1
Frozen	F	👍	3

Idea 0: Use dictionaries after all

Idea 1: Store fingerprint-value pairs instead of key-value pairs

- Sample fingerprint hash function $\text{fp} \sim \mathcal{U}([\ell]^D)$ for small ℓ.

Idea 2: Partition into b small buckets (Example: $b = 3$)

- Sample partition hash function $\text{part} \sim \mathcal{U}([b]^D)$.
- Use many dictionaries $\text{dict}_1, \ldots, \text{dict}_b$ of small capacity c (example: $c = 4$).
- Store $\text{fp}(x) \mapsto f(x)$ in $\text{dict}_{\text{part}(x)}$.

Idea 3: *Bump* inconvenient keys to fallback data structure

Recursively constructed, for small fraction of keys.

problem	**how to address?**
$(x, f(x))$ too large	fingerprinting: $(\text{fp}(x), f(x))$ is small
fingerprint collisions	reduced by partitioning, solved by bumping colliding keys
dict_i **might overflow**	bump excess keys

In expectation: construction time $\mathcal{O}(n)$, query time $\mathcal{O}(1)$, space $\mathcal{O}(\log_2 k)$ bits per key (Müller et al, 2014).

Data Structure:

bucket 1	F	G	T	-	👍	😍	😍	-
bucket 2	A	I	-	-	😍	👎	-	-
bucket 3	I	J	L	T	👍	👍	👎	👎

Algorithm $\text{eval}(x)$**:**

$i \leftarrow \text{part}(x)$
if $\text{dict}_i.\textit{contains}(\text{fp}(x))$ **then**
  return $\text{dict}_i[\text{fp}(x)]$
else
  return $\text{fallback}.\text{eval}(x)$

Content

1. The Retrieval Problem
   - Definition
   - Motivation

2. Retrieval based on Fingerprinting

3. Cuckoo-Style Retrieval
   - Using Peeling
   - In General Using Linear Algebra
   - Teaser: Ribbon Retrieval

4. Summary

Cuckoo-Style Retrieval using Peeling for $f : S \to \{0, \ldots, k-1\}$

Retrieval Data Structure $R = (h_1, h_2, h_3, A)$

- $A \in \{0, \ldots, k-1\}^m$ is array of cleverly chosen values
- $m = \frac{n}{0.81} = 1.23n$ //0.81 is peeling threshold $c_3^\triangle$
- $h_1, h_2, h_3 \sim \mathcal{U}([m]^D)$ //SUHA
- $\mathbf{eval}_R(x) := (A[h_1(x)] + A[h_2(x)] + A[h_3(x)]) \bmod k$

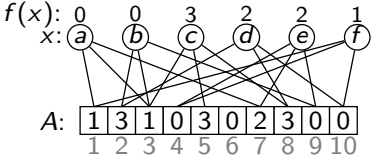
Performance

- space $1.23n\lceil \log_2(k) \rceil$ bits
- construct in $\mathcal{O}(n)$
- eval in $\mathcal{O}(1)$

How does **construct**(f) choose A?

If $A[j]$ is only used by x_i then setting $A[j]$ *in the end* takes care of x_i without affecting other keys.

- ↪ can forget about x_i "for now" and focus on the rest
- ↪ if configuration is peelable, this takes care of all keys



Equations (mod k for $k = 4$)

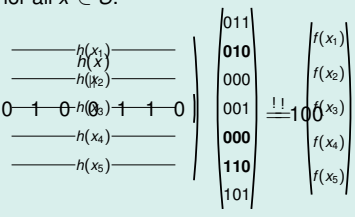
c: $A[5] := 3 - A[3] - A[8]$
d: $A[8] := 2 - A[2] - A[10]$
b: $A[2] := 0 - A[3] - A[9]$
a: $A[3] := 0 - A[1] - A[7]$
e: $A[7] := 2 - A[4] - A[9]$
f: $A[1] := 1 - A[4] - A[10]$

Cuckoo-Style Retrieval using Linear Algebra over the field $\mathbb{F}_2 = \{0, 1\}$

Cuckoo-style retrieval for $f : S \to \mathbb{F}_2^r$ with $|S| = n$

Pick $m \geq n$. Data structure is pair $R = (h : D \to \mathbb{F}_2^m, \vec{z} \in \mathbb{F}_2^{m \times r})$ such that $h(x)^T \cdot \vec{z} = f(x)$ for all $x \in S$.

$r = 3$
$m = 7$
$n = 5$



Goals when Choosing h

i. **success probability**: rows of matrix $(h(x))_{x \in S}$ must be linearly independent
ii. **construction time**: linear system should be easy to solve
iii. **query time**: products $h(x) \cdot z$ should be fast to compute
iv. **space**: $\alpha = \frac{n}{m}$ should be close to 1

What the peeling-based approach achieves

i. $1 - \mathcal{O}(1/m)$ (success iff peelable)
ii. $\mathcal{O}(n)$ (time to run peeling)
iii. $\mathcal{O}(1)$ (three memory accesses two $\oplus$-additions)
iv. $\alpha = 0.81$ (peeling threshold)

What more could we hope for?

i. -
ii. better cache efficiency
iii. better cache efficiency
iv. $\alpha = 1 - o(1)$

Teaser: “Ribbon Retrieval”

Standard Ribbon Retrieval

Idea: $\boldsymbol{h}(\boldsymbol{x})$ = “randomly placed block of $\mathcal{O}(\frac{\log n}{1-\alpha})$ random bits”

$$\begin{pmatrix}
 & & 0 & 1 & 1 & 0 & 0 & 1 & & & & & & \\
1 & 0 & 1 & 0 & 0 & 1 & & & & & & & & \\
 & & & & & 0 & 1 & 0 & 1 & 1 & 0 & & & \\
 & & & & 1 & 0 & 0 & 0 & 1 & 0 & & & & \\
 & 0 & 1 & 1 & 0 & 1 & 1 & & & & & & & \\
 & & & & & & & & 0 & 1 & 0 & 1 & 0 & 1 \\
 & & 0 & 1 & 0 & 1 & 1 & 1 & & & & & & \\
 & & & & & 1 & 1 & 1 & 0 & 0 & 0 & & & \\
\end{pmatrix}
\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \\ z_9 \\ z_{10} \\ z_{11} \\ z_{12} \\ z_{13} \\ z_{14} \\ z_{15} \end{pmatrix}
\overset{!}{=}
\begin{pmatrix} 010 \\ 011 \\ 101 \\ 111 \\ 101 \\ 000 \\ 010 \\ 100 \end{pmatrix}$$

$\underbrace{\qquad\qquad}_{\frac{\log n}{1-\alpha}}$

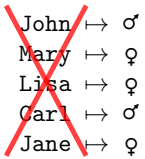
Bumped Ribbon Retrieval

“Bumping” allows reducing the necessary width of the blocks to $\log n$. End result after a lot of effort[a]:

- construction $\mathcal{O}(n \log n)$
  // cache efficient
- query $\mathcal{O}(r)$ // cache efficient
- bits per key: $r + \mathcal{O}(\frac{\log \log n}{\log^2 n})$.

[a] *Fast Succinct Retrieval and Approximate Membership using Ribbon*, Peter C. Dillinger, Lorenz Hübschle-Schneider, Peter Sanders, Stefan Walzer, 2021.

Summary



Retrieval

- constructed for $f : S \to [k]$
- goal: $\approx \log_2(k)$ bits per key
- does not store S

$$\textbf{eval}_R(x) = \begin{cases} f(x) & \text{if } x \in S \\ \textit{unspecified} & \text{if } x \notin S \end{cases}$$

- insert & delete not supported

Dictionary / Hash Table

- constructed for $f : S \to [k]$ // $S \subseteq D$
- goal: $\approx \log_2(D) + \log_2(k)$ bits per key
- stores S

$$\textbf{lookup}_R(x) = \begin{cases} f(x) & \text{if } x \in S \\ \perp & \text{if } x \notin S \end{cases}$$

- insert & delete usually supported

Constructions discussed here

- fingerprint-based approaches
- cuckoo-style retrieval using peeling
- cuckoo-style retrieval using linear algebra with $\mathbb{F}_2$

Remark: There is more...

- compressed retrieval data structures
- learned retrieval data structures
- active research @ITI Sanders!

Anhang: Mögliche Prüfungsfragen

- Was ist der Funktionsumfang einer Retrieval Datenstruktur?
- Was sind die Vorteile und Nachteile im Vergleich zu einer normalen Hashtabelle?
- Welche Anwendungen für Retrieval Datenstrukturen haben wir kennengerlernt?
- Zu Retrieval Datenstruktur mit dem Fingerprint-Ansatz:
  - Wie ist die Datenstruktur aufgebaut? Wie funktioniert der Query Algorithmus?
  - Was ist „Bumping“ und wieso brauchen wir es?
  - Was sind Konstruktions- und Zugriffszeiten sowie Speicherverbrauch?
- Zu Retrieval Datenstruktur basierend auf dem Schälalgorithmus:
  - Wie ist die Datenstruktur aufgebaut? Wie funktioniert der Query Algorithmus?
  - Was sind Konstruktions- und Zugriffszeiten sowie Speicherverbrauch?
- Zu Retrieval Datenstruktur basierend auf linearer Algebra mit dem Körper $\mathbb{F}_2$:
  - Was ist das allgemeine Schema? Inwiefern passt auch der Ansatz mit dem Schälalgorithmus in dieses Schema?
  - Welche Ziele sollte ich bei der Wahl der Funktion h im Auge behalten?
- Ribbon Retrieval ist nicht prüfungsrelevant.

Contents

1. Basic Notions and Notation
2. The Power of Randomness
3. Important Random Variables and How to Sample Them
4. Randomised Complexity Classes
5. Probability Amplification
6. Concentration
7. Classic Hash Tables
8. Bloom Filters
9. Coupling, Balls into Bins, Poissonisation and the Poisson Point Process
10. Approximation Algorithms
11. Streaming
12. Game Theory and Yao's Principle
13. Probabilistic Method
14. Random Graphs
15. Cuckoo Hashing
16. The Peeling Algorithm
17. Retrieval