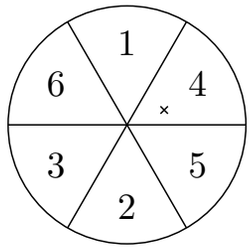
Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 1 – Basic Concepts

## Probability and Computing

### Exercise 1 – I probably still know it ...

A darts player throws a dart at a dartboard. The board is divided into 6 equally sized segments, each assigned a different score from $\{1, \ldots, 6\}$. Since the player is still in training, the dart lands uniformly at random on the board (but never off target). In the example on the right, the throw scored 4 points.




(a) Model the random experiment of the throw (not the resulting score) by describing the sample space and a probability measure of a continous probability space.

(b) We are now particularly interested in the following properties of a throw:

- the distance of the dart tip from the center of the board
- the resulting score
- the resulting score is even
- the resulting score modulo 2

Define the corresponding random variables and events.

(c) Determine the cumulative distribution function of the distance from part (b).

In the following, we are no longer interested in the position of the dart tip, but only in the resulting score.

(d) Model this random experiment by specifying a suitable *discrete* probability space.

(e) Let $X$ be the random variable representing the score. Determine the following values:

- The expectation of $X$
- The variance of $X$
- The expectation of $X \cdot \mathbb{1}_{X \text{ is odd}}$. This is the expectation of the score in a game variant where only odd numbers count.
- The expectation of a throw that produced an odd score.

## Solution 1

(a) The sample space is the unit disk:

$$\Omega = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}.$$

The probability distribution is the uniform distribution over $\Omega$. In other words, the probability measure assigns to each subset of $\Omega$ with area $A$ the probability $A/\pi$. What a probability measure formally is, is not part of this lecture.

(b) We are asked about the following things.

- The *random variable* $D$ that describes the distance. It holds that $D((x, y)) = \sqrt{x^2 + y^2}$ for $(x, y) \in \Omega$.
- The *random variable* $P$ that describes the score. If we number the segments of the dartboard in mathematically positive order as $A_1, A_2, \ldots, A_6 \subseteq \Omega$, then:

$$P(\omega) = \begin{cases} 4 & \text{if } \omega \in A_1 \\ 1 & \text{if } \omega \in A_2 \\ 6 & \text{if } \omega \in A_3 \\ 3 & \text{if } \omega \in A_4 \\ 2 & \text{if } \omega \in A_5 \\ 5 & \text{if } \omega \in A_6. \end{cases}$$

- The event $G$, that the score is even, can now be written in various equivalent ways. Here are three suggestions:

$$G = \{\omega \in \Omega \mid P(w) \in \{2, 4, 6\}\} = \{P \in \{2, 4, 6\}\} = A_1 \cup A_3 \cup A_5.$$

Note: In the middle case, the curly braces do not have their usual set notation meaning, but indicate that an event is being defined.

- The random variable $G(\omega) = P(\omega) \bmod 2$ is special in that it can only take the values 0 and 1. Such random variables are called indicator random variables. It holds that $\mathbb{E}[G] = \Pr[G = 1]$.

(c) For $d \leq 0$, $\Pr[D \leq d] = 0$. For $d \geq 1$, $\Pr[D \leq d] = 1$. For $0 \leq d \leq 1$, consider the event $E_d = \{(x, y) \in \Omega \mid x^2 + y^2 \leq d^2\}$. This has an area of $d^2\pi$. Thus:

$$\Pr[D \leq d] = \Pr[E_d] = d^2\pi/\pi = d^2.$$

(d) $\Omega = \{1, 2, 3, 4, 5, 6\}$ with uniform distribution.

(e) The following quantities were asked for:

- $\mathbb{E}[X] = (1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5$.
- $\mathrm{Var}(X) = ((1-3.5)^2 + (2-3.5)^2 + (3-3.5)^2 + (4-3.5)^2 + (5-3.5)^2 + (6-3.5)^2)/6 \approx 2.92$.
- $\mathbb{E}[X \cdot \mathbb{1}_{X \text{ is odd}}] = (1 + 0 + 3 + 0 + 5 + 0)/6 = 1.5$.
- $\mathbb{E}[X \mid X \text{ is odd}] = (1 + 3 + 5)/3 = 3$.

## Exercise 2 – Analogies to the calculation rules

Let $\Omega$ be the set of inhabitants of the distant land Omegon. Consider the following four statements and identify which of the five calculation rules from the slides each one is analogous to. For the remaining rule, come up with your own analogy. Argue (formally or intuitively, as you wish) why the calculation rules hold.

1. Let $h$ be the proportion of dog owners, $k$ the proportion of cat owners, and $t$ the proportion of inhabitants with a dog or a cat. Then: $t \leq h + k$.
2. Suppose 40% of the inhabitants live in the west, the rest in the east. If $g_1$ is the average height of westerners and $g_2$ the average height of easterners, then $g_1 \cdot 0.4 + g_2 \cdot 0.6$ is the average height in Omegon.
3. Suppose 40% of the inhabitants live in the west, the rest in the east. Let $k_1$ be the proportion of cat owners among westerners and $k_2$ the proportion of cat owners among easterners. Then the total proportion of cat owners is $k = k_1 \cdot 0.4 + k_2 \cdot 0.6$.
4. If an inhabitant eats on average $w$ white and $b$ brown chicken eggs per year, then on average they eat $w + b$ chicken eggs per year.

## Solution 2

1. Union bound. To show it, we need that for disjoint events $A$ and $B$, $\Pr[A \cup B] = \Pr[A] + \Pr[B]$ and that probabilities are non-negative. For two events $E_1, E_2$ it follows that:

$$\begin{aligned}\Pr[E_1 \cup E_2] &= \Pr[E_1 \cup (E_2 \setminus E_1)] = \Pr[E_1] + \Pr[E_2 \setminus E_1] \\ &\leq \Pr[E_1] + \Pr[E_2 \setminus E_1] + \Pr[E_1 \cap E_2] \\ &= \Pr[E_1] + \Pr[(E_2 \setminus E_1) \cup (E_1 \cap E_2)] = \Pr[E_1] + \Pr[E_2].\end{aligned}$$

   For more than two events, one can use induction.

2. Law of total expectation. Here is an informal proof:

$$\begin{aligned}&\sum_{i=1}^{n} \mathbb{E}[X \mid E_i] \cdot \Pr[E_i] \\ &\quad= \sum_{i=1}^{n} \sum_{x} x \cdot \Pr[X = x \mid E_i] \cdot \Pr[E_i] && \text{(Definition of conditional expectation)} \\ &\quad= \sum_{i=1}^{n} \sum_{x} x \cdot \Pr[\{X = x\} \cap E_i] && \text{(Definition of conditional probability)} \\ &\quad= \sum_{x} x \cdot \sum_{i=1}^{n} \Pr[\{X = x\} \cap E_i] = \sum_{x} x \cdot \Pr[X = x] && \text{(Disjointness of } E_1, \ldots, E_n\text{)} \\ &\quad= \mathbb{E}[X] && \text{(Definition of expectation)}\end{aligned}$$

3. Law of total probability. This is a special case of the law of total expectation for an indicator random variable $X = \mathbb{1}_F$, since then $\Pr[F] = \mathbb{E}[X]$ and $\Pr[F \mid E_i] = \mathbb{E}[X \mid E_i]$.

4. Linearity of expectation. The statement is hardly clearer with a proof, but here is one anyway. For discrete probability spaces, by expanding over all possible outcomes:

$$\begin{aligned}\mathbb{E}[X+Y] &= \sum_{\omega\in\Omega} (X(\omega)+Y(\omega)) \cdot \Pr[\{\omega\}] \\ &= \sum_{\omega\in\Omega} X(\omega)\Pr[\{\omega\}] + \sum_{\omega\in\Omega} Y(\omega)\Pr[\{\omega\}] = \mathbb{E}[X]+\mathbb{E}[Y].\end{aligned}$$

5. Missing: Tail Sum Formula. Analogy:

   > Every inhabitant $\omega$ can do a certain number $\ell_\omega$ of push-ups. Let $z_j$ be the number of inhabitants who can do at least $j$ push-ups. Then $\sum_{\omega\in\Omega} \ell_\omega = \sum_{j\geq 1} z_j$, and thus also $\frac{1}{|\Omega|}\sum_{\omega\in\Omega} \ell_\omega = \sum_{j\geq 1} \frac{z_j}{|\Omega|}$. The left sum is the average number of push-ups, and $\frac{z_j}{|\Omega|}$ is the proportion of inhabitants who can do $j$ push-ups.

   The formula can be derived by swapping summations. For arbitrary non-negative real numbers $(x_{i,j})_{i,j\in\mathbb{N}}$ it holds that $\sum_{j\geq 1}\sum_{i=1}^{j} x_{i,j} = \sum_{i\geq 1}\sum_{j\geq i} x_{i,j}$, since in both cases $x_{i,j}$ is counted exactly when $j \geq i$. We apply this to $x_{i,j} = \Pr[X = i]$.

$$\begin{aligned}\mathbb{E}[X] &= \sum_{j\geq 1} j \cdot \Pr[X=j] = \sum_{j\geq 1}\sum_{i=1}^{j} \Pr[X=j] \\ &= \sum_{i\geq 1}\sum_{j\geq i} \Pr[X=j] = \sum_{i\geq 1} \Pr[X\geq i].\end{aligned}$$

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 2 – The Power of Randomness

## Probability and Computing

## Exercise 1 – Checking Polynomial Equations

Let $\mathbb{F}$ be a field, for example $\mathbb{F} = \mathbb{Q}$. Given is a polynomial equation, for example:

$$(x^3 + 2x^2 - 5x - 6)(x^2 + x - 20)(x - 6) \stackrel{?}{=} x^6 - 7x^3 + 25$$

(a) Argue: In the case that the example equation does *not* hold, there are at most 6 values of $x$ for which both sides yield the same result.

(b) Describe a randomized algorithm that decides whether a polynomial equation holds or not. This algorithm may accept false polynomial equations as correct with a small probability. What can be said about this probability?

## Solution 1

(a) The equation has the form $f(x) = g(x)$ and can be rewritten as $f(x) - g(x) = 0$. If the equation does not hold, then $f(x) - g(x)$ is a polynomial of degree $0 \leq d \leq 6$. Such a polynomial has at most 6 roots. Therefore, there are at most 6 values of $x$ for which $f(x) - g(x) = 0$, and hence $f(x) = g(x)$.

(b) First, consider the case $|\mathbb{F}| < \infty$. We then choose $X \sim \mathcal{U}(\mathbb{F})$ uniformly at random and substitute $X$ into the equation. If the polynomial equation holds universally, it also holds for this $X$. If it does not hold, then as in (a) we can consider the maximum degree $d$ of both sides. There are at most $d$ elements of $\mathbb{F}$ for which both sides give the same result, and the probability that we picked one of them is at most $d/|\mathbb{F}|$.

If $|\mathbb{F}| = \infty$, there is no uniform distribution over $\mathbb{F}$. This is somewhat inconvenient, but we can choose $X$ uniformly at random from a large finite subset $S \subseteq \mathbb{F}$. Then the upper bound on the error probability $d/|S|$ can be made arbitrarily small.

**Remark:** This is a false-biased Monte Carlo algorithm (see the lecture on Probability Amplification).

## Exercise 2 – Checking Matrix Products[1]

Let $\mathbb{F}$ be a field, and $n \in \mathbb{N}$.

(a) Show: If $C, C' \in \mathbb{F}^{n \times n}$ are two different matrices and $v \in \{0,1\}^n$ is chosen uniformly at random, then $\Pr[C \cdot v \neq C' \cdot v] \geq \frac{1}{2}$.

(b) Describe an algorithm that, given $A, B, C \in \mathbb{F}^{n \times n}$, outputs a bit $X$ with $X = 1$ if $A \cdot B = C$ and $\Pr[X = 1] \leq 1/2$ if $A \cdot B \neq C$. The algorithm should perform only $O(n^2)$ field operations.

## Solution 2

(a) Let $C_1, \ldots, C_n \in \mathbb{F}^n$ and $C'_1, \ldots, C'_n \in \mathbb{F}^n$ be the columns of $C$ and $C'$ respectively, and let $v_1, \ldots, v_n \in \{0,1\}$ be the entries of $v$. Furthermore, let $i \in [n]$ be an index with $C_i \neq C'_i$, which must exist by assumption. We can then write:

$$D := C \cdot v - C' \cdot v = \underbrace{\sum_{\substack{j=1 \\ j \neq i}}^{n} (C_j - C'_j) \cdot v_j}_{w} + (C_i - C'_i) \cdot v_i.$$

Imagine that all entries of $v$ except the $i$-th have already been chosen, and only $v_i$ remains random. There are two cases:

**Case 1.** $w = 0$. In this case, $v_i = 1$ leads to $D = C_i - C'_i \neq 0^n$.

**Case 2.** $w \neq 0$. In this case, $v_i = 0$ leads to $D = w \neq 0^n$.

In both cases, at least one of the two possibilities for $v_i$ results in $D \neq 0^n$. Hence overall, $\Pr[C \cdot v \neq C' \cdot v] = \Pr[D \neq 0^n] \geq \frac{1}{2}$.

(b) We use (a) with $C' = A \cdot B$.

sample $v \leftarrow \mathcal{U}(\{0,1\}^n)$
$w_1 \leftarrow C \cdot v$
$w_2 \leftarrow A \cdot B \cdot v$ // Computation order: $A \cdot (B \cdot v)$
**return** $X = \mathbb{1}_{w_1 = w_2}$

Clearly, $X = 1$ is guaranteed if $A \cdot B = C$. If $A \cdot B \neq C$, then by (a) $\Pr[X = 1] = \Pr[C \cdot v = C' \cdot v] \leq \frac{1}{2}$.

The three matrix–vector products can each be computed in $O(n^2)$ field operations. In particular, this is faster than a full matrix–matrix multiplication, which (with a naive algorithm) requires $\Omega(n^3)$ field operations.

**Remark:** This is a false-biased Monte Carlo algorithm (see the lecture on Probability Amplification).

[1] Known as Freivalds' Algorithm.

## Exercise 3 – Deterministic Evaluation of $\bar{\wedge}$-Trees

Let $A$ be a deterministic algorithm that takes as input a bit vector $I \in \{0,1\}^n$ (with $n = 2^d$) and computes the value of the complete balanced $\bar{\wedge}$-tree whose leaves are labeled according to $I$. Show: There exists an input $I_A \in \{0,1\}^n$ such that $A$ must inspect every leaf label.

## Solution 3

We act as an adversary to the algorithm and assign the value of a leaf only when it is queried. We ensure that it must query all leaves. The resulting leaf labeling is then the worst-case input $I_A$ for $A$.

We prove the statement by induction. For $d = 1$ (leaf = root), there is nothing to show — the algorithm must obviously query the only leaf.

For $d > 1$, there are two subtrees of depth $d - 1$. In both, we use the strategy given by the induction hypothesis. Thus, the outcome of a subtree can only be determined once all $2^{d-1}$ leaves have been queried. Since the last leaf affects the result of the subtree, we can even choose its value at the end so that the entire subtree evaluates to 1. Hence, the total result at the root is $1 \bar{\wedge} b$, where $b$ is the result of the other subtree. The algorithm therefore also needs to determine $b$, and thus must query all leaves in the second subtree as well.

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 3 – Important Random Variables and How to Sample Them

## Probability and Computing

## Exercise 1 – Ber(1/3) from Ber(1/2)

Design an algorithm that, given a sequence $B_1, B_2, \ldots \sim \text{Ber}(1/2)$ of random bits, computes a sample $B \sim \text{Ber}(1/3)$ in expected time $O(1)$.

## Solution 1

We interpret $B_1, B_2, B_3, \ldots$ as the binary expansion of a number $U = (0.B_1B_2B_3\ldots)_2$. Then $U \sim \mathcal{U}([0,1])$. We define $B := \mathbb{1}_{U<1/3}$. This immediately implies $B \sim \text{Ber}(1/3)$ as desired. The binary expansion of $1/3$ is $1/3 = (0.01010101\ldots)_2$. Thus, the following algorithm results, which always takes the next two digits of $U$'s binary representation and checks whether they allow a decision:

**for** $i = 1$ **to** $\infty$ **do**
  $(x, y) \leftarrow (B_{2i-1}, B_{2i})$
  **if** $(x, y) = (0, 0)$ **then**
    **return** 1
  **else if** $(x, y) = (1, 0)$ *or* $(x, y) = (1, 1)$ **then**
    **return** 0

Each round leads to a decision with probability $3/4$. If $R$ is the number of rounds, then

$$\mathbb{E}[R] = \sum_{i \in \mathbb{N}_0} \Pr[R > i] = \sum_{i \in \mathbb{N}_0} \frac{1}{4^i} = \frac{1}{1 - \frac{1}{4}} = \frac{4}{3}.$$

**Remark:** In practice, one wouldn't actually do it this way. Instead, as in the next exercise, one assumes that one can directly sample $U \sim \mathcal{U}([0,1])$ (as accurately as floating-point numbers allow).

**Remark:** The runtime is unbounded — and this is unavoidable. We can show this by contradiction. Suppose there exists an algorithm that always terminates after reading only a fixed prefix $B_1, \ldots, B_C$ of the random bit sequence for some $C \in \mathbb{N}_0$. Then its output $B$ is a random variable $B : \Omega \to \{0,1\}$ on the probability space $\Omega = \{0,1\}^C$ (with uniform distribution). Each outcome has probability $2^{-C}$. Hence, any event (and thus also the event $\{B = 1\}$) must have probability that is an integer multiple of $2^{-C}$. This contradicts the requirement that $\Pr[B = 1] = 1/3$.

## Exercise 2 – $\mathrm{Ber}(p)$ and $\mathcal{U}(\{1,\dots,n\})$ from $\mathcal{U}([0,1])$

We now assume a machine model that can handle real numbers and allows us to sample $U \sim \mathcal{U}([0,1])$. Show that we can also sample $B \sim \mathrm{Ber}(p)$ for $p \in [0,1]$ and $X \sim \mathcal{U}(\{1,\dots,n\})$ for $n \in \mathbb{N}$.

**Hint:** For the rest of this sheet and the course, we take this result as given.

## Solution 2

Given $U \sim \mathcal{U}([0,1])$, define $B := \mathbb{1}_{U<p}$ and $X := \lceil U \cdot n \rceil$. Then indeed:

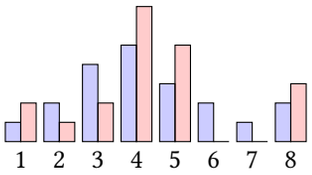
$$\Pr[B = 1] = \Pr[U < p] = p, \text{ and}$$
$$\text{for } 1 \le i \le n\text{: } \Pr[X = i] = \Pr\left[U \cdot n \in (i-1, i]\right] = \Pr\left[U \in (\tfrac{i-1}{n}, \tfrac{i}{n}]\right] = \tfrac{1}{n}.$$

**Remark:** Strictly speaking, since $U \sim \mathcal{U}([0,1])$, the value $U = 0$ is possible, which would yield $X = 0$, even though we want $X \in \{1,\dots,n\}$. However, this happens with probability 0. One can fix this by defining that 0 rounds up to 1, or simply ignore such minor edge cases.

## Exercise 3 – Rejection Sampling in General

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be distributions over a finite set $D$. Assume:

1. We can sample $X \sim \mathcal{D}_1$ in time $O(1)$.
2. For any $x \in D$, $p_1(x) := \Pr_{X\sim\mathcal{D}_2}[X = x]$ as well as $p_2(x) := \Pr_{X\sim\mathcal{D}_1}[X = x]$ can be computed in $O(1)$.
3. There exists $C > 0$ such that for all $x \in D$,
$$p_2(x) \le C \cdot p_1(x).$$



Possible histogram for $\mathcal{D}_1$ (blue, left) and $\mathcal{D}_2$ (red, right). It always holds that "red $\le 2 \cdot$ blue", so condition (3) holds with $C = 2$.

Design an algorithm that samples $Y \sim \mathcal{D}_2$ in expected time $O(C)$.

## Solution 3

The algorithm works as follows:

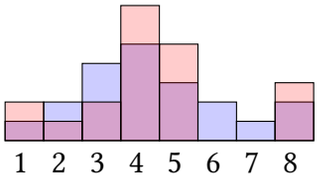
**while** True **do**
  sample $X \sim \mathcal{D}_1$ // $O(1)$
  sample $U \sim \mathcal{U}([0,1])$ // $O(1)$
  **if** $U < \frac{p_2(X)}{C \cdot p_1(X)}$ **then** // $O(1)$
    **return** $X$

To verify correctness, note that $\frac{p_2(X)}{C \cdot p_1(X)} \in [0,1]$ by assumption (3). Let $Y$ be the outcome of a single iteration: $Y = X$ if $X$ is accepted, and $Y = \perp$ otherwise. Then, for $x \in D$:

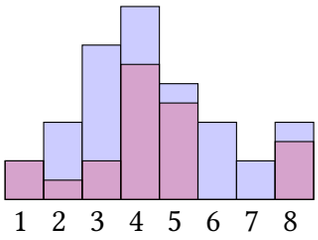
$$\Pr[Y = x] = \Pr[X = x] \cdot \Pr\left[U < \frac{p_2(x)}{C \cdot p_1(x)}\right] = p_1(x) \cdot \frac{p_2(x)}{C \cdot p_1(x)} = \frac{p_2(x)}{C}.$$

Thus, $\Pr[Y = x]$ is *proportional* to $p_2(x)$, and so $\Pr[Y = x \mid Y \neq \perp] = p_2(x)$. In other words: whenever a sample is returned, it follows the distribution $\mathcal{D}_2$ as desired. The success probability per iteration is $\Pr[Y \neq \perp] = \sum_{x \in D} \Pr[Y = x] = 1/C$. Hence, the expected number of rounds until success is $C$.

**Intuition:** You can visualize this process. If we draw the histograms of the two distributions on top of each other:




If we scale up the blue bars by a factor of $C$, the red bars are always below the blue ones.




To sample from the red distribution, it suffices to draw a random red point and return the index of the bar in which it lies. To achieve this, we draw a random blue point (in the illustration: a point that is blue or purple) and keep it if it is red.

In the algorithm, a random blue point is drawn by first choosing a bar $X$, and then selecting a random height $U \cdot C \cdot p_1(X)$ along that bar. This height is then compared with the height of the corresponding red bar.

## Exercise 4 – $G \sim \mathrm{Geom}_1(p)$ with Inverse Transform Sampling

Design an algorithm that, for a given $p \in (0, 1]$, samples a random variable $G \sim \mathrm{Geom}_1(p)$ in time $O(1)$.

## Solution 4

The cumulative distribution function of $G$ is:

$$F_G(i) = \Pr[G \leq i] = 1 - (1-p)^i.$$

For the (generalized) inverse, it follows for $u \in (0, 1]$:

$$\begin{aligned} F_G^{-1}(u) &:= \min\{i \in \mathbb{N}_0 \mid F_G(i) \geq u\} = \min\{i \in \mathbb{N}_0 \mid 1 - (1-p)^i \geq u\} \\ &= \min\left\{i \in \mathbb{N}_0 \mid i \geq \frac{\log(1-u)}{\log(1-p)}\right\} = \left\lceil \frac{\log(1-u)}{\log(1-p)} \right\rceil. \end{aligned}$$

According to the method, the following should work:

sample $U \sim \mathcal{U}([0,1])$

**return** $G = \left\lceil \frac{\log(1-U)}{\log(1-p)} \right\rceil$

We can also verify that everything worked by checking that the $G$ produced by the algorithm has the desired distribution function:

$$\begin{aligned}\Pr[G \le i] &= \Pr\left[\left\lceil \frac{\log(1-U)}{\log(1-p)} \right\rceil \le i\right] = \Pr\left[\frac{\log(1-U)}{\log(1-p)} \le i\right] = \Pr[\log(1-U) \ge i\log(1-p)] \\ &= \Pr[1-U \ge (1-p)^i] = \Pr[U \le 1-(1-p)^i] = 1-(1-p)^i.\end{aligned}$$

## Exercise 5 – Sampling without Replacement

We consider algorithms that, for $k, n \in \mathbb{N}$ with $0 \le k \le n/2$, compute a set $S \subseteq [n]$ of size $k$, chosen uniformly at random among all subsets of $[n]$ of size $k$.

(a) Why can we assume $k \le n/2$ without loss of generality?

(b) Describe an algorithm that has an expected runtime of $O(k \log k)$.
**Hint:** Rejection sampling and search tree.

(c) **Bonus:** Design an algorithm that has a worst-case runtime of $O(k \log k)$.

**Remark:** A nice summary of algorithms is found here: `https://github.com/ciphergoth/sansreplace/blob/master/algorithms.md`

## Solution 5

(a) $S \subseteq [n]$ is a random set of size $k$ if and only if $[n] \setminus S$ is a random set of size $n-k$.

(b) Conceptually, the algorithm samples *with* replacement, stores the results in a search tree, and ignores any samples that have already occurred. It continues until $k$ distinct results have been obtained. This is a form of rejection sampling, and it is quite clear that it is correct.

**Algorithm** SampleWithoutReplacement($n, k$)**:**
- $S \leftarrow \emptyset$ // as search tree
- **while** $|S| < k$ **do**
  - sample $X \sim \mathcal{U}(\{1, \dots, n\})$
  - **if** $X \notin S$ **then**
    - $S \leftarrow S \cup \{X\}$
- **return** $S$

By the assumption from (a) and the loop condition, at the beginning of each iteration we have $|S| < k \le n/2$. Thus, the probability of drawing something we already have is

always at most $1/2$. It follows that the number $F$ of unsuccessful iterations is expected to be at most the number of successful ones, i.e. $\mathbb{E}[F] \leq k$.

The total runtime is $T = (k + F) \cdot O(\log k)$ because there are $k + F$ iterations, each performing search tree operations in $O(\log k)$. Hence $\mathbb{E}[T] = O(k \log k)$.

(c) The idea is to explicitly manage the set of elements that can still be drawn. In the following, Array$[1..n]$ is used, which always contains a permutation of the set $\{1, \ldots, n\}$. At the beginning of iteration $i$, Array$[1..i-1]$ contains the elements already drawn, and Array$[i..n]$ contains those still available.

**Algorithm** SampleWithoutReplacement($n, k$)**:**
  Array $= [1, 2, \ldots, n]$ // everything still drawable
  **for** $i = 1$ **to** $k$ **do**
    sample $j \sim \mathcal{U}(\{i, \ldots, n\})$
    swap Array$[j]$ and Array$[i]$ // does nothing if $j = i$
  **return** Array$[1..k]$

Unfortunately, this results in a runtime of $O(n + k)$ because of the array initialization. However, this can be fixed. Clearly, at most $2k$ indices $i$ can satisfy Array$[i] \neq i$. It therefore suffices to store only these exceptional positions in a search tree. This yields a runtime of $O(k \log k)$.

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 5 – Randomized Complexity Classes

## Probability and Computing

## Exercise 1 – Relations between Complexity Classes

Justify the following inclusions:

(i) $\mathbf{ZPP} \subseteq \mathbf{RP}$ and $\mathbf{ZPP} \subseteq \text{co}-\mathbf{RP}$

(ii) $\mathbf{P} \subseteq \mathbf{ZPP}$

(iii) $\mathbf{RP} \subseteq \mathbf{NP}$ and $\text{co}-\mathbf{RP} \subseteq \text{co}-\mathbf{NP}$

(iv) $\mathbf{RP} \subseteq \mathbf{BPP}$ and $\text{co}-\mathbf{RP} \subseteq \mathbf{BPP}$

(v) $\mathbf{BPP} \subseteq \mathbf{PP}$

## Solution 1

(i) This follows directly from the definition $\mathbf{ZPP} := \mathbf{RP} \cap \text{co}-\mathbf{RP}$.

(ii) Let $L \in \mathbf{P}$. We show that $L \in \mathbf{ZPP}$. Let $T$ be a deterministic polynomial-time Turing machine for $L$. Using the "typecasting" argument from the lecture, we can formally regard $T$ as a probabilistic Turing machine (which in fact uses no randomness) that still decides $L$ and still runs in polynomial time. In particular, this PTM witnesses that $L \in \mathbf{RP}$. We have $\bar{L} \in \mathbf{P}$ because $\mathbf{P} = \text{co}-\mathbf{P}$, and by the same reasoning, $\bar{L} \in \mathbf{RP}$. Therefore, $L \in \text{co}-\mathbf{RP}$. Altogether we obtain $L \in \mathbf{RP} \cap \text{co}-\mathbf{RP} = \mathbf{ZPP}$.

(iii) Let $L \in \mathbf{RP}$ and let $T$ be an $\mathbf{RP}$-PTM for $L$. By "forgetting" the probabilistic choices, we obtain a nondeterministic Turing machine $T'$. Since for every $w \in L$ we have $\Pr[T(w) = \text{YES}] \geq \frac{1}{2}$, there exists at least one accepting computation path for $w$. Thus $T'(w) = \text{YES}$. For every $w \notin L$ we have $\Pr[T(w) = \text{YES}] = 0$, hence there exists no accepting computation for $w$. Therefore $T'(w) = \text{NO}$. Consequently, $T'$ decides $L$, and we have $L \in \mathbf{NP}$. As $L$ was arbitrary, it follows that $\mathbf{RP} \subseteq \mathbf{NP}$.

For the symmetric case we conclude analogously:

$$L \in \text{co}-\mathbf{RP} \Leftrightarrow \bar{L} \in \mathbf{RP} \Rightarrow \bar{L} \in \mathbf{NP} \Leftrightarrow L \in \text{co}-\mathbf{NP}.$$

(iv) Let $L \in \mathbf{RP}$ and let $T$ be an **RP**-PTM for $L$. Define a PTM $T'$ that runs $T$ three times independently and accepts if at least one of the three runs of $T$ accepts. Then, for all $w \in L$,

$$\Pr[T'(w) = \text{NO}] = \Pr[T(w) = \text{NO}]^3.$$

Hence for $w \in L$ we have:

$$\begin{aligned}\Pr[T'(w) = \text{YES}] &= 1 - \Pr[T'(w) = \text{NO}] = 1 - \Pr[T(w) = \text{NO}]^3 \\ &= 1 - (1 - \Pr[T(w) = \text{YES}])^3 \geq 1 - \left(1 - \frac{1}{2}\right)^3 = 1 - \frac{1}{8} > \frac{3}{4}.\end{aligned}$$

For $w \notin L$ we obtain:

$$\Pr[T'(w) = \text{YES}] = 1 - \Pr[T'(w) = \text{NO}] = 1 - \Pr[T(w) = \text{NO}]^3 = 1 - 1^3 = 0 < \frac{1}{4}.$$

Thus, $T'$ is a **BPP**-PTM for $L$. Hence $L \in \mathbf{BPP}$. Since $L$ was arbitrary, we have $\mathbf{RP} \subseteq \mathbf{BPP}$.

The symmetric case follows analogously, since $\mathbf{BPP} = \text{co}-\mathbf{BPP}$.

(v) There is nothing to prove here. Every **BPP**-PTM is also a **PP**-PTM, as the acceptance requirement is merely relaxed. Hence, for every language $L$ that has a **BPP**-PTM, there also exists a **PP**-PTM.

## Exercise 2 – Las Vegas Algorithm for $L$ implies $L \in$ ZPP

Let **LV** (for Las Vegas) denote the class of all languages $L$ for which there exists a probabilistic Turing machine $T$ with the following properties:

- $T$ decides $L$ (that is, $T$ outputs 1 for all $x \in L$ and 0 for all $x \notin L$).
- There exists a polynomial $p(n)$ such that the *expected* runtime of $T$ on input $x$ is bounded by $p(|x|)$.

In the lecture we proved that $\mathbf{ZPP} \subseteq \mathbf{LV}$. Show that also $\mathbf{LV} \subseteq \mathbf{ZPP}$ holds. Thus, the two classes are identical.

**Hint:** Definition of **ZPP**, Markov inequality.

## Solution 2

Let $L \in \mathbf{LV}$. We first show that $L \in \mathbf{RP}$. Let $T$ be an **LV**-TM for $L$ with associated polynomial $p(n)$. We consider the following Turing machine $T'$:

**Algorithm** $T'(w)$**:**
  $t_{\max} \leftarrow 2p(|w|)$
  simulate $T$ on input $w$ for $t_{\max}$ steps
  **if** *$T$ has terminated with output $r$* **then**
    **return** $r$
  **else** // $T$ has not yet terminated
    **return** NO

We now show that $T'$ is an **RP**-TM for $L$. Due to the choice of $t_{\max}$, the runtime of $T'(w)$ is clearly bounded by some polynomial $q(|w|)$. Concerning the outputs of $T'$:

- For $w \notin L$, the output of $T'(w)$ is always NO, either because $T$ decided so or because we reached the else-case. Hence, $\Pr[T'(w) = \text{YES}] = 0$.
- For $w \in L$, let $t(w)$ denote the random runtime of $T$ on $w$. By assumption, $\mathbb{E}[t(w)] \leq p(|w|)$. By Markov's inequality it follows that:

$$\Pr[t(w) > t_{\max}] \leq \frac{\mathbb{E}[t(w)]}{t_{\max}} \leq \frac{p(|w|)}{2p(|w|)} \leq \frac{1}{2}.$$

  Thus, $T$ terminates on $w$ within the given time limit (with the correct result) with probability at least $1/2$. Therefore, $\Pr[T'(w) = \text{YES}] \geq \frac{1}{2}$.

Hence, $T'$ is an **RP**-TM for $L$. Thus, $L \in \mathbf{RP}$. Analogously, $L \in \text{co}-\mathbf{RP}$ follows (by returning YES in the non-termination case). Therefore, $L \in \mathbf{ZPP}$. Since $L$ was arbitrary, we obtain $\mathbf{LV} \subseteq \mathbf{ZPP}$.

## Exercise 3 – Bonus: Probability Amplification for BPP

Look up on Wikipedia what the complexity class **P/poly** means. Show that $\mathbf{BPP} \subseteq \mathbf{P/poly}$. This insight is also known as *Adleman's Theorem*.

**Hint:** Make the error probability smaller than $2^{-n}$. Transform the PTM into a DTM and show that there exists a random string that yields the correct result for all inputs.

## Solution 3

No solution for the bonus exercise.

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 4 – Probability Amplification

## Probability and Computing

## Exercise 1 – Probability Amplification with Two-Sided Error

Suppose a Monte Carlo algorithm $A$ solves a decision problem correctly with probability $1/2 + \varepsilon$ and incorrectly otherwise (for some $\varepsilon > 0$). Let $A'$ be the algorithm that executes $A$ independently $t$ times and decides according to the majority outcome. Show that the error probability of $A'$ is at most $e^{-2t\varepsilon^2}$.

**Hint**: The following may be useful: $\sum_{i=0}^{t} \binom{t}{i} = 2^t$.

## Solution 1

Let $I \in \{0, \ldots, t\}$ denote the number of executions producing the correct result. For $A'$ to return an incorrect answer, it must hold that $I \leq t/2$. The result follows from the following estimation:

$$\begin{aligned}
\Pr[I \leq t/2] &= \sum_{i=0}^{\lfloor t/2 \rfloor} \Pr[I = i] = \sum_{i=0}^{\lfloor t/2 \rfloor} \binom{t}{i} (\tfrac{1}{2} + \varepsilon)^i (\tfrac{1}{2} - \varepsilon)^{t-i} \leq \sum_{i=0}^{\lfloor t/2 \rfloor} \binom{t}{i} (\tfrac{1}{2} + \varepsilon)^{t/2} (\tfrac{1}{2} - \varepsilon)^{t/2} \\
&\leq (\tfrac{1}{4} - \varepsilon^2)^{t/2} \sum_{i=0}^{\lfloor t/2 \rfloor} \binom{t}{i} \leq (\tfrac{1}{4} - \varepsilon^2)^{t/2} \sum_{i=0}^{t} \binom{t}{i} = (\tfrac{1}{4} - \varepsilon^2)^{t/2} \cdot 2^t \\
&= (1 - 4\varepsilon^2)^{t/2} \leq (e^{-4\varepsilon^2})^{t/2} = e^{-2t\varepsilon^2}.
\end{aligned}$$

## Exercise 2 – Pathfinder

Let $G$ be an undirected graph with $n$ vertices and $m$ edges. We seek a path of length $k$ that does not visit any vertex more than once. The naive brute-force approach runs in time $O(n^k)$. We now consider a simple randomized approach that works as follows. In the first step, each vertex $v$ is assigned a label $L(v)$ uniformly at random from $\{1, \ldots, k\}$. In the second step, for every vertex $v$ with $L(v) = 1$, a modified breadth-first search is started, where a vertex $w$ can be discovered from a vertex $u$ only if $L(u) = L(w) + 1$ holds. If a vertex with label $k$ is reached, a path of length $k$ is constructed and output.

(a) Show that the algorithm finds a path of length $k$ with probability $1/k^k$, if such a path exists.

(b) Improve the success probability to $1 - 1/n$ by probability amplification. What is the resulting total running time?

**Remark:** This idea is known as "Color Coding". There exist more refined variants.

## Solution 2

(a) Let $P = (v_1, \ldots, v_k)$ be a path of length $k$ in $G$. With probability $1/k^k$, each $v_i$ receives color $i$ for all $i \in [k]$. In this case, the breadth-first search will reach all vertices $v_1, \ldots, v_k$ (along $P$ or another path) and thus find a path of length $k$.

(b) We repeat the algorithm $t = k^k \cdot \ln n$ times. The success probability then becomes:

$$1 - (1 - k^{-k})^t \geq 1 - (e^{-k^{-k}})^t = 1 - e^{-\ln n} = 1 - \tfrac{1}{n}.$$

Since $n$ breadth-first searches can be performed in time $O(nm)$, the overall runtime is $O(nm \cdot k^k \cdot \ln n)$.

## Exercise 3 – Bonus: Random-Walk Solver for 3-SAT

Let $\varphi(x_1, \ldots, x_n)$ be a 3-SAT formula with $n$ variables and $m$ clauses. We seek a satisfying assignment using the following algorithm:

**Algorithm** randomWalkSolver($\varphi$)**:**
- sample $x_1, \ldots, x_n \sim \mathrm{Ber}(1/2)$
- **for** $k = 1$ **to** $n/2$ **do**
  - **if** $\varphi(x_1, \ldots, x_n) = 1$*)* **then**
    - **break**
  - let $C$ be an arbitrary unsatisfied clause of $\varphi$
  - sample $j \sim \mathcal{U}(\{1, 2, 3\})$
  - let $x_i$ be the $j$th variable in $C$
  - $x_i \leftarrow 1 - x_i$
- **if** $\varphi(x_1, \ldots, x_n) = 1$*)* **then**
  - **return** $(x_1, \ldots, x_n)$
- **return** $\perp$

If $\varphi$ is unsatisfiable, clearly randomWalkSolver($\varphi$) $= \perp$. Otherwise, let $x^*$ be a satisfying assignment. Show that:

(a) With probability at least $1/2$, the initial random assignment $(x_1, \ldots, x_n)$ agrees with $x^*$ on at least $n/2$ variables.

(b) If $\varphi(x_1, \ldots, x_n) = 0$, then one iteration of the loop will, with probability $1/3$, flip a variable so that it matches $x^*$ on one more position.

(c) Use probability amplification to obtain an algorithm with success probability $1 - 1/n$. What is the total running time?

## Solution 3

(a) For every "bad" assignment that agrees with $x^*$ on fewer than $n/2$ variables, the bitwise complement agrees with $x^*$ on more than $n/2$ variables. Thus, "bad" assignments make up at most half of all possible assignments. (Assignments agreeing on exactly $n/2$ variables yield a slight bias in our favor when $n$ is even).

(b) In the unsatisfied clause $C$ considered during the loop, at least one variable must differ between $(x_1, \ldots, x_n)$ and $x^*$, since $x^*$ satisfies $C$. With probability $1/3$, we select this variable.

(c) From (a) and (b), the success probability for one run is at least $\frac{1}{2} \cdot 3^{-n/2}$ (intuitively: we start near $x^*$ and move toward it). Repeating the algorithm $t = 2 \cdot 3^{n/2} \cdot \ln n$ times yields a success probability of

$$1 - (1 - \tfrac{1}{2} \cdot 3^{-n/2})^t \geq 1 - (e^{-\frac{1}{2} \cdot 3^{-n/2}})^t = 1 - e^{-\ln n} = 1 - \tfrac{1}{n}.$$

If $m$ is the number of clauses in $\varphi$, then randomWalkSolver runs in time $O(nm)$. Altogether, this gives a total runtime of $O(3^{n/2} \cdot nm)$, which can be asymptotically better than the naive $O(2^n)$ algorithm.

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 6 – Concentration Bounds

## Probability and Computing

## Exercise 1 – Algebraic Rule for Expectation

Let $X, Y$ be independent random variables whose expectation exists. Show that

$$\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y].$$

**Hint:** Use the definition of independence for discrete random variables, which guarantees

$$\Pr[X = i \wedge Y = j] = \Pr[X = i] \cdot \Pr[Y = j] \quad \text{for all } i, j.$$

## Solution 1

Let $R_X, R_Y \subseteq \mathbb{R}$ be countable sets containing all possible values of $X$ and $Y$. Then:

$$\begin{aligned}
\mathbb{E}[X \cdot Y] &= \sum_{(x,y) \in R_X \times R_Y} x \cdot y \cdot \Pr[X = x \wedge Y = y] \\
&\overset{\text{indep.}}{=} \sum_{x \in R_X} \sum_{y \in R_Y} x \cdot y \cdot \Pr[X = x] \Pr[Y = y] \\
&= \sum_{x \in R_X} \left( x \cdot \Pr[X = x] \sum_{y \in R_Y} y \cdot \Pr[Y = y] \right) \\
&= \left( \sum_{x \in R_X} x \cdot \Pr[X = x] \right) \left( \sum_{y \in R_Y} y \cdot \Pr[Y = y] \right) \\
&= \mathbb{E}[X] \cdot \mathbb{E}[Y].
\end{aligned}$$

## Exercise 2 – Algebraic Rules for Variance

Let $X, Y$ be independent random variables with existing variance. Let $s, t > 0$. Show:

(a) $\mathrm{Var}(sX) = s^2 \mathrm{Var}(X)$

(b) $\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$

(c) $\mathrm{Var}(sX + tY) = s^2 \mathrm{Var}(X) + t^2 \mathrm{Var}(Y)$

**Hint:** Use linearity of expectation and the result of the previous exercise, i.e., $\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$ for independent $X$ and $Y$.

## Solution 2

We now prove the three variance rules (very explicitly).

(a) Here, besides the definition of variance, we only need the insight that $\mathbb{E}[sZ] = s\mathbb{E}[Z]$ (linearity of expectation). The latter holds for every random variable $Z$ whose expectation exists, and for every $s \in \mathbb{R}$. Thus:

$$\begin{aligned}\mathrm{Var}(sX) &= \mathbb{E}[(sX - \mathbb{E}[sX])^2] = \mathbb{E}[(sX - s\mathbb{E}[X])^2] \\ &= \mathbb{E}[s^2(X - \mathbb{E}[X])^2] = s^2\mathbb{E}[(X - \mathbb{E}[X])^2] = s^2\,\mathrm{Var}(X).\end{aligned}$$

(b) First, centered random variables have expectation 0:

$$\mathbb{E}[X - \mathbb{E}[X]] = \mathbb{E}[X] - \mathbb{E}[\mathbb{E}[X]] = \mathbb{E}[X] - \mathbb{E}[X] = 0. \tag{1}$$

If $X, Y$ are independent, then $X - c$ and $Y - d$ are also independent for constants $c, d \in \mathbb{R}$. Setting $c = \mathbb{E}[X]$, $d = \mathbb{E}[Y]$, we obtain independence of $X - \mathbb{E}[X]$ and $Y - \mathbb{E}[Y]$. Thus:

$$\mathbb{E}[(X - \mathbb{E}[X]) \cdot (Y - \mathbb{E}[Y])] = \mathbb{E}[X - \mathbb{E}[X]] \cdot \mathbb{E}[Y - \mathbb{E}[Y]] \overset{(1)}{=} 0 \cdot 0 = 0. \tag{2}$$

Now consider $(X+Y-\mathbb{E}[X+Y])^2$, the expectation of which is the variance we are looking for:

$$\begin{aligned}(X + Y - \mathbb{E}[X + Y])^2 &= (X + Y - \mathbb{E}[X] - \mathbb{E}[Y])^2 \\ &= ((X - \mathbb{E}[X]) + (Y - \mathbb{E}[Y]))^2 \\ &= (X - \mathbb{E}[X])^2 + (Y - \mathbb{E}[Y])^2 + 2(X - \mathbb{E}[X])(Y - \mathbb{E}[Y]).\end{aligned}$$

Taking expectations:

$$\begin{aligned}\mathrm{Var}(X + Y) &= \mathbb{E}[(X + Y - \mathbb{E}[X + Y])^2] \\ &= \mathbb{E}[(X - \mathbb{E}[X])^2] + \mathbb{E}[(Y - \mathbb{E}[Y])^2] + 2\mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] \\ &= \mathrm{Var}(X) + \mathrm{Var}(Y) + 0\end{aligned}$$

using the definition of variance and (2).

(c) Since $sX$ and $tY$ are independent when $X$ and $Y$ are, we obtain directly from (a) and (b):

$$\mathrm{Var}(sX + tY) \overset{(b)}{=} \mathrm{Var}(sX) + \mathrm{Var}(tY) \overset{(a)}{=} s^2\,\mathrm{Var}(X) + t^2\,\mathrm{Var}(Y).$$

## Exercise 3 – Chernoff in Even Simpler Form for Large Deviations

Let $X = X_1 + \cdots + X_n$ be a sum of independent Bernoulli random variables with $\mu = \mathbb{E}[X]$ and let $b \geq 6\mu$. Show

$$\Pr[X \geq b] \leq 2^{-b}.$$

**Hint:** Use the Chernoff bound $\Pr[X \geq (1+\delta)\mu] \leq (\frac{e^\delta}{(1+\delta)^{1+\delta}})^\mu$.

## Solution 3

Following the hint and setting $\delta = b/\mu - 1$, we obtain:

$$\begin{aligned}\Pr[X \geq b] = \Pr[X \geq (1+\delta)\mu] &\leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \\ &\leq \left(\frac{e^{b/\mu}}{(b/\mu)^{b/\mu}}\right)^{\mu} = \left(\frac{e}{b/\mu}\right)^{b} \leq \left(\frac{e}{6}\right)^{b} \leq 2^{-b}.\end{aligned}$$

## Exercise 4 – Comparing Concentration Inequalities

For $n \in \mathbb{N}$ let $X_n$ be the number of sixes when rolling a fair die $n$ times. Let $p_n$ be the probability that $X_n$ exceeds its expectation by at least 10%. For each of the following, find an upper bound on $p_n$ using...

(a) ... Markov's Inequality.

(b) ... Chebyshev's Inequality.

(c) ... the Chernoff bound (or a variant).

(d) Compare the asymptotic strength of the bounds.

## Solution 4

Preparations:

- $\mu := \mathbb{E}[X] = n/6$.
- $\mathrm{Var}(X) = n \cdot (\frac{1}{6}(\frac{5}{6})^2 + \frac{5}{6}(\frac{1}{6})^2) = \frac{5}{36}n$.
- We bound $p_n = \Pr[X \geq 1.1\mu] \leq \Pr[|X - \mu| \geq 0.1\mu]$.

(a) $p_n = \Pr[X \geq 1.1\mu] \leq \mu/(1.1\mu) = \frac{10}{11} = \Theta(1)$.

(b) $p_n \leq \Pr[|X - \mu| \geq 0.1\mu] \leq \frac{\mathrm{Var}(X)}{(0.1\mu)^2} = \frac{5n/36}{0.01 \cdot n^2/36} = \frac{500}{n} = \Theta(1/n)$.

(c) $p_n = \Pr[X \geq (1+0.1)\mu] \leq \exp(-\frac{0.1^2}{2+0.1}\mu) = \exp(-\frac{1}{210}\mu) = \exp(-\frac{1}{1260}n) = \exp(-\Theta(n))$.

(d) Asymptotically, the Chernoff bound is the strongest and Markov the weakest.
**Remark:** While Markov and Chernoff give bounds $< 1$ for all $n \in \mathbb{N}$, Chebyshev becomes nontrivial only for $n \geq 501$.

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 7 – Classic Hash Tables

## Probability and Computing

### Exercise 1 – 2-Independence vs. 1-Universality

Let $\mathcal{H} \subseteq [m]^D$ be a family of hash functions mapping $D$ to $[m]$. Prove or disprove the following implications:

(a) $\mathcal{H}$ is 2-independent $\Rightarrow$ $\mathcal{H}$ is 1-universal.

(b) $\mathcal{H}$ is 1-universal $\Rightarrow$ $\mathcal{H}$ is 2-independent.

**Hint:** In one case, the implication is straightforward. In the other, trivial counterexamples exist.

### Solution 1

(a) The implication holds. For any $x \neq y \in D$, by the definition of 2-independence:

$$\forall i, j \in [m] : \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x) = i \wedge h(y) = j] = \frac{1}{m^2}.$$

Consequently, the collision probability for $x$ and $y$ under $\mathcal{H}$ is bounded as follows:

$$\begin{aligned}\Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x) = h(y)] &= \sum_{i=1}^{m} \Pr_{h \sim \mathcal{U}(\mathcal{H})}[h(x) = i \wedge h(y) = i] \\ &= \sum_{i=1}^{m} \frac{1}{m^2} = \frac{1}{m}.\end{aligned}$$

This verifies the condition for 1-universality.

(b) The implication does not hold. This is primarily due to “trivial” counterexamples.

**Example 1.** Take $D = [m]$ and $\mathcal{H} = \{\mathrm{id}\}$. The identity function never causes collisions; hence, $\mathcal{H}$ is even 0-universal (and thus 1-universal). However, $\mathcal{H}$ is not 2-independent: the hash values are not uniformly distributed in $[m]$—in fact, they are deterministic.

**Example 2.** Consider the class $\mathcal{H} = \mathcal{H}^{\text{lin}}_{p,m}$ from lecture, parameterized by $p$ and $m$, such that $m$ does not divide $p(p-1)$. As shown in lecture, $\mathcal{H}$ is 1-universal. Since $|\mathcal{H}| = p(p-1)$, all relevant probabilities (of the form $\Pr_{h\sim\mathcal{H}}[\ldots]$) are multiples of $\frac{1}{p(p-1)}$. However, $\frac{1}{m}$ is not such a multiple. Consequently, $\Pr_{h\sim\mathcal{H}}[h(x) = 0] = \frac{1}{m}$ cannot hold for any $x$. Hence, the hash value of $x$ is not uniformly distributed in $[m]$.

## Exercise 2 – $d$-Independence without Mutual Independence

Alice and Bob each spin a roulette wheel with 10 equally sized segments labeled 0 to 9. Let $A$ and $B$ denote Alice's and Bob's outcomes, respectively. Define $C = (A + B) \bmod 10$.

(a) Show that $A$, $B$, and $C$ are pairwise independent.

(b) Show that $A$, $B$, and $C$ are not mutually independent.

(c) For any $d \in \mathbb{N}$, construct a family of random variables that is $d$-independent but not fully independent.

## Solution 2

We solve the problem for arbitrary $d, m \in \mathbb{N}$ (instead of $d = 2$ and $m = 10$). Specifically, let $A_1, A_2, \ldots, A_d \sim \mathcal{U}([m])$ be $d$ mutually independent uniform random variables over $[m]$, and define $C := (A_1 + \cdots + A_d) \bmod m$. It is straightforward to verify $C \sim \mathcal{U}([m])$: regardless of the values $A_1, \ldots, A_{d-1}$, the $m$ possible values of $A_d$ yield each residue modulo $m$ for $C$ with equal probability. We prove two properties:

**The family $\{A_1, \ldots, A_d, C\}$ is not mutually independent.** We have $\Pr[\forall i \in [d] : A_i = 0] = m^{-d}$ and $\Pr[C = 0] = m^{-1}$. However, the event $\forall i \in [d] : A_i = 0$ implies $C = 0$, so $\Pr[C = 0 \wedge \forall i \in [d] : A_i = 0] = m^{-d}$. Had mutual independence of $\{A_1, \ldots, A_d, C\}$ held, we would have obtained $m^{-(d+1)}$.

**The family $\{A_1, \ldots, A_d, C\}$ is $d$-wise independent.** Consider any selection of $d$ variables and the event that they attain specific values. We must show the probability of this event equals the product of individual probabilities. By symmetry (since $A_1, \ldots, A_d$ play identical roles), we only distinguish two cases: whether $C$ is selected or not:

$$\Pr[A_1 = a_1 \wedge \cdots \wedge A_d = a_d] \overset{!}{=} \prod_{i=1}^{d} \Pr[A_i = a_i] = m^{-d},$$

$$\Pr[A_1 = a_1 \wedge \cdots \wedge A_{d-1} = a_{d-1} \wedge C = c] \overset{!}{=} \Pr[C = c] \cdot \prod_{i=1}^{d-1} \Pr[A_i = a_i] = m^{-d}.$$

The "$\overset{!}{=}$" must be proven. In the first case, this holds trivially by the independence of $A_1, \ldots, A_d$. For the second case, consider the event:

$$E = \{A_1 = a_1 \wedge \cdots \wedge A_{d-1} = a_{d-1} \wedge C = c\},$$

where $a_1, \ldots, a_{d-1}, c$ are fixed. By definition of $C$,

$$E = \{A_1 = a_1 \wedge \cdots \wedge A_{d-1} = a_{d-1} \wedge A_1 + \cdots + A_{d-1} + A_d = c\}.$$

Since $A_1, \ldots, A_{d-1}$ are fixed, this is equivalent to:

$$E = \{A_1 = a_1 \wedge \cdots \wedge A_{d-1} = a_{d-1} \wedge A_d = c - a_1 - \cdots - a_{d-1}\}.$$

In this form, it is clear that $\Pr[E] = m^{-d}$, since $A_1, \ldots, A_d$ are independent and uniformly distributed.

**A Remark.** If one examines our definitions very closely, one might still have a concern. Although we showed above that for any selection of $d$ variables, we can factor the probability of a joint event into a product, the definition of $d$-independence refers to "up to" $d$ variables. What if we select only $k$ variables, where $k < d$? Does it automatically follow that these $k$ variables are also independent? The answer is "yes".

Consider, for example, the event:

$$E = \{A_1 = a_1 \wedge \cdots \wedge A_{k-1} = a_{k-1} \wedge C = c\}.$$

We must show that for this event $E$,

$$\Pr[E] \overset{!}{=} \Pr[C = c] \cdot \prod_{i=1}^{k-1} \Pr[A_i = a_i] = m^{-k}.$$

This is achieved by introducing additional case distinctions over the remaining random variables and reusing the prior result:

$$\begin{aligned}
\Pr[E] &= \Pr[A_1 = a_1 \wedge \cdots \wedge A_{k-1} = a_{k-1} \wedge C = c] \\
&= \sum_{a_k=0}^{m-1} \sum_{a_{k+1}=0}^{m-1} \cdots \sum_{a_{d-1}=0}^{m-1} \Pr[A_1 = a_1 \wedge \cdots \wedge A_{d-1} = a_{d-1} \wedge C = c] \\
&= \sum_{a_k=0}^{m-1} \sum_{a_{k+1}=0}^{m-1} \cdots \sum_{a_{d-1}=0}^{m-1} m^{-d} = m^{d-k} \cdot m^{-d} = m^{-k}.
\end{aligned}$$

## Exercise 3 – Find the Error

Let $p$ be prime, $\mathbb{F}_p = \{0, \ldots, p-1\}$ and $m \in \mathbb{N}$. Consider the following class of hash functions from $\mathbb{F}_p$ to $[m]$, also mentioned in the lecture.

$$\mathcal{H} = \{x \mapsto ((a \cdot x) \bmod p) \bmod m \mid a \in \mathbb{F}_p^*\}.$$

Consider the following argument that $\mathcal{H}$ is 1-universal. Find the mistake in the proof.

The proof considers arbitrary $x, y \in \mathbb{F}_p$ with $x \neq y$. It has six steps.

$$\begin{aligned}
\Pr_{h\sim\mathcal{H}}[h(x) = h(y)] &\overset{1}{=} \Pr_{a\sim\mathcal{U}(\mathbb{F}_p^*)}[(ax \bmod p) \bmod m = (ay \bmod p) \bmod m] \\
&\overset{2}{=} \Pr_{a\sim\mathcal{U}(\mathbb{F}_p^*)}[((ax \bmod p) - (ay \bmod p)) \bmod m = 0] \\
&\overset{3}{=} \Pr_{a\sim\mathcal{U}(\mathbb{F}_p^*)}[((ax - ay) \bmod p) \bmod m = 0] \\
&\overset{4}{=} \Pr_{a\sim\mathcal{U}(\mathbb{F}_p^*)}[(a(x - y) \bmod p) \bmod m = 0] \\
&\overset{5}{=} \Pr_{u\sim\mathcal{U}(\mathbb{F}_p^*)}[u \bmod m = 0] \\
&\overset{6}{=} \frac{|\{m, 2m, 3m, \ldots, \} \cap \mathbb{F}_p^*|}{|\mathbb{F}_p^*|} \\
&\overset{7}{\leq} \frac{1}{m}.
\end{aligned}$$

In Step 5 we use that the function $a \mapsto az \bmod p$ is a bijection on $\mathbb{F}_p^*$ for any fixed $z \in \mathbb{F}_p^*$. Therefore, if $a \sim \mathcal{U}(\mathbb{F}_p^*)$ and $u := az$ then $u \sim \mathcal{U}(\mathbb{F}_p^*)$.

## Solution 3

The error is in Step 3. In general it is not true that

$$(c \bmod p) - (d \bmod p) = (c - d) \bmod p.$$

The left hand side may even produce negative values! It is true however that

$$(c \bmod p) - (d \bmod p) \in \{(c - d) \bmod p, (c - d) \bmod p - p\}.$$

**Remark:** Adapting the argument to track through both cases we can get an upper bound of $\frac{2}{m} + \frac{1}{p-1}$. This almost proves 2-universality (assuming $p \gg m$). The details are somewhat annoying.

## Exercise 4 – Bonus: Concentration Bounds for Sums of $d$-wise Independent Random Variables

Let $d \in \mathbb{N}$ be even, and $\{X_1, \ldots, X_n\}$ be a $d$-wise independent family of random variables, each distributed as $\mathrm{Ber}(p)$ with $p = \Omega(1/n)$.

Define $X = \sum_{i=1}^n X_i$. Note: $X$ is not necessarily binomially distributed since the $X_i$ are not mutually independent.

The goal is to prove the concentration bound: for any $\delta > 0$,

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

To this end, consider the "centered" random variables $Y_i := X_i - p$, their sum $Y = \sum_{i=1}^n Y_i$, and the moment $\mathbb{E}[Y^d]$.

(i) Warm-up: Let $d \geq 3$ and $n \geq 3$. Verify and briefly explain why the following hold:

(a) $\mathbb{E}[Y_1^5 Y_2^{42}] = \mathbb{E}[Y_1^5]\mathbb{E}[Y_2^{42}]$

(b) $\mathbb{E}[Y_1^5 Y_2^{42} Y_3] = 0$

(c) $\mathbb{E}[Y_1^5] \leq \mathbb{E}[Y_1^2]$

In subsequent steps, you may apply these insights without further justification.

(ii) Show: $\mathbb{E}[Y_1^2] \leq p$.

(iii) Let $i_1, \ldots, i_d \in [n]$ (not necessarily distinct) and $S = \{i_1, \ldots, i_d\}$. Prove:

- If $|S| > d/2$, then $\mathbb{E}[Y_{i_1} \cdots Y_{i_d}] = 0$.
- Otherwise, $\mathbb{E}[Y_{i_1} \cdots Y_{i_d}] \leq p^{|S|}$.

(iv) Show: $\mathbb{E}[Y^d] = O((np)^{d/2})$. You may assume $d = O(1)$. **Hint:** Expand $(\sum_{i=1}^n Y_i)^d$. Yes, this yields $n^d$ terms.

(v) Prove the original goal by applying Markov's inequality to $Y^d$.

## Solution 4

(i) Since $d \geq 3$, for any distinct $i_1, i_2, i_3 \in [n]$, the random variables $X_{i_1}, X_{i_2}, X_{i_3}$ are mutually independent. Hence, $Y_1^5$, $Y_2^{42}$, and $Y_3$ (as functions of $X_1, X_2, X_3$) are also mutually independent.

(a) For independent random variables, the expectation of the product equals the product of expectations, by definition.

(b) Factor the expectation: $\mathbb{E}[Y_1^5 Y_2^{42} Y_3] = \mathbb{E}[Y_1^5]\mathbb{E}[Y_2^{42}]\mathbb{E}[Y_3]$. Since $\mathbb{E}[Y_3] = \mathbb{E}[X_3 - p] = p - p = 0$, the product is zero.

(c) Since $|Y_1| \leq 1$ and $x^i$ is non-increasing for $x \in [0, 1]$ as $i$ increases,

$$\mathbb{E}[Y_1^5] \leq \mathbb{E}[|Y_1^5|] = \mathbb{E}[|Y_1|^5] \leq \mathbb{E}[|Y_1|^2] = \mathbb{E}[Y_1^2].$$

(ii) $\mathbb{E}[Y_1^2] = \mathbb{E}[(X_1 - p)^2] = p(1-p)^2 + (1-p)(0-p)^2 = p(1-p)(1-p+p) \leq p$.

(iii) The key question is whether any index appears exactly once in the multiset $\{i_1, \ldots, i_d\}$.

- If $|S| > d/2$, then at least one index $j$ appears exactly once. Then $\mathbb{E}[Y_{i_1} \cdots Y_{i_d}]$ factors such that $\mathbb{E}[Y_j] = 0$ appears as a multiplicative factor, so the expectation is zero.
- If $|S| \leq d/2$, then the product involves at most $d/2$ distinct variables. Factor as in part (i)(a). If any variable appears with exponent 1, the entire product vanishes (since $\mathbb{E}[Y_j] = 0$). Otherwise, each exponent is at least 2. Then apply (i)(c) and (ii) to bound each distinct factor by $p$, giving $p^{|S|}$.

(iv) We compute:

$$\begin{aligned}
\mathbb{E}[Y^d] &= \mathbb{E}\left[\left(\sum_{i=1}^{n} Y_i\right)^d\right] = \mathbb{E}\left[\sum_{i_1=1}^{n}\cdots\sum_{i_d=1}^{n} Y_{i_1}\cdots Y_{i_d}\right] \\
&\overset{(1)}{=} \sum_{i_1=1}^{n}\cdots\sum_{i_d=1}^{n}\mathbb{E}[Y_{i_1}\cdots Y_{i_d}] \\
&\overset{(2)}{=} \sum_{i_1,\ldots,i_d}\mathbb{E}[Y_{i_1}\cdots Y_{i_d}] \overset{(3)}{=} \sum_{r=1}^{d}\sum_{\substack{S\subseteq[n]\\|S|=r}}\sum_{i_1,\ldots,i_d} \mathbb{1}_{\{i_1,\ldots,i_d\}=S}\cdot\mathbb{E}[Y_{i_1}\cdots Y_{i_d}] \\
&\overset{(4)}{\le} \sum_{r=1}^{d/2}\sum_{\substack{S\subseteq[n]\\|S|=r}}\sum_{i_1,\ldots,i_d} \mathbb{1}_{\{i_1,\ldots,i_d\}=S}\cdot p^{|S|} \overset{(5)}{=} \sum_{r=1}^{d/2}\sum_{\substack{S\subseteq[n]\\|S|=r}} p^{|S|}\sum_{i_1,\ldots,i_d}\mathbb{1}_{\{i_1,\ldots,i_d\}=S} \\
&\overset{(6)}{\le} \sum_{r=1}^{d/2}\sum_{\substack{S\subseteq[n]\\|S|=r}} p^{|S|}\cdot|S|^d \overset{(7)}{=} \sum_{r=1}^{d/2}\binom{n}{r}p^r r^d \overset{(8)}{\le} (d/2)^d\sum_{r=1}^{d/2} n^r p^r \overset{(9)}{\le} O(n^{d/2}p^{d/2}).
\end{aligned}$$

(1) Linearity of expectation.

(2) Compact notation.

(3) Group terms by the set $S = \{i_1, \ldots, i_d\}$.

(4) By part (iii), terms with $|S| > d/2$ vanish; the others are bounded by $p^{|S|}$.

(5) Factor out $p^{|S|}$.

(6) For the indicator to be 1, all indices $i_1, \ldots, i_d$ must lie in $S$, which can occur in at most $|S|^d$ ways.

(7) The inner sum depends only on $r = |S|$, and there are $\binom{n}{r}$ such sets.

(8) Use $\binom{n}{r} \le n^r$ and $r \le d/2$.

(9) Since $d = O(1)$, $(d/2)^d = O(1)$. Since $p = \Omega(1/n)$, we have $np = \Omega(1)$, so the term at $r = d/2$ dominates the constant number of other terms.

(v) First, the calculation:

$$\begin{aligned}
\Pr[X - \mathbb{E}[X] \ge \delta\mathbb{E}[X]] &\overset{(1)}{=} \Pr[Y \ge \delta np] \le \Pr[|Y| \ge \delta np] = \Pr[|Y|^d \ge (\delta np)^d] \\
&\overset{(2)}{=} \Pr[Y^d \ge (\delta np)^d] \overset{(3)}{\le} \frac{\mathbb{E}[Y^d]}{(\delta np)^d} \overset{(4)}{\le} O\left(\frac{n^{d/2}p^{d/2}}{(\delta np)^d}\right) = O\big(\delta^{-d}(np)^{-d/2}\big).
\end{aligned}$$

(1) By definition of $Y$ and linearity, $\mathbb{E}[X] = np$.

(2) Since $d$ is even, $|Y|^d = Y^d$.

(3) Apply Markov's inequality to $Y^d$. Note that $Y^d \ge 0$ because $d$ is even.

(4) Substitute the result from part (iv).

Karlsruhe Institute of Technology

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 8 – Bounded Differences and Bloom Filters

## Probability and Computing

## Exercise 1 – Balls, Bins and Bounded Differences

Let $\lambda > 0$ be a constant, $m = \lambda n$ the number of balls, and $n$ the number of bins. The $j$-th ball is placed in bin $X_j$, where $X_1, \dots, X_m \sim \mathcal{U}([n])$ are independent random variables. The collision count is defined as $C = |\{(i,j) \mid 1 \le i < j \le m, X_i = X_j\}|$. The goal of this exercise is to derive a concentration bound for $C$.

(i) Show that $\mathbb{E}[C] = \Theta(n)$.

For $i \in [n]$, let $L_i = |\{j \in [m] \mid X_j = i\}|$ denote the load of bin $i$. It is neither difficult nor interesting to show that $\Pr[\max_{i\in[n]} L_i \le \log n] \ge 1 - O(n^{-100})$. Assume this to hold in the following (proof provided in the sample solution).

(ii) Define $C_{\text{cap}} := \sum_{i\in[n]} \binom{\min(\log n, L_i)}{2}$. Show that $\Pr[C_{\text{cap}} = C] = 1 - O(n^{-100})$.

(iii) Show that $\Pr[C_{\text{cap}} - \mathbb{E}[C_{\text{cap}}] \ge t] \le \exp(-2t^2/(m \cdot \log^2 n))$.

(iv) Show that $\Pr[C - \mathbb{E}[C] \ge n^{2/3}] = O(n^{-100})$.

(v) Assume an algorithm inserts $n$ keys into a hash table and then queries every key exactly once. Show that the algorithm runs in time $O(n)$ except with probability $O(n^{-100})$ under suitable assumptions. Note that we are *not* talking about *expected* running time.

## Solution 1

(i) Any two distinct balls collide with probability $1/n$. Hence, $\mathbb{E}[C] = \binom{m}{2} \cdot \frac{1}{n} = \frac{m(m-1)}{2n} = \frac{\lambda(m-1)}{2} = \Theta(n)$.

We now examine the claimed bound on bin loads. Fix arbitrary $i \in [n]$ and $k \in \mathbb{N}$. For a subset $S \subseteq [m]$ of size $k$, let $E_{S,i}$ denote the event that all balls in $S$ are placed in bin $i$. Then $\Pr[E_{S,i}] = n^{-k}$. It follows that:

$$\Pr[\max_{i\in[n]} L_i \ge k] = \Pr\left[\bigcup_{\substack{S\subseteq[m], i\in[n] \\ |S|=k}} E_{S,i}\right] \overset{\text{UB}}{\le} \sum_{\substack{S\subseteq[m], i\in[n] \\ |S|=k}} \Pr[E_{S,i}] = \binom{m}{k} \cdot n \cdot n^{-k} \le \frac{m^k}{k!} n^{-k+1} = \frac{\lambda^k n}{k!}.$$

Substituting $k = \log n$ and observing that $\lambda^k = \text{poly}(n)$ while $(\log n)! = n^{\Omega(\log \log n)}$ grows faster than any polynomial, completes the argument.

(ii) If $\max_{i \in [n]} L_i \leq \log n$, then

$$C_{\text{cap}} = \sum_{i \in [n]} \binom{L_i}{2}.$$

The latter is an alternative definition of the collision count $C$ (summing collisions within each bin).

(iii) $C_{\text{cap}}$ is a function of the $m$ independent random variables $X_1, \ldots, X_m$. Changing any single $X_j$ alters $C_{\text{cap}}$ by at most $\log n$. The result follows directly from McDiarmid's inequality.
**Remark:** The same strategy does not apply well to $C$ itself. In the extreme case where $X_1 = \cdots = X_{m-1} = 1$ and $X_m = 2$, changing $X_m$ to 1 induces $m - 1$ additional collisions.

(iv) We combine the previous two results:

$$\begin{aligned}
\Pr[C - \mathbb{E}[C] \geq n^{2/3}] &\leq \Pr[C \neq C_{\text{cap}} \vee C_{\text{cap}} - \mathbb{E}[C] \geq n^{2/3}] \\
&\overset{\text{UB}}{\leq} \Pr[C \neq C_{\text{cap}}] + \Pr[C_{\text{cap}} - \mathbb{E}[C] \geq n^{2/3}] \\
&\overset{C \geq C_{\text{cap}}}{\leq} \Pr[C \neq C_{\text{cap}}] + \Pr[C_{\text{cap}} - \mathbb{E}[C_{\text{cap}}] \geq n^{2/3}] \\
&\leq O(n^{-100}) + \exp(-2n^{4/3}/(m \log^2 n)) = O(n^{-100}).
\end{aligned}$$

(v) We use hashing with linear chaining at load factor $\alpha = 1$ and make the simple uniform hashing assumption. The distribution of keys to buckets matches the setting at hand with $\lambda = 1$. Inserting the $n$ keys takes $O(n)$ times and querying the keys takes $O(n + C)$ time because very collision between two keys $x \neq y$ increases either the query time of $x$ or the query time of $y$ by one step. From (i) we know that $\mathbb{E}[C] = O(n)$ and from (iv) we know that $\Pr[C \geq 2\mathbb{E}[C]] \leq O(n^{-100})$. This implies a running time of $O(n + 2\mathbb{E}[C]) = O(n)$, except with probability $O(n^{-100})$.

*The following exercise is not directly related to randomized algorithms, except that we used the bound in lecture. Hence, "Bonus".*

## Exercise 2 – Bonus: Approximations of $e$

You know that $e = \lim_{n \to \infty} (1 + \frac{1}{n})^n$ (this is even a common *definition* of $e$). Derive from this that the following two inequalities hold for all $n \in \mathbb{N}$:

$$\begin{aligned}
(1 + \tfrac{1}{n})^n \leq e \leq (1 + \tfrac{1}{n})^{n+1} \\
(1 - \tfrac{1}{n})^n \leq e^{-1} \leq (1 - \tfrac{1}{n})^{n-1}.
\end{aligned}$$

The following steps are suggested:

(i) Prove the left-hand inequalities.
**Hint:** Use again $1 + x \leq e^x$.

(ii) Show that the right-hand sides are monotonically decreasing in $n$.

(iii) Deduce the right-hand inequalities from (ii) and a limit argument.

## Solution 2

(i) We have:

$$(1 + \tfrac{1}{n})^n \leq (e^{1/n})^n = e$$
$$(1 - \tfrac{1}{n})^n \leq (e^{-1/n})^n = e^{-1}$$

(ii) First, taking logarithms of the hint for (i), we obtain:

$$\forall x \in (-1, \infty) : \ln(1 + x) \leq x$$

To show that $f(n) = (1 + \frac{1}{n})^{n+1}$ is monotonically decreasing in $n \in \mathbb{N}$, it suffices to show that $\ln(f(x)) = (x + 1)\ln(1 + \frac{1}{x})$ is monotonically decreasing in $x \in (0, \infty)$. Consider its derivative and show it is everywhere $\leq 0$:

$$(\ln(f(x)))' = \ln(1 + \tfrac{1}{x}) + \frac{x+1}{1+\frac{1}{x}} \cdot (-\tfrac{1}{x^2}) \leq \tfrac{1}{x} - \frac{x+1}{(x+1)x} = 0.$$

Analogously, for $g(x) = (1 - \frac{1}{n})^{n-1}$, we show the derivative of $\ln(g(x))$ is $\leq 0$:

$$(\ln(g(x)))' = ((n-1)\ln(1 - \tfrac{1}{n}))' = \ln(1 - \tfrac{1}{x}) + \frac{x-1}{1-\frac{1}{x}} \cdot \frac{1}{x^2} \leq -\tfrac{1}{x} + \frac{x-1}{(x-1)x} = 0.$$

(iii) We have by separating a factor:

$$\lim_{n\to\infty} (1 + \tfrac{1}{n})^{n+1} = \lim_{n\to\infty} (1 + \tfrac{1}{n})^n \cdot \lim_{n\to\infty} (1 + \tfrac{1}{n}) = e \cdot 1 = e.$$

The limit of $(1 - \frac{1}{n})^{n-1}$ can be computed as follows:

$$\lim_{n\to\infty} (1 - \tfrac{1}{n})^{n-1} = \lim_{n\to\infty} (\tfrac{n-1}{n})^{n-1} = \lim_{n\to\infty} \left(\frac{1}{\frac{n}{n-1}}\right)^{n-1} = \frac{1}{\lim_{n\to\infty}(\frac{n}{n-1})^{n-1}}$$
$$= \frac{1}{\lim_{n\to\infty}(1 + \frac{1}{n-1})^{n-1}} = \frac{1}{e} = e^{-1}.$$

Thus, the right-hand sides converge to $e$ and $e^{-1}$, respectively. By (ii), they converge "from above". Hence, the inequalities hold as claimed.

## Exercise 3 – Counting Bloom Filter (a.k.a.: Count-Min Sketch)

A Counting Bloom Filter is initially like a standard Bloom Filter: it manages $n$ keys in an array of size $m$, with load factor $\alpha := \frac{n}{m}$ and $k$ hash functions. The parameters are chosen as in lecture so that a standard Bloom Filter would have false-positive probability $\varepsilon$. The array now contains natural numbers instead of bits. In addition to insert and query, a delete operation is now available.

**Algorithm** insert($x$)**:**
- **for** $i \in [k]$ **do**
  - $A[h_i(x)]{+}{+}$

**Algorithm** delete($x$)**:**
- **for** $i \in [k]$ **do**
  - $A[h_i(x)]{-}{-}$

**Algorithm** query($x$)**:**
- **for** $i \in [k]$ **do**
  - **if** $A[h_i(x)] = 0$ **then**
    - **return** false
- **return** true

We allow a key to be inserted *multiple times* into the Counting Bloom Filter. Hence, the managed object $S$ is now a *multiset* with $n$ elements $\{x_1, \ldots, x_n\}$, each with multiplicity $a_1, \ldots, a_n$. The operation insert($x$) adds one copy of $x$ to the multiset $S$, i.e., increments the multiplicity of $x$ if $x$ is already in $S$, or adds a new element with multiplicity 1 if $x$ is not yet in $S$. The meaning of delete is analogous. As before, query may return false positives but not false negatives.

(a) We require that delete is called only for elements that are actually in $S$ (with multiplicity at least 1). What breaks if we do not enforce this?

Additional useful operations can be implemented with Counting Bloom Filters.

(b) Implement an operation count that, for $x \in D$, returns an estimate count($x$) of the multiplicity $a$ of $x$ in $S$. Show that $\Pr[a \neq \text{count}(x)] \leq \varepsilon$.

(c) The primary argument for using (Counting) Bloom Filters is their low memory footprint compared to exact data structures. We should therefore avoid using large integer types (e.g., 64 bits) for counters. Consider an application where "most" counters never exceed 8 bits. We therefore use an array $A$ of 8-bit counters. Briefly discuss the following proposals for handling counter overflows. What are the advantages and what compromises are made?

- Alice merely prevents counter overflows, i.e., adapts the $A[h_i(x)]{+}{+}$ operation in insert and the $A[h_i(x)]{-}{-}$ operation in delete so that the counter is incremented/decremented only if the maximum/minimum representable value has not yet been reached.
- Bob proposes to "freeze" a counter that reaches $(11111111)_2 = 255$, i.e., no future insert or delete operations will modify this counter.
- Carol proposes to use the bit string $(11111111)_2 = 255$ as a marker indicating that the true counter value exceeds 254. In this special case, the true counter value is stored in a hash table.

## Solution 3

(a) A delete($y$) for a $y$ that was never inserted reduces $k$ counters in the Counting Bloom Filter uncontrollably. Thus, counters may become 0. Consequently, for some $x \in S$, query($x$) might return false, yielding a false negative — which is not allowed.

(b) We return the minimum of the counters associated with $x$:

**Algorithm** count($x$)**:**
- $r \leftarrow \infty$
- **for** $i \in [k]$ **do**
  - $r \leftarrow \min(r, A[h_i(x)])$
- **return** $r$

Note that $\text{count}(x) < a$ is impossible, since every occurrence of $x$ is counted by every associated counter. However, it is possible that $\text{count}(x) > a$ if all $k$ counters are also incremented by other keys. To understand this, let $A'[1..m] \in \{0,1\}^m$ be the standard Bloom Filter into which all elements of $S$ except $x$ have been inserted (using the same hash functions as for the Counting Bloom Filter). Then:

$$\begin{aligned}
\text{count}(x) \neq a &\Leftrightarrow \text{count}(x) > a \\
&\Leftrightarrow \min_{i \in [k]} A[h_i(x)] > a \\
&\Leftrightarrow \forall i \in [k] : A[h_i(x)] > a \\
&\Leftrightarrow \forall i \in [k] : \text{some key in } S \text{ other than } x \text{ uses } A[h_i(x)] \\
&\Leftrightarrow \forall i \in [k] : A'[h_i(x)] = 1 \\
&\Leftrightarrow x \text{ is a false positive for } A'
\end{aligned}$$

The latter event has probability at most $\varepsilon$ by the choice of configuration parameters. Hence, the same holds for the equivalent first event.

(c) Regarding the proposals:

- Alice's proposal is simple to implement, but false negatives become possible. The simplest example is inserting the same key $x$ 256 times and then deleting it 255 times. The final insertion is lost, and the deletions reset all relevant counters to 0. Subsequently, query($x$) returns false, although $x$ should still be in the data structure. Not good.
- Bob's proposal ensures counters never falsely return to zero, so false negatives are impossible. However, frozen counters can never be released. In the long run, the false-positive rate may increase.
- Carol's proposal makes no compromises in functionality. The additional hash table naturally consumes space, and accesses to it incur time costs.

## Exercise 4 – Estimating Set Intersections with Bloom Filters

Let $n, m, k \in \mathbb{N}$. Alice and Bob wish to estimate how similar their musical tastes are. Let $X$ be Alice's $n$ favorite songs and $Y$ be Bob's $n$ favorite songs. The goal is to estimate $\gamma := \frac{|X \cap Y|}{n} \in [0, 1]$. They proceed as follows:

- Alice constructs a Bloom filter $A[1..m] \in \{0,1\}^m$ for $X$ using $k$ hash functions $h_1, \ldots, h_k$.
- Bob constructs a Bloom filter $B[1..m] \in \{0,1\}^m$ for $Y$ using the *same* $k$ hash functions.
- Alice and Bob exchange their filters and compute $\delta := \frac{|\{i \in [m] | A[i] \neq B[i]\}|}{m}$.
- Alice and Bob compute an estimate $\bar{\gamma}$ for $\gamma$ based on $\delta$.

Solve the following subproblems:

(a) Discuss: What advantages and disadvantages might this method have compared to direct exchange of $X$ and $Y$?

(b) Gain intuition: What values of $\delta$ do you expect (approximately) for the extreme cases $\gamma = 1$ and $\gamma = 0$?
**Hint:** You may assume here and in the following that the Bloom filters use an "optimal" configuration with $\alpha k = \ln(2)$.

(c) Compute $\mathbb{E}[\delta]$ as a function of $\gamma$. You may drop lower-order terms, e.g., write $(1 - \frac{1}{m})^m \approx e^{-1}$ without carrying an $o(1)$ term.
**Hint:** At first glance, other parameters (e.g., $n, m, k, \alpha, \varepsilon$) might appear relevant. Their influence vanishes in lower-order terms.

(d) Discuss: Which concentration bound is suitable for proving that $\delta$ is close to $\mathbb{E}[\delta]$ with high probability?

(e) Rearrange the equation from (c) to show how an estimate $\bar{\gamma}$ for $\gamma$ can be computed from $\delta$.

(f) Speculate: What role does the choice of $k$ (or $\varepsilon$) play in this context?

## Solution 4

(a)
- Advantage: Space usage may be smaller.
- Disadvantage: We can ensure $\bar{\gamma}$ is close to $\gamma$ with high probability, but cannot compute $\gamma$ error-free.
- Advantage: Elements of $X$ and $Y$ are not revealed; i.e., Alice can plausibly deny any $x \in X$ without being contradicted.

(b) For $\gamma = 1$, clearly $A[i] = B[i]$ for all $i$, so $\delta = 0$. For $\gamma = 0$, the two Bloom filters are independent. By lecture, in a Bloom filter with $\alpha k = \ln(2)$, approximately half the entries are 1 and half are 0. Thus, $\Pr[A[i] \neq B[i]] \approx \Pr_{C,D \sim \mathrm{Ber}(1/2)}[C \neq D] = 1/2$, so we expect $\delta \approx 1/2$.

(c) We write $h(z) := \{h_1(z), \ldots, h_k(z)\}$. Note: Events concerning different keys or hash functions can be separated by independence. Let $x_0$ be an arbitrary element in $X$ and $y_0$ an arbitrary element in $Y \setminus X$.

$$\begin{aligned}
\mathbb{E}[\delta] &= \frac{\mathbb{E}[|\{i \in [m] \mid A[i] \neq B[i]\}|]}{m} = \frac{1}{m}\sum_{i=1}^{m} \Pr[A[i] \neq B[i]] = \Pr[A[i_0] \neq B[i_0]] \\
&= \Pr[(A[i_0] = 0 \wedge B[i_0] = 1) \vee (A[i_0] = 1 \wedge B[i_0] = 0)] = 2\Pr[A[i_0] = 0 \wedge B[i_0] = 1] \\
&= 2\Pr[\forall x \in X : i_0 \notin h(x) \wedge \exists y \in Y \setminus X : i_0 \in h(y)] \\
&= 2\Pr[\forall x \in X : i_0 \notin h(x)] \cdot \Pr[\exists y \in Y \setminus X : i_0 \in h(y)] \\
&= 2\Pr[\forall x \in X : i_0 \notin h(x)] \cdot (1 - \Pr[\forall y \in Y \setminus X : i_0 \notin h(y)]) \\
&= 2\Pr[i_0 \notin h(x_0)]^{|X|} \cdot (1 - \Pr[i_0 \notin h(y_0)]^{|Y \setminus X|}) \\
&= 2\Pr[i_0 \neq h_1(x_0)]^{k|X|} \cdot (1 - \Pr[i_0 \neq h_1(y_0)]^{k|Y \setminus X|}) \\
&= 2(1 - \tfrac{1}{m})^{k|X|} \cdot (1 - (1 - \tfrac{1}{m})^{k|Y \setminus X|}) \\
&= 2(1 - \tfrac{1}{m})^{kn} \cdot (1 - (1 - \tfrac{1}{m})^{k(1-\gamma)n}) \\
&= 2(1 - \tfrac{1}{m})^{k\alpha m} \cdot (1 - (1 - \tfrac{1}{m})^{k(1-\gamma)\alpha m}) \\
&\approx 2e^{-k\alpha} \cdot (1 - e^{-k(1-\gamma)\alpha}) \\
&= 2e^{-\ln(2)} \cdot (1 - e^{-(1-\gamma)\ln(2)}) \\
&= 1 - \left(\tfrac{1}{2}\right)^{(1-\gamma)}.
\end{aligned}$$

(d) The method of bounded differences (or McDiarmid's inequality) is suitable here. The $k \cdot |X \cup Y|$ relevant hash values are all independent. Changing any one alters $\delta$ by at most $\pm\frac{1}{m}$. The lecture's analysis of the concentration of $Z$ (number of zeros) transfers directly.

(e) From (c), we have $\mathbb{E}[\delta] = 1 - (\frac{1}{2})^{(1-\gamma)}$. We drop the "$\mathbb{E}$" (since we have only $\delta$, not $\mathbb{E}[\delta]$) and replace $\gamma$ with $\bar{\gamma}$ (since we cannot compute $\gamma$ exactly but only estimate it). Solving $\delta = 1 - (\frac{1}{2})^{(1-\bar{\gamma})}$ yields: $\bar{\gamma} = 1 - \log_2(1/(1-\delta))$.

(f) The larger $k$, the stronger the concentration bound and the better the estimate $\bar{\gamma}$. However, there is little reason not to use $k = 1$. It is even conceivable to choose $k \in (0, 1)$, meaning a key is assigned a position with probability $k$ and discarded with probability $1 - k$. These random choices must be made via a hash function known to both Alice and Bob. With $k = \Theta(1/n)$, one could achieve memory usage independent of $n$. Similar to approximation algorithms, one could introduce relative error and failure probability and compute the required $k$ as a function of these parameters.

Karlsruhe Institute of Technology

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 9 – Coupling, Balls into Bins and Poissonisation

## Probability and Computing

## Exercise 1 – Coupling of a Random Walk

Let $X_1, X_2, \ldots \sim \mathcal{U}(\{-1, 1\})$ be independent random variables. For $n \in \mathbb{N}_0$, define $W_n := \sum_{i=1}^{n} X_i$. The sequence $(W_n)_{n\in\mathbb{N}_0}$ is called a random walk. We may also consider a shifted random walk $(V_n)_{n\in\mathbb{N}_0}$ defined by $V_n := W_n + 42$, which therefore has initial position $V_0 = 42$ instead of $W_0 = 0$. We aim to show that the choice of initial position typically does not matter in the long run. We will use without proof that the random walk visits every integer at least once with probability 1. In particular, $\lim_{n\to\infty} \Pr[\max\{W_1, \ldots, W_n\} < c] = 0$ for all $c \in \mathbb{N}$.

(i) Let $S_1, S_2, \ldots \subseteq \mathbb{Z}$ be arbitrary sets. Show that $\lim_{n\to\infty} |\Pr[W_n \in S_n] - \Pr[V_n \in S_n]| = 0$.
**Hint:** Construct a coupling $(W'_n, V'_n)_{n\in\mathbb{N}_0}$ of $(W_n)_{n\in\mathbb{N}_0}$ and $(V_n)_{n\in\mathbb{N}_0}$ such that $\lim_{n\to\infty} \Pr[W'_n = V'_n] = 1$.

(ii) Show that the result of part (i) does not hold in this form for a shift of 43 instead of 42.

## Solution 1

(i) We take $(W'_n) = (W_n)$ and describe $(V'_n)$ in natural language. Initially, $(V'_n)$ behaves exactly oppositely to $(W_n)$, i.e., uses the inverted increments $-X_1, -X_2, -X_3, \ldots$ and so on. Let $T = \min\{t \in \mathbb{N} \mid W_t = 21\}$. Then $W_T = 21$ and $V'_T = 42 - 21 = 21$, meaning the random walks meet at time $T$. From this point onward, $(V'_n)$ behaves identically to $(W_n)$, using the same increments $X_{T+1}, X_{T+2}, \ldots$.

It is evident that $(V'_n)_{n\in\mathbb{N}_0} \overset{\mathrm{d}}{=} (V_n)_{n\in\mathbb{N}_0}$, since the accumulated increments remain independent random variables uniformly distributed in $\mathcal{U}(\{-1, 1\})$ (whether we add or subtract $X_i$ is determined before observing the value of $X_i$). Thus, we have a valid coupling. In this coupling, the implication $W_n \neq V'_n \Rightarrow T \geq n$ holds. We now perform an auxiliary calculation for arbitrary random variables $X$, $Y$ and arbitrary sets $S$.

$$
\begin{aligned}
&|\Pr[X \in S] - \Pr[Y \in S]| \\
&= \left|\Pr[X \in S \wedge X \neq Y] + \Pr[X \in S \wedge X = Y] - \Pr[Y \in S \wedge X = Y] - \Pr[Y \in S \wedge X \neq Y]\right| \\
&= \left|\Pr[X \in S \wedge X \neq Y] - \Pr[Y \in S \wedge X \neq Y]\right| \\
&\leq \max\{\Pr[X \in S \wedge X \neq Y], \Pr[Y \in S \wedge X \neq Y]\} \leq \Pr[X \neq Y].
\end{aligned}
$$

Applying this to $S = S_n$, $X = W_n$, and $Y = V'_n$, we obtain:

$$\begin{aligned}|\Pr[W_n \in S_n] - \Pr[V_n \in S_n]| &= |\Pr[W_n \in S_n] - \Pr[V'_n \in S_n]| \le \Pr[W_n \neq V'_n] \\ &= \Pr[T > n] = \Pr[\max\{W_1, \dots, W_n\} < 21] \xrightarrow{n\to\infty} 0\end{aligned}$$

where the last step applies the hint for $c = 21$.

(ii) As defined, the random walk "forgets" its precise starting point but not the parity of this starting point. In other words, if we define $S_n := S := 2 \cdot \mathbb{Z}$, then random walks alternate between being in $S$ and not in $S$. For a shift of 23, we would then have $|\Pr[W_n \in S_n] - \Pr[V_n \in S_n]| = 1$ for all $n \in \mathbb{N}$.

## Exercise 2 – Coupling and Total Variation Distance

Let $X$ and $Y$ be two random variables taking values in $\mathbb{N}$. The total variation distance between $X$ and $Y$ (or their distributions) is defined as[1]

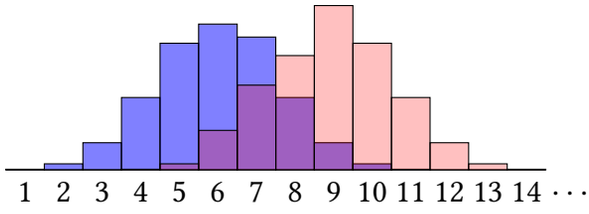
$$d(X, Y) = \frac{1}{2} \sum_{i \in \mathbb{N}} |\Pr[X = i] - \Pr[Y = i]|.$$

(i) Show: There exists a coupling $(X', Y')$ of $X$ and $Y$ such that $\Pr[X' \neq Y'] = d(X, Y)$.

(ii) Show: No coupling $(X', Y')$ of $X$ and $Y$ satisfies $\Pr[X' \neq Y'] < d(X, Y)$.

## Solution 2

As preparation, consider a joint histogram of $X$ (blue) and $Y$ (red).




Let all bars have width 1. Denote by red, blue, and purple the sets of points of the respective colors, and by $A_{\text{red}}$, $A_{\text{blue}}$, and $A_{\text{purple}}$ the corresponding areas. Since the bars represent distributions, we have $A_{\text{blue}} + A_{\text{purple}} = 1$ and $A_{\text{red}} + A_{\text{purple}} = 1$, implying $A_{\text{blue}} = A_{\text{red}}$. Both equal the total variation distance $d(X, Y)$. This is seen as follows:

$$\begin{aligned}d(X, Y) &= \tfrac{1}{2} \sum_{i \in \mathbb{N}} |\Pr[X = i] - \Pr[Y = i]| \\ &= \tfrac{1}{2} \Bigg( \sum_{\substack{i \in \mathbb{N} \\ \Pr[X=i] \ge \Pr[Y=i]}} (\Pr[X = i] - \Pr[Y = i]) + \sum_{\substack{i \in \mathbb{N} \\ \Pr[X=i] < \Pr[Y=i]}} (\Pr[Y = i] - \Pr[X = i]) \Bigg) \\ &= \tfrac{1}{2} \big( A_{\text{blue}} + A_{\text{red}} \big) = \tfrac{1}{2} \big( A_{\text{blue}} + A_{\text{blue}} \big) = A_{\text{blue}}.\end{aligned}$$

[1] A general definition applicable also to continuous probability spaces can be found on Wikipedia.

(i) We sample a pair $(P, Q)$ of points as follows:

- Sample $P \sim \mathcal{U}(\text{blue} \cup \text{purple})$.
- If $P \in \text{purple}$, set $Q = P$.
- Otherwise, sample $Q \sim \mathcal{U}(\text{red})$.

It should be clear that then $Q \sim \mathcal{U}(\text{red} \cup \text{purple})$. We now define $X'$ as the index of the bar containing $P$ and $Y'$ as the index of the bar containing $Q$. It should then be clear that $X' \overset{\text{d}}{=} X$ and $Y' \overset{\text{d}}{=} Y$. The useful property we will use is $\Pr[X' = Y'] = \Pr[P = Q] = A_{\text{purple}}$. From this it follows as desired:

$$\Pr[X' \neq Y'] = 1 - A_{\text{purple}} = A_{\text{blue}} = d(X, Y).$$

(ii) Let $S = \{i \in \mathbb{N} \mid \Pr[X = i] > \Pr[Y = i]\}$. Let $(X', Y')$ be any coupling of $X$ and $Y$. Then:

$$\begin{aligned}\Pr[X' \neq Y'] &\geq \Pr[X' \in S \wedge Y' \notin S] = \Pr[X' \in S] - \Pr[X' \in S \wedge Y' \in S] \\ &\geq \Pr[X' \in S] - \Pr[Y' \in S] = \Pr[X \in S] - \Pr[Y \in S] \\ &= \sum_{i \in S} \Pr[X = i] - \Pr[Y = i] = A_{\text{blue}} = d(X, Y).\end{aligned}$$

## Exercise 3 – Properties of the Poisson Distribution

Let $X \sim \text{Pois}(\lambda)$. Show:

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

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

(iii) For $Y \sim \text{Pois}(\rho)$ independent of $X$, we have $X + Y \sim \text{Pois}(\lambda + \rho)$.

(iv) For $X' \sim \text{Bin}(X, p)$, we have $X' \sim \text{Pois}(\lambda p)$.
**Note:** Here, a two-stage random experiment is performed. The outcome $X$ of the first stage serves as a parameter of the second stage.

## Solution 3

In the following, we constantly use the definition of the exponential function, i.e., $e^t = \sum_{i=0}^{\infty} \frac{t^i}{i!}$.

(i) $\mathbb{E}[X] = \sum_{i=0}^{\infty} e^{-\lambda} \frac{\lambda^i}{i!} \cdot i = e^{-\lambda} \cdot \lambda \sum_{i=1}^{\infty} \frac{\lambda^{i-1}}{(i-1)!} = e^{-\lambda} \cdot \lambda \sum_{i=0}^{\infty} \frac{\lambda^i}{i!} = e^{-\lambda} \cdot \lambda \cdot e^{\lambda} = \lambda.$

(ii) We first compute the second *un*centered moment:

$$\begin{aligned}
\mathbb{E}[X^2] &= \sum_{i=0}^{\infty} e^{-\lambda}\frac{\lambda^i}{i!}\cdot i^2 = e^{-\lambda}\cdot\lambda\sum_{i=1}^{\infty}\frac{\lambda^{i-1}}{(i-1)!}\cdot i \\
&= e^{-\lambda}\cdot\lambda\Big(\sum_{i=1}^{\infty}\frac{\lambda^{i-1}}{(i-1)!}\cdot(i-1)+\sum_{i=1}^{\infty}\frac{\lambda^{i-1}}{(i-1)!}\Big) \\
&= e^{-\lambda}\cdot\lambda\Big(\lambda\sum_{i=2}^{\infty}\frac{\lambda^{i-2}}{(i-2)!}+\sum_{i=1}^{\infty}\frac{\lambda^{i-1}}{(i-1)!}\Big) \\
&= e^{-\lambda}\cdot\lambda\Big(\lambda\sum_{i=0}^{\infty}\frac{\lambda^{i}}{i!}+\sum_{i=0}^{\infty}\frac{\lambda^{i}}{i!}\Big) \\
&= e^{-\lambda}\cdot\lambda\Big(\lambda e^{\lambda}+e^{\lambda}\Big) = \lambda^2+\lambda
\end{aligned}$$

Moreover, we know $\mathbb{E}[X]^2 = \lambda^2$. It follows that

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = \lambda^2+\lambda-\lambda^2 = \lambda.$$

(iii) Let $k \in \mathbb{N}$. We consider all $k+1$ possibilities by which $X+Y$ can sum to $k$, and then apply the binomial theorem.

$$\begin{aligned}
\Pr[X+Y=k] &= \sum_{i=0}^{k}\Pr[X=i\wedge Y=k-i] = \sum_{i=0}^{k}\Pr[X=i]\Pr[Y=k-i] \\
&= \sum_{i=0}^{k} e^{-\lambda}\frac{\lambda^i}{i!}e^{-\rho}\frac{\rho^{k-i}}{(k-i)!} = e^{-(\lambda+\rho)}\frac{1}{k!}\sum_{i=0}^{k}\frac{k!}{i!(k-i)!}\lambda^i\rho^{k-i} \\
&= e^{-(\lambda+\rho)}\frac{1}{k!}\sum_{i=0}^{k}\binom{k}{i}\lambda^i\rho^{k-i} = e^{-(\lambda+\rho)}\frac{(\lambda+\rho)^k}{k!} = \Pr_{Z\sim\mathrm{Pois}(\lambda+\rho)}[Z=k].
\end{aligned}$$

(iv) Let $k \in \mathbb{N}$. For the final outcome to be $k$, it must have held that $X \geq k$. We consider all possibilities.

$$\begin{aligned}
\Pr[X'=k] &= \sum_{i\geq k}\Pr[X=i\wedge X'=k] = \sum_{i\geq k}\Pr[X=i]\cdot\Pr[X'=k\mid X=i] \\
&= \sum_{i\geq k} e^{-\lambda}\frac{\lambda^i}{i!}\cdot\binom{i}{k}p^k(1-p)^{i-k} = e^{-\lambda}\cdot\sum_{i\geq k}\frac{\lambda^i}{k!(i-k)!}p^k(1-p)^{i-k} \\
&= e^{-\lambda}\frac{(\lambda p)^k}{k!}\cdot\sum_{i\geq k}\frac{\lambda^{i-k}(1-p)^{i-k}}{(i-k)!} = e^{-\lambda}\frac{(\lambda p)^k}{k!}\cdot\sum_{i\geq 0}\frac{(\lambda(1-p))^i}{i!} \\
&= e^{-\lambda}\frac{(\lambda p)^k}{k!}e^{\lambda(1-p)} = e^{-\lambda p}\frac{(\lambda p)^k}{k!} = \Pr_{Z\sim\mathrm{Pois}(\lambda p)}[Z=k].
\end{aligned}$$

## Exercise 4 – Poissonised Bloom Filters

We consider a Poisson model of Bloom filters, i.e., we assume that each position in the array independently appears as a hash value $\mathrm{Pois}(\alpha k)$-many times.

(i) We again choose $\alpha k = \ln 2$. How can we show that the fraction $\frac{Z}{m}$ of zeros is with high probability close to $\frac{1}{2}$?

(ii) How could this result be transferred to a non-Poissonised model?

## Solution 4

(i) If $X \sim \mathrm{Pois}(\ln 2)$, then $\Pr[X = 0] = e^{-\ln 2} = \frac{1}{2}$. Since each position is now independently empty or non-empty, we have $Z \sim \mathrm{Bin}(m, \frac{1}{2})$. It follows that $\mathbb{E}[\frac{Z}{m}] = \frac{1}{2}$, and Chernoff bounds apply directly to $Z$.

(ii) The quantity $m - Z$ is a monotone function in the sense of the Poissonisation theorem from the lecture. Accordingly, the exact "$nk$ balls into $m$ bins" model can be sandwiched between two Poissonised models, as discussed.

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 10
# Approximation Algorithms
## Probability and Computing

## Exercise 1 – Jensen's Inequality

Let $D \subseteq \mathbb{R}$ be a connected domain and $f : D \to \mathbb{R}$ be a function. The function $f$ is called convex if it is "curved to the left" and concave if it is "curved to the right".[1] A function is convex if and only if its negation is concave. For a formal definition see: Wikipedia

(a) Decide (without proof) for the following functions whether they are convex on their respective domains, concave, both, or neither.

$$f_1(x) = x, \qquad f_2(x) = x^2, \qquad f_3(x) = x^3, \qquad f_4(x) = \log(x), \qquad f_5(x) = \log^2(x).$$

(b) Let $f$ be a convex function with domain $D$. Argue geometrically that for every $x_0 \in D$ there exists a linear function $g$ such that:

(i) $f(x) \geq g(x)$ for all $x \in D$

(ii) $f(x_0) = g(x_0)$.

(c) Conclude that for every convex function $f$ and for every random variable $X$ with values in the domain $D$ of $f$ the following holds:

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]).$$

**Hint:** Consider $x_0 = \mathbb{E}[X]$ and the corresponding $g$ from the previous subproblem.

(d) Show that analogously, for every concave function $f$ with domain $D$ and for every random variable $X$ with values in $D$ the following holds:

$$\mathbb{E}[f(X)] \leq f(\mathbb{E}[X]).$$

The inequality from (c) as well as variants as in (d) are called Jensen's inequality.

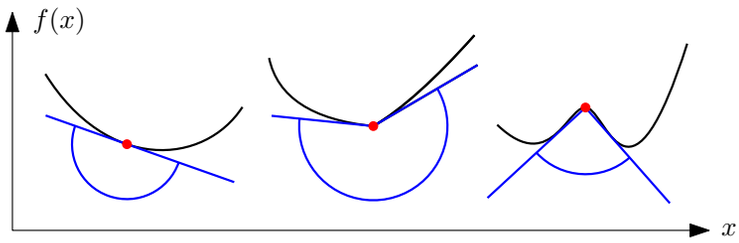
[1] The "curved to the left" in quotation marks allows, besides left curvatures (of a twice continuously differentiable function), also left kinks and linear behavior.

## Solution 1

(a) The functions are all twice continuously differentiable. If the second derivative is everywhere non-negative, then the function is convex; if it is everywhere non-positive, then the function is concave.

- $f_1$ is convex and concave
- $f_2$ is convex
- $f_3$ is neither convex nor concave
- $f_4$ is concave
- $f_5$ is neither convex nor concave

(b) Because $f$ is curved to the left, one can place a tangent $g$ to the graph of $f$ at the point $(x_0, f(x_0))$ such that $f$ lies entirely above $g$.




That this is possible can be seen as follows (illustrated in the figure): One considers, below the point $(x_0, f(x_0))$ (red) on the graph of $f$ (black), the angular region of those directions that never go beyond the graph of the function (blue). If this region is smaller than 180° (right in the figure), then one obtains a contradiction to the convexity of $f$. If this region is larger than 180° (center) or equal to 180° (left), then there exists at least one line through $(x_0, f(x_0))$ that avoids going beyond the graph of $f$.

(c) Let $g(x)$ be the function from (b) for $x_0 = \mathbb{E}[X]$. Then $f(x) \geq g(x)$ for all $x \in D$ and $f(\mathbb{E}[X]) = g(\mathbb{E}[X])$. Since $g$ is a line, there exist $a, b \in \mathbb{R}$ such that $g(x) = ax + b$. It follows that

$$\mathbb{E}[f(X)] \geq \mathbb{E}[g(X)] = \mathbb{E}[aX + b] = a\mathbb{E}[X] + b = g(\mathbb{E}[X]) = f(\mathbb{E}[X]).$$

(d) Since $-f$ is convex, it follows directly from (c):

$$\mathbb{E}[f(X)] = -\mathbb{E}[-f(X)] \overset{(c)}{\leq} -(-f(\mathbb{E}[X])) = f(\mathbb{E}[X]).$$

## Exercise 2 – Analysis of Lossy Counting

Reminder: Lossy Counting is a simple streaming algorithm that approximately counts the length $m$ of a stream. It involves a parameter $p \in (0, 1]$. The algorithm itself as well as the way it is used are shown on the right. Prove:

(a) $\mathbb{E}[\text{result}] = m$

(b) $\Pr[|\text{result} - m| \leq \varepsilon m] \geq 1 - 2\exp(-\varepsilon^2 pm/3)$.

(c) $\mathbb{E}[\text{space}] \leq \log(1 + mp) + 1$.
**Hint:** By space we denote the maximum memory usage required for the state $Z$ of LossyCounting. A number $i \in \mathbb{N}$ can be encoded with $\lceil \log_2(i+1) \rceil$ bits. Use Jensen's inequality from Exercise 1.

**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** $Z/p$

Usage:
  $Z \leftarrow \text{init}()$
  **for** $i = 1$ **to** $m$ **do**
    $Z \leftarrow \text{update}(Z, a_i)$
  **return** $\text{result}(Z)$

## Solution 2

(a) Let $X_1, \ldots, X_m \sim \text{Ber}(p)$ be independent random variables, where $X_i$ indicates whether the $i$-th element of the stream leads to an increment of the counter $Z$. Then $X := \sum_{i=1}^{m} X_i$ is the value of $Z$ after the last update. The estimate of the algorithm for $m$ is thus $\text{result} = X/p$. Hence:

$$\mathbb{E}[\text{result}] = \mathbb{E}[X/p] = \tfrac{1}{p}\mathbb{E}\Big[\sum_{i=1}^{m} X_i\Big] = \tfrac{1}{p}\sum_{i=1}^{m}\mathbb{E}[X_i] = \tfrac{1}{p}\sum_{i=1}^{m} p = m.$$

(b) Using the stated Chernoff bound, we obtain

$$\begin{aligned}\Pr[|\text{result} - m| \geq \varepsilon m] &= \Pr[|X/p - m| \geq \varepsilon m] = \Pr[|X - mp| \geq \varepsilon mp] \\ &= \Pr[|X - \mathbb{E}[X]| \geq \varepsilon\mathbb{E}[X]] \leq 2\exp(-\varepsilon^2\mathbb{E}[X]/3) = 2\exp(-\varepsilon^2 mp/3).\end{aligned}$$

The claim follows by considering the complementary probability.

(c) Since $Z$ grows monotonically, the memory requirement for $Z$ is largest at the very end, namely $\lceil \log_2(1 + X) \rceil$. Since $f(x) = \log(1 + x)$ is concave on $[0, \infty)$, Jensen's inequality yields

$$\begin{aligned}\mathbb{E}[\text{space}] &= \mathbb{E}[\lceil \log_2(1 + X) \rceil] \leq \mathbb{E}[\log_2(1 + X)] + 1 \\ &\overset{\text{Jensen}}{\leq} \log_2(1 + \mathbb{E}[X]) + 1 = \log_2(1 + mp) + 1.\end{aligned}$$

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 11
# Game Theory & Yao's Principle

## Probability and Computing

## Exercise 1 – Lower bounds for randomized sorting

Let $n \in \mathbb{N}$ and **Inputs** be the set of all permutations of $\{1, \dots, n\}$. Let **Algos** be the set of all comparison-based *deterministic* sorting algorithms.

(i) Give an $A \in$ **Algos** (without proof) with $\max_{I \in \textbf{Inputs}} C(A, I) = O(n \log n)$.

(ii) Warm-up: For $n = 3$ there exists a randomized algorithm $\mathcal{A}$ that beats every deterministic algorithm in the following sense:

$$\min_{A \in \textbf{Algos}} \max_{I \in \textbf{Inputs}} C(A, I) = 3 > 2 + \tfrac{2}{3} = \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$

Our plan in the following is to show that this advantage disappears in $O$-notation.

(iii) Show that there is no "hard input", that is, that the following holds:

$$\max_{I \in \textbf{Inputs}} \min_{A \in \textbf{Algos}} C(A, I) = n - 1.$$

In order to immediately focus on the best possible cost for an input *distribution*, we need some preparation:

(iv) Show that in a binary tree with $k$ leaves, the average depth of a leaf is at least $\lfloor \log_2(k) \rfloor$.
**Hint:** To do so, show that the average leaf depth is minimal for balanced trees; more precisely, that any tree in which the leaf depths differ by at least 2 can be rearranged such that the average leaf depth decreases while the number of leaves remains the same.

(v) Now conclude using Yao's principle that:

$$\min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \Omega(n \log n).$$

## Solution 1

(i) Mergesort.

(ii) We first show the left equality. Let $A \in$ **Algos** be arbitrary. On input $I = (x, y, z)$ we may assume, by symmetry (between the elements), that $A$ first compares $x$ and $y$ and finds $x < y$. By symmetry (between $<$ and $>$) we may assume that it next compares $x$ and $z$. Now it may be the case that $x < y$ and $x < z$ hold; then the order of $y$ and $z$ is still undetermined and another comparison is necessary. Hence there exists an input $I$ such that $C(A, I) = 3$. Therefore $\max_{I \in \textbf{Inputs}} C(A, I) = 3$. Since $A$ was arbitrary, this implies:

$$\min_{A \in \textbf{Algos}} \max_{I \in \textbf{Inputs}} C(A, I) = 3.$$

A randomized algorithm can "guess" which element is the middle element and compare the other two elements with it. If the algorithm guessed correctly, the complete order is determined after two comparisons. Otherwise, a third comparison is necessary. This $\mathcal{A}$ satisfies $\mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \frac{1}{3} \cdot 2 + \frac{2}{3} \cdot 3 = 2 + \frac{2}{3}$ for every input $I$, as desired.

(iii) The order of "min" and "max" allows us to tailor the algorithm to the input $I$. Let $\pi$ be the permutation[1] that sorts $I$. We define $A$ depending on $I$ as follows:

```
Algorithm A(I'):
    swap elements of I' according to π
    fail ← FALSE
    for i = 1 to n − 1 do // check whether sorted
        if I'[i] > I'[i + 1] then
            fail ← TRUE
            break
    if fail then
        MergeSort(I')
```

We have to consider three things:

**The algorithm is correct, i.e. $A \in$ Algos.** This is clear: The algorithm first reorders its input $I'$ according to $\pi$ and checks whether $I'$ is sorted as a result. If so, it does nothing further; otherwise, it uses MergeSort.

**The algorithm is fast for $I$.** If the input $I'$ coincides with $I$ (the input to which $A$ is tailored), then $I'$ is sorted after the swaps according to $\pi$. Then only the $n - 1$ comparisons of the loop are needed to verify this.

**It cannot be done any faster.** Suppose fewer than $n - 1$ comparisons were made. Consider the graph $G = ([n], E)$ that contains an edge between $i$ and $j$ if the $i$th element of the input was compared with the $j$th element of the input. Since

[1] By "permutation" one sometimes means an ordering of a set (as at the beginning of the exercise), sometimes an operator that reorders a given sequence (this is what is meant here).

$|E| < n-1$, $G$ is disconnected. Hence the relative order of these connected components is not determined. (Formally: If one input is consistent with the information of the decision tree, then so is another input. Thus the algorithm does not yet "know" what the input was, which is necessary in order to construct the output.)

(iv) Let $T$ be a binary tree with $k$ leaves and minimal average leaf depth. Then $T$ is a full binary tree, i.e. every vertex has 0 or 2 children (reason: internal vertices with only one child can be removed and the average leaf depth can be reduced). Let $L$ and $L'$ be the maximum and minimum depth of a leaf in $T$, respectively. We now show that $L' \geq L-1$. Suppose this is not the case; then $L' \leq L-2$. Let $\ell_1$ be a leaf at depth $L$ and let $\ell_2$ be its sibling (which exists because $T$ is full), which must also be a leaf (by choice of $L$ as the maximum depth). Let $\ell'$ be a leaf at depth $L'$. We now reattach $\ell_1$ and $\ell_2$ and make them children of $\ell'$. This again yields a binary tree. The sum of leaf depths changes for the following reasons:

- The leaves $\ell_1$ and $\ell_2$ now have depth $L'+1$ instead of $L$.
- The vertex $p$, which was the parent of $\ell_1$ and $\ell_2$, is now a leaf of depth $L-1$.
- The vertex $\ell'$ at depth $L'$ is no longer a leaf.

Thus the sum of leaf depths changes by

$$2((L'+1)-L)+(L-1)-L' = L'-L+1 \leq L-2-L+1 = -1,$$

i.e. it has decreased. Hence the average leaf depth has decreased, which contradicts the choice of $T$.

Therefore $L' \geq L-1$, i.e. the leaf depths differ by at most 1. If the average leaf depth of $T$ were less than $\lfloor \log_2(k) \rfloor$, then at least one leaf would have depth less than $\lfloor \log_2(k) \rfloor$ and all leaves would have depth at most $\lfloor \log_2(k) \rfloor$. Hence there would be fewer than $2^{\lfloor \log_2(k) \rfloor} \leq k$ leaves—a further contradiction.

Thus $T$ has average leaf depth at least $\lfloor \log_2(k) \rfloor$. Since $T$ was chosen to have minimal average leaf depth, this also holds for all other binary trees.

(v) Consider the decision tree $T_A$ describing an arbitrary $A \in \mathbf{Algos}$. Without loss of generality, we consider only algorithms that perform a comparison "$x < y$" only if both "true" and "false" are still possible outcomes, i.e. every computation path is followed by at least one input. This means that $T_A$ is a full binary tree. For each of the $n!$ possible inputs, a different permutation of the elements is required; hence each input must lead to a different leaf in $A$. Thus $A$ has exactly $n!$ leaves. By (iii), the average leaf depth of $A$ is at least $\lfloor \log_2(n!) \rfloor$. If we now consider the uniform distribution $\mathcal{I}$ on the inputs, then the expected number of comparisons that $A$ performs for $I \sim \mathcal{I}$ is exactly the average leaf depth of $T_A$, and hence also at least $\lfloor \log_2(n!) \rfloor$. From $n! \geq (n/2)^{n/2}$ it follows that $\lfloor \log_2(n!) \rfloor \geq \lfloor (n/2)\log_2(n/2) \rfloor = \Omega(n \log n)$. Thus, by Yao's theorem,

$$C \overset{\text{Yao}}{\geq} \min_{A \in \mathbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}}[C(A,I)] \geq \lfloor \log_2(n!) \rfloor = \Omega(n \log n).$$

## Exercise 2 – Yao's principle without game theory

Prove Yao's principle without resorting to game-theoretic theorems (no Nash theorem, Loomis theorem, etc.). That is, prove that in the setting of the lecture, for an arbitrary distribution $\mathcal{A}_0$ on **Algos** and an arbitrary distribution $\mathcal{I}_0$ on **Inputs**, the following holds:

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A,I)] \geq \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A,I)].$$

**Hint**: The game-theoretic framework of the lecture is not required here, because we do not wish to show that "=" is achievable.

## Solution 2

We have:

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A,I)] \geq \mathbb{E}_{A \sim \mathcal{A}_0, I \sim \mathcal{I}_0}[C(A,I)] \geq \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A,I)].$$

In words: In the middle, both $A$ and $I$ are chosen *at random*. On the left, $I$ is not chosen at random but with the goal of maximization, which makes the result larger. On the right, $A$ is not chosen at random but with the goal of minimization, which makes the result smaller. This already solves the exercise.

**Remark.** Formally, one could write the first step (and analogously the second) in more fine-grained detail by using the following two abstract observations:

1. $\max_{x \in X} f(x) \geq \mathbb{E}_{x \sim \mathcal{X}}[f(x)]$.
2. $\mathbb{E}_{x \sim \mathcal{X}}[\mathbb{E}_{y \sim \mathcal{Y}}[g(x,y)]] = \mathbb{E}_{x \sim \mathcal{X}, y \sim \mathcal{Y}}[g(x,y)]$.

where $f$ and $g$ are functions, $\mathcal{X}$ is a distribution on a set $X$, and $\mathcal{Y}$ is a distribution on a set $Y$.

The first equation allows one to upper bound a maximum by an expectation, and the second equation allows one to convert a two-stage random experiment and an "expected expectation" into a single-stage random experiment. Applying these equations with $X = \textbf{Inputs}$, $Y = \textbf{Algos}$, $\mathcal{X} = \mathcal{I}_0$, $\mathcal{Y} = \mathcal{A}_0$, $f(I) = \mathbb{E}_{A \sim \mathcal{A}_0}[C(A,I)]$, and $g(A,I) = C(A,I)$ yields the first inequality above.

## Exercise 3 – Recommendation: Simulating the Evolution of Teamwork

The YouTube channel *Primer* deals with evolutionary game theory. In the following video, among other things, all possible 2-player games with two pure strategies are classified according to how many and which types of Nash equilibria they possess. The video is entertaining and invites active thinking, but is only marginally relevant to the lecture.

https://www.youtube.com/watch?v=TZfh8hpJIxo

Karlsruhe Institute of Technology

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 12 – Probabilistic Method

## Probability and Computing

## Exercise 1 – A children's game

Alice and Bob play an asymmetric game on a sequence of fields $0, 1, 2, \ldots, n$. Initially, $k$ tokens are placed on field 0. Each round proceeds as follows.

1. Alice chooses two disjoint sets $T_1$ and $T_2$ of tokens.
2. Bob then chooses $i \in \{1, 2\}$.
3. The tokens from $T_i$ are removed.
4. The tokens from $T_{2-i}$ each move one field to the right.

Alice wins as soon as a token reaches field $n$. She loses as soon as she chooses two empty sets. Solve the following tasks.

(i) Give a strategy for Alice with which she wins for $k \geq 2^n$.

(ii) Use the probabilistic method to show that there is a winning strategy for Bob if $k < 2^n$.

(iii) Bonus: Construct a winning strategy for Bob (without the probabilistic method).

## Solution 1

(i) Without loss of generality let $k = 2^n$, since Alice can ignore additional tokens. Alice's strategy is to always split the remaining tokens evenly between $T_1$ and $T_2$. Then exactly half of the tokens will always move on, and the other half will be removed. A simple induction shows that after $0 \leq i \leq n$ rounds exactly $2^{n-i}$ tokens lie on field $i$. Thus Alice has won after $n$ rounds.

(ii) Bob plays uniformly at random. For Alice we consider an arbitrary strategy. We can now view the game from the perspective of a single token $t$. Each time $t$ is selected by Alice, it moves one step to the right with probability $1/2$ and is removed with probability $1/2$. We may assume without loss of generality that Alice never loses (i.e. chooses two empty sets) as long as tokens still exist, and continues playing even if she has already won. Then the probability that token $t$ ever reaches position $i \in \mathbb{N}$ is exactly $2^{-i}$. The

probability that $t$ ever reaches position $n$ is therefore $2^{-n}$. The expected number of tokens that reach position $n$ is thus $2^{-n} \cdot k$, which by assumption is $< 1$. Hence it is possible that no token reaches position $n$. Therefore the probability that Bob wins is positive. Thus Alice's strategy is not a winning strategy. Since Alice's strategy was arbitrary, there is no winning strategy for Alice. Hence there exists a winning strategy for Bob.

(iii) We imagine that a token lying on field $i$ has a value of $2^i$, that is, the value of a token doubles when it moves one field to the right. If Alice now chooses two sets $T_1$ and $T_2$ with values $w_1$ and $w_2$, then Bob should always remove the set of tokens with the larger value. Without loss of generality let this be $T_1$. He then allows the value of the tokens in $T_2$ to double. Since for $w_1 \geq w_2$ the inequality $2w_2 \leq w_1 + w_2$ holds, the total value cannot have increased. Since initially a value of $k$ is present and a win for Alice requires a value of at least $2^n$, Alice cannot win if $k < 2^n$ holds.

## Exercise 2 – Larger[1] independent sets

Let $G = (V, E)$ be a graph with $n$ vertices and $m$ edges. Show using the probabilistic method that $G$ contains an independent set of size $\sum_{v \in V} \frac{1}{\deg(v)+1}$.

**Hint:** Random permutation of the vertices.

## Solution 2

We randomly permute the vertices. Let

$$I = \{v \in V \mid v \text{ appears in the permutation before all of its neighbors}\}.$$

It should be clear that $I$ is an independent set. It is also clear that $v$ is included in $I$ with probability $\frac{1}{\deg(v)+1}$, since for this to happen $v$ must be the first among $\deg(v) + 1$ vertices in the random permutation. Thus

$$\mathbb{E}[|I|] = \mathbb{E}\Big[\sum_{v \in V} [v \in I]\Big] = \sum_{v \in V} \Pr[v \in I] = \sum_{v \in V} \frac{1}{\deg(v) + 1}.$$

By the expectation argument, in particular there exists an independent set of the required size.

[1] **Remark:** Let $d = \frac{2m}{n}$ be the average degree of the vertices. In the lecture we constructed an independent set of size $\frac{n}{2d}$. For the size $U$ of the independent set guaranteed by this exercise, the following holds using an inequality between arithmetic and harmonic means:

$$U = \sum_{v \in V} \frac{1}{\deg(v) + 1} = n \cdot \Big(\frac{1}{n} \sum_{v \in V} \frac{1}{\deg(v) + 1}\Big) \geq n\Big(\frac{1}{n} \sum_{v \in V} \deg(v)+1\Big)^{-1} = \frac{n}{d+1}.$$

This is larger than $\frac{n}{2d}$ for $d > 1$.[2]

[2] "*But what if $d < 1$ holds?*" Then the theorem from the lecture is not applicable at all.

## Exercise 3 – Independent rainbow sets again

Let $G = (V, E)$ be a graph with $|V| = kc$ vertices that are colored with $c$ colors, where each color appears exactly $k$ times. The maximum degree is $\Delta$. Show: If $k \geq 8\Delta$, then there exists an independent rainbow set.

## Solution 3

This problem is a simple generalization of the necklace analysis from the lecture.

We again choose the rainbow set $R$ by selecting one vertex uniformly at random from each color class. The goal is to show that $R$ is an independent set with positive probability.

For this we define a bad event $B_{\{u,v\}}$ for each edge $\{u, v\}$ of the graph, which states that both $u$ and $v$ are contained in $R$. Again we have $\Pr[B_{\{u,v\}}] \leq \frac{1}{k^2} =: p$.

Another event $B_{\{u',v'\}}$ can only interact with $B_{\{u,v\}}$ if $u'$ or $v'$ has a color that is also the color of $u$ or $v$. Hence there are at most $d = 2k\Delta - 2$ such events (2 relevant colors, $k$ relevant vertices each, $\Delta$ incident edges each, where $\{u, v\}$ itself is not counted from either $u$ or $v$).

Thus we have $4pd = 4\frac{1}{k^2}(2k\Delta - 2) < \frac{8\Delta}{k} \leq 1$. Therefore we can apply the Lovász Local Lemma.

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 13 – Random Graphs

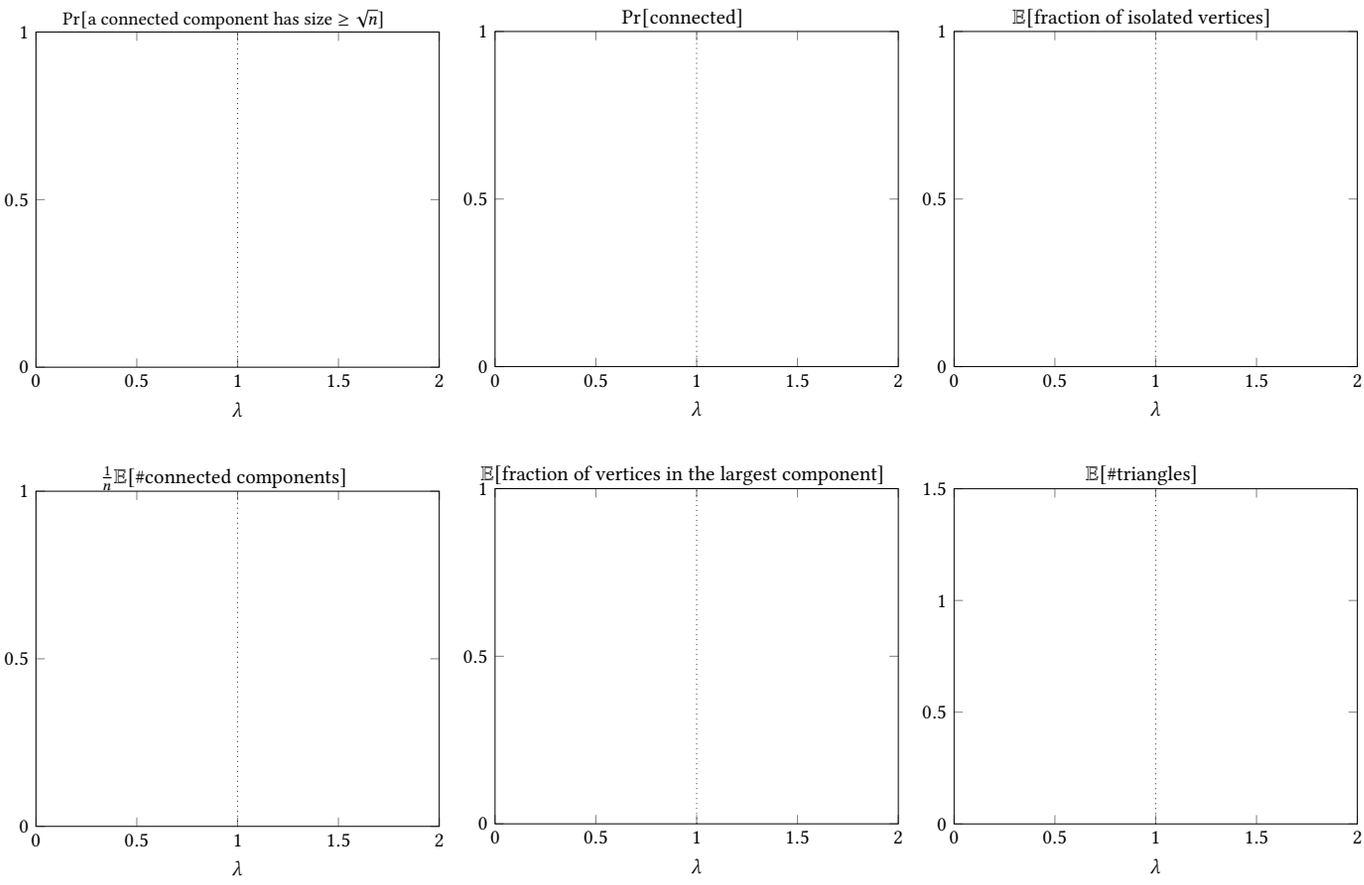
## Probability and Computing

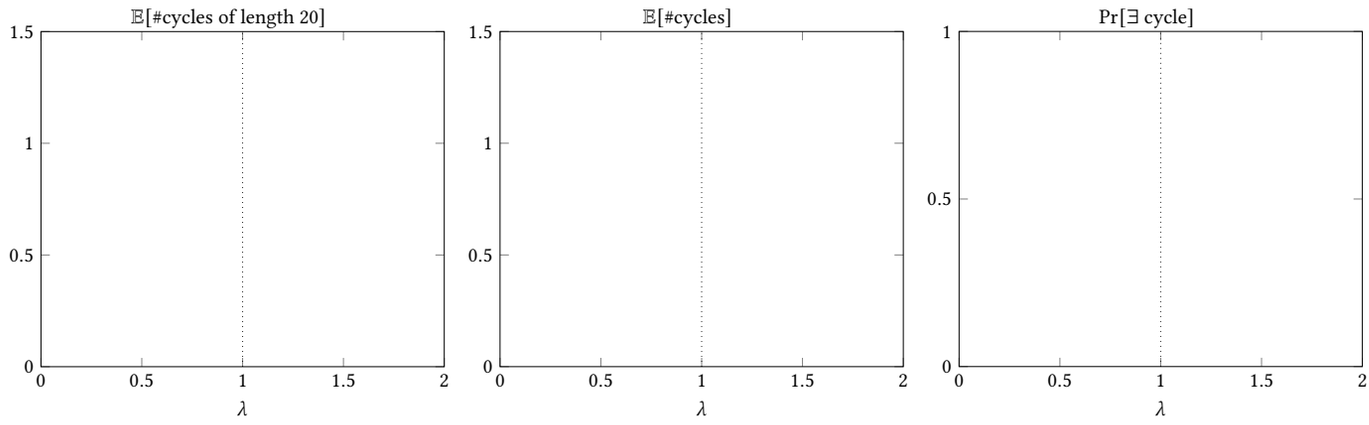
## Exercise 1 – Intuition for Erdős–Rényi Graphs

We consider the Erdős–Rényi graph $G(n, \lambda n/2)$ or the Gilbert graph $G(n, \lambda/n)$ (both lead to the same result). The expected vertex degree is therefore $\lambda \pm O(1/n)$.

(i) Sketch the behavior of the following probabilities and expectations for $n \to \infty$ as a function of $\lambda \in [0, 2]$.

**Hint:** This is not about numerical exactness but about the qualitative behavior. Where does the curve take the value 0, 1, or $\infty$? Does anything special happen at $\lambda = 1$?

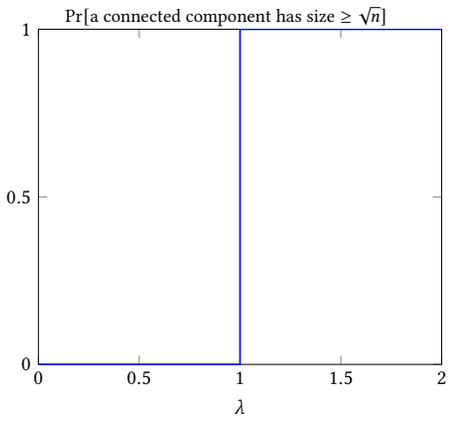




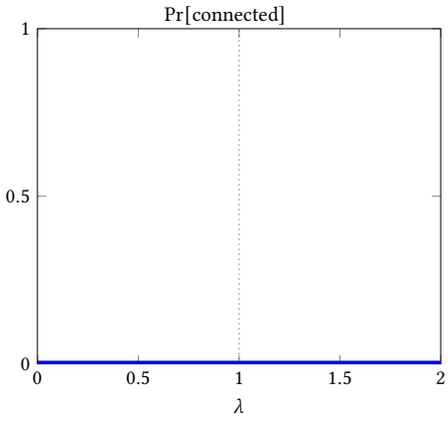

(ii) Let $\lambda = \Theta(1)$. Guess: What holds with high probability for (the order of magnitude of) the minimum and maximum degree as a function of $n$?

$$\min_{v\in[n]} \deg(v) = \text{.....} \qquad \max_{v\in[n]} \deg(v) = \Theta\Big(\text{.....}\Big)$$

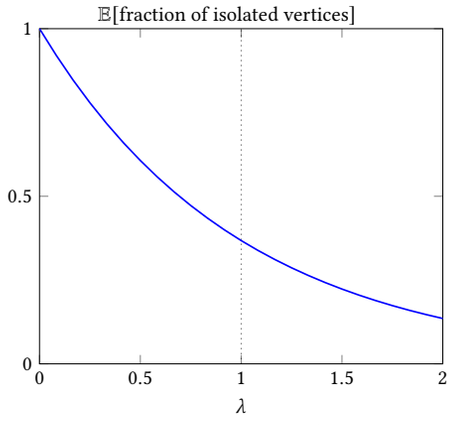
## Solution 1



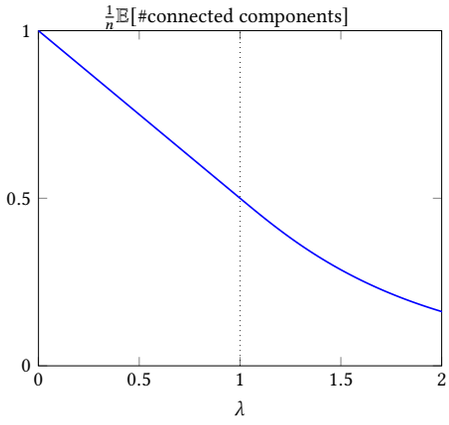

follows from the "Sudden Emergence" result



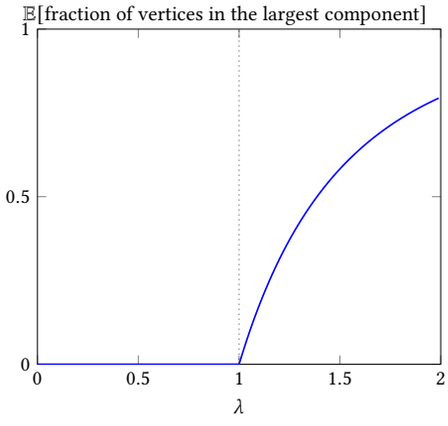

see next question



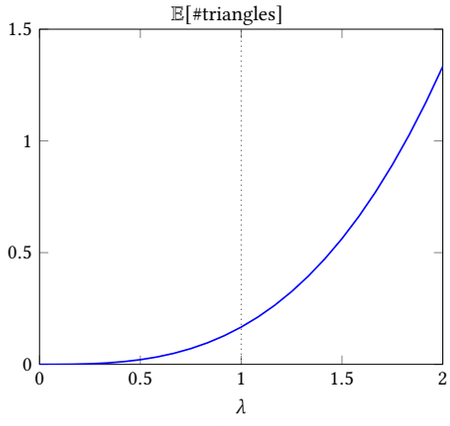

$\Pr_{X\sim \mathrm{Po}(\lambda)}[X = 0] = e^{-\lambda}$



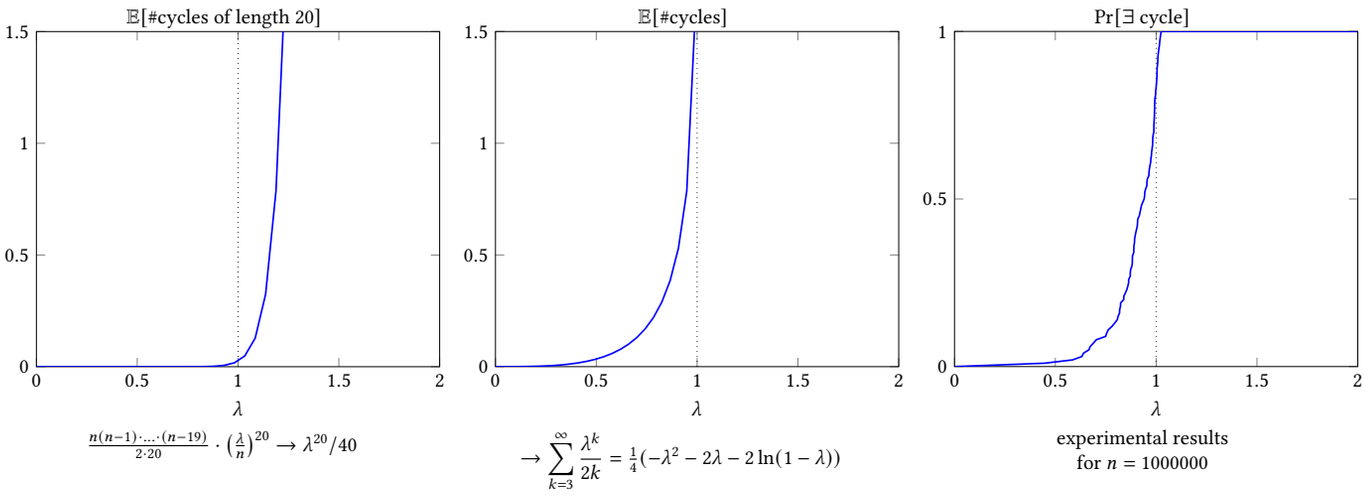

experimental results
for $n = 1000000$




$\frac{\lambda+W(-e^{-\lambda}\lambda)}{\lambda}$ for $\lambda \geq 1$,
see Exercise 3




$\binom{n}{3}\left(\frac{\lambda}{n}\right)^3 \to \lambda^3/6$




(ii) The minimum degree is 0 with high probability. In part (a) we even saw that there are in expectation $\Theta(n)$ isolated vertices. A formal proof can be somewhat tricky.

The maximum degree is $\Theta(\frac{\log n}{\log\log n})$ with high probability. This can be seen by noting that the expected number of vertices of degree $d$ is approximately $\mathbb{E}[N_d] \approx n \cdot e^{-\lambda}\frac{\lambda^d}{d!}$. Setting $\mathbb{E}[N_d] = 1$ yields $d = \Theta(\frac{\log n}{\log\log n})$. The main issue here is which $d$ is needed so that $d! \approx n$ holds.

## Exercise 2 – Vertex Degrees in Erdős–Rényi Graphs

*On slide 15 of the section on balls into bins and Poissonization we showed that* $\mathrm{Bin}(n, \frac{\lambda}{n})$ *converges to* $\mathrm{Pois}(\lambda)$ *as* $n \to \infty$ *(or, respectively, that the CDFs converge). In the following exercise you may use without proof:*

> **Lemma.** Let $t_1, t_2, \cdots \in \mathbb{N}$ and $p_1, p_2, \cdots \in (0,1)$ as well as $\lambda \in \mathbb{R}_+$. Furthermore let $X_n \sim \mathrm{Bin}(t_n, p_n)$ for all $n \in \mathbb{N}$ and $X \sim \mathrm{Pois}(\lambda)$. If $t_n \to \infty$ and $t_n \cdot p_n \to \lambda$ as $n \to \infty$, then $X_n \xrightarrow{\mathrm{d}} X$ as $n \to \infty$.

Let $\lambda > 0$ and $X \sim \mathrm{Pois}(\lambda)$. We consider variants of Erdős–Rényi graphs from the lecture. We set the expected degree to approximately $\lambda$ and want to show (using the above lemma) that the distribution of a single vertex converges asymptotically to $\mathrm{Pois}(\lambda)$ as $n \to \infty$ (while $\lambda$ remains constant).

(i) Let $X_n$ be the degree of vertex 1 in $G(n,p)$ with $p = \lambda/n$. Show $X_n \xrightarrow{\mathrm{d}} X$.

(ii) Let $X_n$ be the degree of vertex 1 in $G^{\mathrm{UE}}(n,m)$ with $m = \lfloor \lambda n/2 \rfloor$. Show $X_n \xrightarrow{\mathrm{d}} X$.

(iii) Let $X_n$ be the degree of vertex 1 in $G(n,m)$ with $m = \lfloor \lambda n/2 \rfloor$. Show $X_n \xrightarrow{\mathrm{d}} X$.

**Hint:** The last part is by far the most difficult. It is helpful to "trap" $G(n,m)$ using a coupling between two Gilbert graphs $G^- = (n, p^-)$ and $G^+ = (n, p^+)$.

## Solution 2

(i) Since each of the $n-1$ edges incident to vertex 1 is present with probability $\lambda/n$, we have $X_n \sim \mathrm{Bin}(n-1, \lambda/n)$. We apply the lemma with $t_n = n-1$ and $p_n = \lambda/n$. Clearly $t_n \to \infty$ and $t_n p_n \to \lambda$. Thus $X_n \to X$ as desired.

(ii) Since each of the $2m$ edge endpoints is attached to vertex 1 with probability $1/n$, we have $X_n = \mathrm{Bin}(2m, \frac{1}{n})$. We apply the lemma with $t_n = 2m = 2\lfloor \lambda n/2 \rfloor$ and $p_n = 1/n$. Clearly $t_n \to \infty$ and $t_n p_n \to \lambda$. Thus $X_n \to X$ as desired.

(iii) Let $G = G(n,m)$. We additionally consider the Gilbert graphs $G^- = G(n,p^-)$ and $G^+ = G(n,p^+)$ where $p^- := \frac{\lambda}{n} - n^{-4/3}$ and $p^+ := \frac{\lambda}{n} + n^{-4/3}$. Since $p^-$ and $p^+$ differ only slightly from $p = \lambda/n$, the degree $X_n^-$ of vertex 1 in $G^-$ and the degree $X_n^+$ of vertex 1 in $G^+$ converge as in part (i), i.e. $X_n^- \xrightarrow{\mathrm{d}} X$ and $X_n^+ \xrightarrow{\mathrm{d}} X$. To "trap" $X_n$ between $X_n^-$ and $X_n^+$ we use a coupling.

Let $\mathcal{E}$ be the set of the $\binom{n}{2}$ possible edges. In a common probability space, for each $e \in \mathcal{E}$ there is an independent random variable $Z_e \sim \mathcal{U}([0,1])$. The edge sets $E^-$, $E$, and $E^+$ of $G^-$, $G$, and $G^+$ (or formally of their "copies" $G^{-\prime}$, $G'$, and $G^{+\prime}$ with the same distribution) are defined as:

$$\begin{aligned} E^- &= \{e \in \mathcal{E} \mid Z_e \le p^-\} \\ E &= \{e \in \mathcal{E} \mid Z_e \text{ belongs to the } m \text{ smallest numbers in } (Z_e)_{e\in\mathcal{E}}\} \\ E^+ &= \{e \in \mathcal{E} \mid Z_e \le p^+\} \end{aligned}$$

It is clear that this yields the correct distributions, i.e. that this is a valid coupling. Let $m^- = |E^-|$ and $m^+ = |E^+|$. We have:

$$\begin{aligned} m^- &\sim \mathrm{Bin}\left(\tbinom{n}{2}, p^-\right) \\ \mathbb{E}[m^-] &= \tbinom{n}{2} p^- = \tfrac{n(n-1)}{2}(\tfrac{\lambda}{n} - n^{-4/3}) = \tfrac{\lambda n}{2} - \Theta(n^{2/3}) = m - \Theta(n^{2/3}) \\ \Pr[m^- \ge m] &\le \Pr[|m^- - \mathbb{E}[m^-]| \ge \Theta(n^{2/3})] \overset{\text{Cheb.}}{\le} \frac{\mathrm{Var}(m^-)}{\Theta(n^{4/3})} = \frac{\Theta(n)}{\Theta(n^{4/3})} = \Theta(n^{-1/3}). \end{aligned}$$

Analogously one shows that $\Pr[m^+ \le m] = \Theta(n^{-1/3})$. We now consider the event $\mathrm{succ} = \{m^- \le m \le m^+\}$. If succ occurs, then by construction $E^- \subseteq E \subseteq E^+$ and thus $X_n^- \le X_n \le X_n^+$. By the above calculation and a union bound we have $\Pr[\mathrm{succ}] = 1 - \Theta(n^{-1/3})$. It follows for $i \in \mathbb{N}_0$:

$$\begin{aligned} \Pr[X_n \le i] &= \Pr[X_n \le i \wedge \mathrm{succ}] + \Pr[X_n \le i \wedge \overline{\mathrm{succ}}] \le \Pr[X_n^- \le i \wedge \mathrm{succ}] + \Theta(n^{-1/3}) \\ &\le \Pr[X_n^- \le i] + \Theta(n^{-1/3}) \longrightarrow \Pr[X \le i]. \quad \text{// since } X_n^- \xrightarrow{\mathrm{d}} X \end{aligned}$$

Similarly one obtains

$$\begin{aligned} \Pr[X_n \le i] &\ge \Pr[X_n \le i \wedge \mathrm{succ}] \ge \Pr[X_n^+ \le i \wedge \mathrm{succ}] = \Pr[X_n^+ \le i] - \Pr[X_n^+ \le i \wedge \overline{\mathrm{succ}}] \\ &\ge \Pr[X_n^+ \le i] - \Theta(n^{-1/3}) \longrightarrow \Pr[X \le i]. \quad \text{// since } X_n^+ \xrightarrow{\mathrm{d}} X \end{aligned}$$

Hence $\Pr[X_n \le i] \to \Pr[X \le i]$ for all $i \in \mathbb{N}_0$, and thus $X_n \xrightarrow{\mathrm{d}} X$.

## Exercise 3 – Extinction Probability in Galton–Watson Trees

Let GWT($\lambda$) be the Galton–Watson tree with offspring distribution Pois($\lambda$).

(i) Let $n_i$ be the number of vertices in level $i$ of GWT($\lambda$). The root level is level 0. What is $\mathbb{E}[n_i]$?

(ii) For $i \in \mathbb{N}_0$ let $p_i$ be the probability that GWT($\lambda$) has at least one vertex at level $i$. Express $p_{i+1}$ in terms of $p_i$.
**Hint:** Use Exercise 3 (iv) from Sheet 9.

(iii) Determine, for $\lambda \in \{0, 0.5, 1, 1.1, 1.5\}$ (or for arbitrary $\lambda$), approximations for the probability $s(\lambda)$ that GWT($\lambda$) is infinite. A computer algebra system may be useful (e.g. Wolfram Alpha).

**Additional consideration:** What is the expected number of vertices at level $i$?

## Solution 3

(i) Each vertex has expected $\lambda$ children in the next level. It is quite intuitive that therefore $\mathbb{E}[n_i] = \lambda^i$. Formally one can use conditional expectations and induction. Written compactly:

$$\mathbb{E}[n_i] = \mathbb{E}[\mathbb{E}[n_i \mid n_{i-1}]] = \mathbb{E}[\lambda n_{i-1}] = \lambda \mathbb{E}[n_{i-1}] \overset{\text{Ind.}}{=} \lambda \lambda^{i-1} = \lambda^i.$$

(ii) Let $\operatorname{depth}(T) \in \mathbb{N}_0 \cup \{\infty\}$ be the depth of a tree $T$. Let $X \sim \text{Pois}(\lambda)$ be the number of children of the root of GWT($\lambda$) and let $T^{(1)}, T^{(2)}, \ldots, T^{(X)}$ be the subtrees starting at these children. Let $Y = |\{i \in \{1, \ldots, X\} \mid \operatorname{depth}(T^{(i)}) \geq i - 1\}|$ be the number of subtrees of depth at least $i-1$. Since the subtrees have the same distribution as GWT($\lambda$), we have $\Pr[\operatorname{depth}(T^{(i)}) \geq i - 1] = p_{i-1}$ for all $i \in \{1, \ldots, X\}$. Since the subtrees are independent, we furthermore have $Y \sim \text{Bin}(X, p_{i-1})$. By Exercise 3 (iv) from Sheet 9 it follows that $Y \sim \text{Pois}(\lambda p_{i-1})$. Hence:

$$\begin{aligned} p_i &= \Pr[\operatorname{depth}(\text{GWT}(\lambda)) \geq i] = \Pr[\exists j \in \{1, \ldots, X\} : \operatorname{depth}(T^{(j)}) \geq i - 1] \\ &= \Pr[Y > 0] = 1 - \Pr[Y = 0] = 1 - e^{-\lambda p_{i-1}}. \end{aligned}$$

(iii) It is intuitive that $s(\lambda) = \lim_{i \to \infty} p_i$, but we nevertheless show this formally. The probability that GWT($\lambda$) has *exactly* $i$ levels is $q_i = p_i - p_{i+1}$. Clearly $s(\lambda) + q_0 + q_1 + \cdots = 1$. It follows:

$$s(\lambda) = 1 - \lim_{i \to \infty} \sum_{j=0}^{i-1} q_j = \lim_{i \to \infty} \left( p_0 - \sum_{j=0}^{i-1} (p_j - p_{j+1}) \right) = \lim_{i \to \infty} p_i.$$

By (ii) we have $p_0 = 1$ and $p_{i+1} = 1 - e^{-\lambda p_i}$ for all $i \geq 0$. It is convenient to consider the function $f(x) = 1 - e^{-\lambda x}$. Iterating this function starting at $x = 1$ generates the

sequence $(p_i)_{i\in\mathbb{N}}$. Since $f$ is monotonically decreasing and our starting value satisfies $f(p_0) < p_0$, the sequence is monotonically decreasing. Hence $s(\lambda) = \lim_{i\to\infty} p_i$ is the largest fixed point of $f$. For $\lambda \leq 1$ we have $f(x) = 1 - e^{-\lambda x} \leq 1 - e^{-x} \leq 1 - (1 - x) = x$ with equality only for $x = 0$. For such $\lambda$ we therefore have $s(\lambda) = 0$.

For $\lambda > 1$ the largest solution of $x = 1 - e^{-\lambda x}$ differs from the trivial solution $x = 0$. A computer algebra system can determine this as $s(\lambda) = \frac{\lambda + W(-\lambda e^{-\lambda})}{\lambda}$, where $W$ denotes the so-called Lambert $W$-function. Numerically one obtains $s_{1.1} = 0.176134$ and $s_{1.5} = 0.582812$. A plot can be found in Exercise 1.

Karlsruhe Institute of Technology

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 14 – Cuckoo Hashing

## Probability and Computing

## Exercise 1 – Cuckoo Hashing & Erdős–Rényi Graphs

*Note: For this exercise, $n$ denotes a number of edges and $m$ a number of vertices.*

The "sudden emergence" result of Erdős and Rényi (slide 19, Random Graphs chapter) also holds if one replaces $G(m, \lambda/m)$ by $G^{\text{UE}}(m, \frac{\lambda m}{2})$.

(i) Describe a variant of Cuckoo Hashing which, with $n$ keys and $m$ table slots, is based on the graph $G^{\text{UE}}(m, n)$.

(ii) A practical disadvantage arises in the implementation of insert. What is it?

(iii) We want to insert keys up to a load of $\frac{n}{m} = \alpha < \frac{1}{2} - \varepsilon$ for some constant $\varepsilon > 0$. Deduce from the "sudden emergence" result that this is possible with high probability.

(iv) Assume now $\alpha = \frac{1}{2} + \varepsilon$ for some constant $\varepsilon > 0$. Deduce from the "sudden emergence" result that inserting up to load factor $\alpha$ will fail with high probability.

## Solution 1

(i) We use a table of size $m$ and two fully random hash functions $h_1, h_2 : D \to [m]$. Each key $x \in S$ corresponds to an edge $\{h_1(x), h_2(x)\}$ in the graph whose vertex set consists of the table positions. For $n = |S|$ keys, the graph has exactly the distribution of $G^{\text{UE}}(m, n)$ (multiple edges and loops are also possible here).

(ii) When displacing a key, it is initially unclear whether it was placed according to its first or its second hash function. One therefore has to try both in case of doubt in order to find the alternative position. This introduces a branch or duplicate hashing work. In practice, the "bipartite" variant as presented in the lecture is therefore common.

(iii) If the load $\frac{n}{m} = \alpha < \frac{1}{2} - \varepsilon$ holds, then the average degree is $\lambda = \frac{2n}{m} < 1 - 2\varepsilon$.

From the "sudden emergence" result and $\lambda < 1$ it follows that $G^{\text{UE}}(m, n)$ consists with high probability only of trees and pseudotrees, i.e., of components with at most one cycle. For these, it is always possible to orient the edges without collisions and thus place the keys without collisions.

(iv) We will show that whp $G^{\mathrm{UE}}(m,n)$ contains a "giant" component with more edges than vertices. (This means that some set of $k$ keys (corresponding to the edges) compete for less than $k$ table positions (corresponding to vertices), implying that cuckoo hashing fails by pidgeon hole principle as claimed.)

Imagine the edges of $G^{\mathrm{UE}}(m,n)$ are generated one by one. After the first $n' = (\frac{1}{2}+\varepsilon/2)m$ edges are generated, the average degree is $\lambda = \frac{2n'}{m} = 1+\varepsilon > 1$, so by the "sudden emergence" result with high probability there already exists a "giant" component of size $s(\lambda)n$ for some constant $s(\lambda) > 0$. Every subsequent edge we generate closes a cycle in the giant component with probability at least $s(\lambda)^2 = \Theta(1)$. A simple argument shows that the remaining $\varepsilon m/2 = \Theta(m)$ edges we generate close at least 2 cycles in the giant component with high probability, causing it to have more edges than vertices.

Karlsruhe Institute of Technology

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 15 – Peeling

## Probability and Computing

## Exercise 1 – Deterministic peeling in linear time

(a) The algorithm constructByPeeling is nondeterministic (in the choice of $i$ in the while-loop). Show that nevertheless, for any two possible executions of constructByPeeling, the same set of keys remains unplaced. (In particular, the nondeterminism does not affect success/failure.)

(b) Refine the pseudocode so that a deterministic algorithm results, which clearly has running time $O(n + m)$. Use suitable auxiliary data structures of your choice.

## Solution 1

(a) Let $X = (x_1, \ldots, x_k)$ be the sequence of keys that a complete execution of constructByPeeling places (in this order) before the while-loop terminates. Let $Y = (y_1, \ldots, y_\ell)$ be another possible sequence. It suffices to show that every element that occurs in $X$ also occurs in $Y$: by symmetry it then follows that $X$ and $Y$ contain the same elements (possibly in a different order).

We give a proof by contradiction and assume that $j \in [k]$ is the smallest index such that $x_j$ does not occur in $Y$. In the $X$-execution, $x_j$ is placed at an index $i_j$ that is not feasible for any key from $S \setminus \{x_1, \ldots, x_{j-1}\}$. Thus, the index $i_j$ is also not feasible for any key from $S \setminus \{y_1, \ldots, y_\ell\}$, since $\{y_1, \ldots, y_\ell\} \supseteq \{x_1, \ldots, x_{j-1}\}$ by the choice of $j$. By assumption, $x_j$ is still in $S$ at the end of the $Y$-execution. But then $i_j$ would be an index that is feasible only for $x_j$ and for no other remaining key. Hence another iteration of the while-loop would be possible. Thus $Y$ does not describe a complete execution of constructByPeeling. Contradiction.

(b) We base the algorithm on the bipartite cuckoo graph as defined in the lecture. We use a queue $Q$. The invariant is that $Q$ contains all table vertices of degree 1 (and possibly additional table vertices). It does not matter whether $Q$ is a FIFO, LIFO, or another type of queue.

```
Algorithm constructByPeeling-Linear(S, h_1, h_2, h_3):
    T ← [⊥, ..., ⊥] // empty table
    G ← (S, [m], {(x, h_i(x)) | x ∈ S, i ∈ [3]}) // cuckoo graph
    Q ← ∅ // empty queue
    for i ∈ [m] do
        if deg_G(i) = 1 then
            Q.push(i)
    while not Q.isEmpty do
        i ← Q.pop
        if deg_G(i) = 1 then
            x ← unique neighbor of i
            T[i] ← x
            S ← S \ {x}
            for j ∈ {h_1(x), h_2(x), h_3(x)} do
                delete the edge (x, j) from G
                if deg_G(j) = 1 then
                    Q.push(j)
    if S = ∅ then
        return T
    else
        return NOT-PEELABLE
```

$G$ can be constructed initially in time $O(n + m)$ and every further individual operation can be performed in $O(1)$. Since each vertex can become a degree-1 vertex only once (because we only delete edges), $Q$.push($i$) is executed at most once for each $i \in [m]$. Thus, the number of iterations of the while-loop is also at most $m$ (since in each iteration one element is removed from $Q$). Overall this yields $O(m)$.

**Remark.** Instead of choosing an adjacency-list representation of $G$, in the case at hand a more space-efficient representation suffices. For each table position we store two things: its degree (which it would have in $G$) and the *sum* of the elements from $S$ connected to it (this assumes that the universe $D$ is a group, for example $D = \mathbb{Z}_{2^{64}}$). From this representation one can determine whether the degree of a table vertex is 1 and in this case extract the unique neighbor. If the degree is larger than 1, one can remove a given neighbor by subtraction. This data structure does not require pointers.

## Exercise 2 – Peeling with 2 hash functions

Suppose we use the peeling algorithm constructByPeeling in a setting with only two hash functions $h_1$ and $h_2$. For simplicity we assume that $h_1(x) \neq h_2(x)$ for all $x \in D$, and under this restriction the pairs $(h_1(x), h_2(x))$ are uniformly random and independent for different

$x \in D$.[1] Show:

(a) All keys are placed if and only if the following graph is acyclic:

$$G = ([m], \{\{h_0(x), h_1(x)\} \mid x \in S\})$$

**Note:** $G$ is to be understood as a multigraph.

(b) Let $\alpha > 0$ be a constant and $n = \lfloor \alpha m \rfloor$. Show that (for $m \to \infty$) there is a cycle with probability $\Omega(1)$.
**Hint:** It suffices to look for cycles of length 2 (i.e., double edges).
See `https://en.wikipedia.org/wiki/Birthdayproblem#Arbitrarynumberofdays`.

Hence there is no peelability threshold as there is for $k \geq 3$, since for no $\alpha = \Theta(1)$ does constructByPeeling succeed with high probability.

## Solution 2

(a) We can view constructByPeeling as an algorithm on $G$ that repeatedly searches for vertices of degree 1. If $v$ is such a vertex whose incident edge comes from a key $x$, then $x$ is placed at $v$ and the edge to $x$ is deleted. Thus, all keys are placed if the successive deletion of edges with an endpoint of degree 1 ends with the empty graph.

If we start with a forest $G$, then $G$ is a forest at every step (deleting edges from a forest yields a forest again). As long as the forest is not empty, i.e., at least one tree with at least one edge remains, it has a leaf of degree 1 at which the process can continue. Hence: If $G$ is a forest, all keys are placed.

If, on the other hand, $G$ has at least one cycle originating from keys $x_1, \ldots, x_k$, then none of these keys can be the first to be placed by constructByPeeling, since both possible positions of the key are also feasible for a cyclically adjacent key. Thus, not all keys are placed.

(b) We can imagine that when determining the hash values of all keys, we draw $n$ times with replacement from an urn with $\binom{m}{2}$ balls, which correspond to the possible edges. The probability that we draw pairwise distinct balls is:

$$\begin{aligned}
&1 \cdot \left(1 - \frac{1}{\binom{m}{2}}\right) \cdot \left(1 - \frac{2}{\binom{m}{2}}\right) \cdot \ldots \cdot \left(1 - \frac{n-1}{\binom{m}{2}}\right) \\
&\leq 1 \cdot \left(1 - \frac{1}{m^2/2}\right) \cdot \left(1 - \frac{2}{m^2/2}\right) \cdot \ldots \cdot \left(1 - \frac{n-1}{m^2/2}\right) \\
&\leq \exp\left(-\frac{1}{m^2/2}\right) \cdot \exp\left(-\frac{2}{m^2/2}\right) \cdot \ldots \cdot \exp\left(-\frac{n-1}{m^2/2}\right) \\
&= \exp\left(-\frac{n(n-1)/2}{m^2/2}\right) = \exp\left(-\frac{n(n-1)}{m^2}\right) \xrightarrow{m\to\infty} \exp(-\alpha^2).
\end{aligned}$$

[1] To ensure this we can, for example, define $h_2(x) := (h_1(x) + h_{\text{diff}}(x)) \bmod m$ for some $h_{\text{diff}} : D \to [m-1]$.

In the first step we use that $1 + x \leq e^x$ holds for $x \in \mathbb{R}$. Later we use $1 + 2 + \cdots + n - 1 = n(n-1)/2$. The probability that we do not draw distinct balls and hence that a double edge occurs in $G$ is therefore at least $1 - e^{-\alpha^2}$ in the limit, and thus $\Omega(1)$. In particular, there is a cycle with probability $\Omega(1)$.

## Exercise 3 – The peeling algorithm does not get stuck late[2]

For a set of keys $S \subseteq D$ of size $n = |S|$ and hash functions $h_1, h_2, h_3 \sim \mathcal{U}([m]^D)$ we consider the cuckoo graph as in the lecture:

$$G = G_{S,h_1,h_2,h_3} = (S, [m], \{(x, h_i(x)) \mid x \in S, i \in [3]\})$$

We assume only $\alpha = \frac{n}{m} \leq 1$. Let $S' \subseteq S$ be the set of keys that cannot be removed by the peeling algorithm (we have $|S'| \in \{0, \ldots, n\}$). We want to show that $\Pr[|S'| \in \{1, \ldots, \delta m\}] = O(1/m)$ for a constant $\delta > 0$ (to be chosen later).

*Intuition: Either $S' = \emptyset$ or $|S'|$ is $\Omega(m)$; everything in between is unlikely.*

(a) Observe: $|S'| = 1$ is not possible.

(b) Show: $|N(S')| \leq \frac{3}{2}|S'|$. Here $N(S')$ is the set of all neighbors of $S'$ in $G$.

(c) Show that there exists a constant $C$ such that for $s \in \{2, \ldots, n\}$ the following holds:

$$p_s := \Pr[\exists X \subseteq S, |X| = s : \exists Y \subseteq [m], |Y| = \lfloor \tfrac{3}{2}s \rfloor : N(X) \subseteq Y] \leq \left(C \cdot \frac{s}{m}\right)^{s/2}.$$

**Hint:** Simply apply a brute-force union bound over all choices of $X$ and $Y$. The bound $\binom{n}{k} \leq (\frac{ne}{k})^k$ for binomial coefficients is also useful. Ignore the floor functions; everyone does that.

(d) Show (i) $p_2 + p_3 + p_4 + p_5 = O(1/m)$, (ii) $\sum_{s=6}^{\sqrt{m}} p_s = O(1/m)$, (iii) $\sum_{s=\sqrt{m}}^{m/(2C)} p_s = O(1/m)$.

(e) Choose $\delta = 1/2C$ and show $\Pr[|S'| \in \{1, \ldots, \delta m\}] \leq \sum_{s=2}^{\delta m} p_s = O(1/m)$.

## Solution 3

(a) A single key cannot block itself.

(b) Depending on whether we count multiple edges or not, exactly $3|S'|$ or at most $3|S'|$ edges are incident to $S'$. Since every bucket in $N(S')$ must be hit by two *different* keys from $S'$ (otherwise peeling could continue), each vertex in $N(S')$ has at least two neighbors in $S'$. Thus, for the number $e$ of edges between $S'$ and $N(S')$ we have $3|S'| \geq e \geq 2|N(S')|$. Rearranging yields the claim.

[2] This exercise is somewhat involved. It is mainly on the sheet because otherwise the proof from the lecture would be incomplete.

(c) There are $\binom{n}{s}$ ways to choose $X$ and $\binom{m}{(3/2)s}$ ways to choose $Y$. For any $x \in X$ we have $\Pr[N(\{x\}) \subseteq Y] = (|Y|/m)^3$, since all three hash values must (independently) land in the set $Y$. We thus obtain the following (along the way we combine some constants into a $C = \Theta(1)$):

$$\begin{aligned} p_s &= \Pr[\exists X \subseteq S, |X| = s : \exists Y \subseteq [m], |Y| = \lfloor \tfrac{3}{2}s \rfloor : N(X) \subseteq Y] \\ &\leq \binom{n}{s}\binom{m}{(3/2)s}\left(\frac{(3/2)\cdot s}{m}\right)^{3s} \leq \left(\frac{ne}{s}\right)^{s}\left(\frac{me}{(3/2)s}\right)^{(3/2)s}\left(\frac{(3/2)\cdot s}{m}\right)^{3s} \\ &\leq \left(\frac{n^2e^2}{s^2}\right)^{s/2}\left(\frac{2^3m^3e^3}{3^3s^3}\right)^{s/2}\left(\frac{3^6s^6}{2^6m^6}\right)^{s/2} \leq \left(\frac{Cm^2m^3s^6}{s^2s^3m^6}\right)^{s/2} = \left(\frac{Cs}{m}\right)^{s/2}. \end{aligned}$$

(d) (i) We have $p_2 = O(m^{-1})$, $p_3 = O(m^{-3/2})$, $p_4 = O(m^{-2})$, $p_4 = O(m^{-5/2})$. This works.

(ii) Each summand in $\sum_{s=6}^{\sqrt{m}} \left(\frac{Cs}{m}\right)^{s/2}$ is at most $\left(\frac{C\sqrt{m}}{m}\right)^{6/2} = O(m^{-3/2})$. Since there are $O(m^{1/2})$ summands, they sum to at most $O(m^{-1})$.

(iii) Each summand in $\sum_{s=\sqrt{m}}^{m/(2C)} \left(\frac{Cs}{m}\right)^{s/2}$ is at most $\left(\frac{1}{2}\right)^{\sqrt{m}/2} = O(m^{-2})$ (a *very* rough bound). Since there are $O(m)$ summands, they sum to at most $O(m^{-1})$.

(e) Choose $\delta = \frac{1}{2C}$. Suppose $|S'| = s$ for some $s \in \{1, \ldots, \delta m\}$. Let $X = S'$ and $Y' := N(S')$. By (b) we have $|Y'| \leq \frac{3}{2}s$, so there exists a set $Y \supseteq Y'$ with $|Y| = \lfloor \frac{3}{2}s \rfloor$ and $N(S') \subseteq Y$. The sets $X$ and $Y$ satisfy the event whose probability we bounded by $p_s$. Summarizing, we can conclude:

$$\Pr[|S'| \in \{1, \ldots, \delta m\}] \leq \underbrace{\Pr[|S'| = 1]}_{0} + \sum_{s=2}^{\delta m} \Pr[|S'| = s] \leq \sum_{s=2}^{m/(2C)} p_s = O(1/m).$$

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 16 – Retrieval

## Probability and Computing

## Exercise 1 – AMQ from Retrieval

Let $b \in \mathbb{N}$ and $S$ be a set of size $n = |S|$. Use the peeling-based retrieval data structure from the lecture as a black box to construct a static filter (i.e., an approximate membership query data structure) for $S$ with false-positive probability $\varepsilon = 2^{-b}$.

State advantages and disadvantages of the resulting data structure compared to a Bloom filter with the same false-positive probability. You may assume that $b = O(\log n)$, so that bit strings of length $b$ can be processed in time $O(1)$.

## Solution 1

Let $f \sim \mathcal{U}([2^b]^D)$ be a fully random hash function. According to the Simple Uniform Hashing Assumption, we may store $f$ without using any memory. We call $f(x)$ the *fingerprint* of $x \in D$. Let $f_S : S \to [2^b]$ be the restriction of $f$ to $S$ ("the same function" but with a smaller domain).

We construct a retrieval data structure $R$ for $f_S$. We interpret $R$ as a filter data structure that returns YES on a query $x \in D$ if and only if $\text{eval}(R, x) = f(x)$. By construction, there are no false-negative answers. Now assume that $x \in D \setminus S$. The fingerprint $f(x)$ is uniformly distributed in $[2^b]$ and independent of everything that played a role in the construction of $R$. Whatever $\text{eval}(R, x)$ is, the probability that it coincides with $f(x)$, and hence that $x$ is a false-positive element, is therefore $2^{-b} = \varepsilon$ as desired.

The comparison to a Bloom filter with $\varepsilon = 2^{-b}$ is as follows:

- Lower space usage: about $1.23bn$ instead of about $1.44bn$.
- Faster construction: $O(n)$ instead of $O(nb)$.
- Faster queries: $O(1)$ instead of $O(b)$.
- No support for `insert`.

## Exercise 2 – Learned Data Structures

Let $S$ be a set of $n = |S|$ names with a uniquely associated gender $f : S \to \{\text{F}, \text{M}\}$. A clever student observes that most $x \in S$ with $f(x) = \text{F}$ end in a vowel and most $x \in S$ with $f(x) = \text{M}$ end in a consonant. This simple rule works for all but $\delta n$ of the names, for a small $\delta > 0$.

Construct a data structure with expected space usage $O(\delta n \log(1/\delta))$ that returns the correct gender $f(x)$ for every $x \in S$.
**Hint:** Combine an AMQ filter and a retrieval data structure in a clever way.
**Remark:** By *learned data structures* one understands a combination of classical data structures and machine learning techniques. As indicated by this exercise, the idea is to beneficially combine the pattern-recognition capabilities of machine learning techniques with the reliability guarantees of classical data structures.

## Solution 2

Let $F \subseteq S$ be the set of the $\delta n$ names for which the heuristic fails. Let $B$ be an AMQ filter for $F$ with false-positive probability $\delta$. Let $S^+$ be the set of those names from $S$ for which the filter returns a positive answer. Besides $F$, $S^+$ also contains all $x \in S$ that are false-positive elements of $B$. The expected size of $S^+$ is at most $|F| + |S \setminus F| \cdot \delta = \delta n + (1-\delta)n \cdot \delta \leq 2\delta n$. Let $f_{S^+} : S^+ \to \{\text{F}, \text{M}\}$ be the restriction of $f$ to $S^+$. We construct a retrieval data structure $R$ for $f_{S^+}$.

The intuition is as follows: The AMQ filter identifies the names for which the heuristic fails, as well as some false positives. The retrieval data structure then has the task of separating true positives from false-positive elements. Formally, we can define our heuristic-supported retrieval data structure $R_H$ as follows:

**Algorithm** eval$(R_H, x)$**:**
  **if** query$(B, x) =$ NO **then**
    **return** Heuristic$(x)$
  **else**
    **return** eval$(R, x)$

The total space usage is $O(\delta n \log(1/\delta))$ bits for $B$ and expected at most $O(\delta n)$ bits for $R$.

## Exercise 3 – Retrieval with Variable Bit Length

According to the lecture, for every universe $D$, every set $S \subseteq D$, and every function $f : S \to \{0,1\}$ we can construct a retrieval data structure for $f$ with space usage $1.23|S|$. This shall be used here as a black box.

Construct a retrieval data structure for the case in which the range of the function $f$ contains bit strings of variable length.

More precisely, let $C \subseteq \{0,1\}^*$ be a prefix-free code, $D'$ a universe, $T \subseteq D'$, and $g : T \to C$ a function. Construct a data structure $R$ with space usage $1.23 \cdot \sum_{x \in T} |g(x)|$ and a corresponding algorithm eval such that for every $x \in T$ we have $\text{eval}(R, x) = g(x)$.
**Hint:** Introduce as many keys for each $x \in T$ as the length $|g(x)|$ of $g(x)$.

## Solution 3

The idea is to introduce, for each $x \in T$ with $g(x) = (b_1, \ldots, b_k)$, the keys $\{(x,1), (x,2), \ldots, (x,k)\} \subseteq D' \times \mathbb{N}$, where $(x,i)$ is assigned the value $b_i$. This yields a function $f : S \to \{0,1\}$ where

$S \subseteq D' \times \mathbb{N}$ is a set of pairs with $|S| = \sum_{x \in T} |g(x)|$.

We can thus, according to the lecture, construct a retrieval data structure $R$ with space usage $1.23|S|$ for the universe $D = D' \times \mathbb{N}$ as well as $S$ and $f$ as described. To extract the value $g(x)$ from this data structure for $x \in T$, we successively consider $\text{eval}(R, (x, 1))$, $\text{eval}(R, (x, 2))$, $\text{eval}(R, (x, 3)), \ldots$ until the observed sequence forms a code from $C$. Since $C$ is prefix-free, we know that no extension of the sequence lies in $C$, and thus we have completely determined $g(x)$.

For a query $x \in D' \setminus T$, it does not matter which element of $C$ is returned; we only need to ensure that eval does not crash or enter an infinite loop. This is easy (and in fact almost automatic).

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet 17 – Perfect Hashing

## Probability and Computing

## Exercise 1 – Minimal Perfect Hash Function with Brute Force

Consider the following pair of algorithms for the construction and evaluation of minimal perfect hash functions[1]. Let $S \subseteq D$ and $n = |S|$. According to the SUHA, we assume that $h_1, h_2, h_3, \ldots \sim \mathcal{U}([n]^D)$ are independent, fully random hash functions that themselves require no storage space.[2]

**Algorithm** construct($S$)**:**
  **for** seed $= 1$ **to** $\infty$ **do**
    **if** $|\{h_{\text{seed}}(x) \mid x \in S\}| = n$ **then**
      **return** seed

**Algorithm** eval(seed, $x$)**:**
  **return** $h_{\text{seed}}(x)$

(a) Argue: Each loop iteration in construct succeeds with probability $\frac{n!}{n^n}$.

(b) Argue: The return value seed of construct satisfies $\mathbb{E}[\text{seed}] = \frac{n^n}{n!}$.

(c) The space requirement is space $= \lceil \log_2(\text{seed}) \rceil$ bits. Show: $\mathbb{E}[\text{space}] \leq n \log_2(e) + 1$.
**Hint:** Use Jensen's inequality (Exercise Sheet 10) as well as Stirling's approximation of the factorial function. (see en.wikipedia.org/wiki/Stirling's_approximation).

(d) Comment on the expected space requirement and the expected construction time of the presented method.

## Solution 1

(a) The condition $|\{h_{\text{seed}}(x) \mid x \in S\}| = n$ is equivalent to the $n$ hash values of the $n$ keys in $S$ under $h_{\text{seed}}$ being pairwise distinct. Abstractly speaking, we draw $n$ times with replacement from an urn containing $n$ balls and each time must draw a ball that has not been drawn before. The probability of this event is

$$\frac{n}{n} \cdot \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \ldots \cdot \frac{2}{n} \cdot \frac{1}{n} = \frac{n!}{n^n}.$$

[1] Note: The "data structure" here consists solely of the number "seed".

[2] Alternatively, one may imagine a single hash function $h : \mathbb{N} \times D \to [n]$ that is fully random and, in addition to the actual key, accepts a natural number as a seed.

(b) How many times do you have to roll a fair six-sided die on average until you obtain the first six? Six times. The same reasoning applies here:

The random variable seed denotes how many times we perform a random experiment with success probability $p = n!/n^n$ until the first success occurs. Hence seed $\sim \text{Geom}(p)$. By direct computation or consulting a reference on geometric random variables, we obtain $\mathbb{E}[\text{seed}] = 1/p = \frac{n^n}{n!}$.

(c) From Stirling's approximation we obtain $n! \geq \left(\frac{n}{e}\right)^n$. Thus:

$$\begin{aligned}\mathbb{E}[\text{space}] &= \mathbb{E}[\lceil \log_2(\text{seed}) \rceil] \leq \mathbb{E}[\log_2(\text{seed})] + 1 \overset{\text{Jensen}}{\leq} \log_2(\mathbb{E}[\text{seed}]) + 1 \\ &= \log_2\left(\frac{n^n}{n!}\right) + 1 \leq \log_2\left(\frac{n^n}{(n/e)^n}\right) + 1 = \log_2(e^n) + 1 = n \cdot \log_2(e^n) + 1.\end{aligned}$$

(d) The expected space requirement is (almost) optimal in view of the lower bound that we will learn in the next exercise. The expected construction time is at least $n^n/n! \approx e^n$, which is therefore.

## Exercise 2 – Lower Space Bounds for MPHF

We will show that, in general, a minimal perfect hash function cannot be represented with fewer than $\log_2(e) \approx 1.44$ bits per element.

Recall the definition of a perfect hash function from the lecture. Let $\varepsilon = 0$, $n = m \in \mathbb{N}$, $d = |D|$. Let $\mathcal{I} := \{S \subseteq D \mid |S| = n\}$ denote the set of all possible inputs of size $n$. We consider the number of inputs for which a given data structure $P$ can simultaneously serve as a perfect hash function. Formally:

$$\text{cov}(P) := \left\{ S \in \mathcal{I} \;\middle|\; \{\text{eval}_P(x) \mid x \in S\} = [n] \right\}.$$

(a) As a warm-up: Suppose $n = 3$, $D = \{\text{a}, \text{b}, \text{c}, \text{d}, \text{e}, \text{f}, \text{g}, \text{h}\}$ and $P$ is a data structure for which eval behaves as shown in the following table. Argue: $P$ can serve as an MPHF for $|\text{cov}(P)| = 12$ different $S \in \mathcal{I}$.

| $x \in D$ | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| $\text{eval}_P(x)$ | 3 | 3 | 2 | 1 | 1 | 3 | 1 | 3 |

(b) Now let $n$ and $d$ as well as $P$ be arbitrary. Show $\text{cov}(P) \leq (\frac{d}{n})^n$.
**Hint:** Use without proof that among all rectangular boxes with a given sum of edge lengths, the cube has maximum volume. In $n$ dimensions this means $\max_{\substack{0 \leq c_1, \ldots, c_n \leq d \\ c_1 + \cdots + c_n = d}} c_1 \cdot \ldots \cdot c_n = (d/n)^n$.

Let now $\mathcal{P} = \{\text{construct}(S) \mid S \in \mathcal{I}\}$ be the set of all distinct data structures produced by construct.

(c) Argue $\bigcup_{P \in \mathcal{P}} \mathrm{cov}(P) = \mathcal{I}$ and conclude $|\mathcal{P}| \geq \binom{d}{n} / (\frac{d}{n})^n$.

(d) Show $\binom{d}{n} / (\frac{d}{n})^n \geq \frac{n^n}{n!} \cdot (1 - o(1))$ if $d = \omega(n^2)$.
**Hint:** In an intermediate step, show that $(1 - \frac{n}{d})^n \geq 1 - \frac{n^2}{d} = 1 - o(1)$.

(e) Argue: If the data structures produced by construct are each encoded as a bit string of length $\ell$, then $\ell \geq \lceil \log_2(|\mathcal{P}|) \rceil$.

(f) Show that $\ell \geq \log_2(e) \cdot n - o(n)$ if $d = \omega(n^2)$.
**Hint:** Combine the previous parts and use Stirling's approximation of the factorial function.

## Solution 2

(a) For $S \in \mathrm{cov}(P)$ to hold, $S$ must contain exactly one element $x_1$ such that $\mathrm{eval}_P(x) = 1$, exactly one element $x_2$ such that $\mathrm{eval}_P(x) = 2$, and exactly one element $x_3$ such that $\mathrm{eval}_P(x) = 3$. From the table we see that there are 3 possible choices for $x_1$, only one possible choice for $x_2$, and 4 possible choices for $x_3$. Multiplying yields $3 \cdot 1 \cdot 4 = 12$ possibilities.

(b) In the spirit of the warm-up exercise, let $S_i = \{x \in D \mid \mathrm{eval}_P(x) = i\}$ for $i \in [n]$. For $P$ to be minimal perfect on $S$, the set $S$ must contain exactly one element from each of the sets $S_1, \ldots, S_n$. Define $c_i := |S_i|$. Then $\mathrm{cov}(P) = c_1 \cdot \ldots \cdot c_n$. Since the sets $(S_i)_{i \in [n]}$ form a partition of $D$, we have $c_1 + \cdots + c_n = d$. By the hint it follows that $\mathrm{cov}(P) \leq (d/n)^n$ as claimed.

(c) Correctness of the data structure guarantees that for every $S \in \mathcal{I}$, we have $S \in \mathrm{cov}(\mathrm{construct}(S))$. Hence every $S \in \mathcal{I}$ is covered by the union. Using a (non-probabilistic) union bound together with the previous part yields:

$$\binom{d}{n} = |\mathcal{I}| = \left| \bigcup_{P \in \mathcal{P}} \mathrm{cov}(P) \right| \leq \sum_{P \in \mathcal{P}} |\mathrm{cov}(P)| \leq \sum_{P \in \mathcal{P}} \left(\frac{d}{n}\right)^n = |\mathcal{P}| \cdot \left(\frac{d}{n}\right)^n.$$

(d) First, we show by induction that for every $\varepsilon > 0$ and every $i \in \mathbb{N}_0$ it holds that $(1 - \varepsilon)^i \geq 1 - \varepsilon i$. The base case $i = 0$ is clear. The induction step proceeds as follows:

$$(1 - \varepsilon)^{i+1} = (1 - \varepsilon)^i \cdot (1 - \varepsilon) \overset{\text{Ind}}{\geq} (1 - \varepsilon i) \cdot (1 - \varepsilon) = 1 - \varepsilon i - \varepsilon + \varepsilon^2 i \geq 1 - \varepsilon(i + 1).$$

Applying this with $\varepsilon = \frac{n}{d}$ and $i = n$ yields the hint. We now compute:

$$\begin{aligned} \frac{\binom{d}{n}}{\left(\frac{d}{n}\right)^n} = \frac{\frac{d \cdot (d-1) \cdot \ldots \cdot (d-n+1)}{n!}}{\left(\frac{d}{n}\right)^n} \geq \frac{\frac{(d-n)^n}{n!}}{\left(\frac{d}{n}\right)^n} = \frac{n^n}{n!} \cdot \left(\frac{d-n}{d}\right)^n = \frac{n^n}{n!} \cdot \left(1 - \frac{n}{d}\right)^n \\ \overset{\text{Hint}}{\geq} \quad \frac{n^n}{n!} \cdot \left(1 - \frac{n^2}{d}\right) = \frac{n^n}{n!} \cdot (1 - o(1)). \end{aligned}$$

(e) For each element of $\mathcal{P}$, a distinct bit string must be available. Since there are only $2^\ell$ bit strings of length $\ell$, we must have $|\mathcal{P}| \leq 2^\ell$. Rearranging and using integrality of $\ell$ yields the desired formula.

(f) Stirling's approximation yields $n! = \Theta((n/e)^n \cdot \sqrt{n})$. Hence:

$$\begin{aligned}
\ell &\geq \log_2(|\mathcal{P}|) \geq \log_2\left(\binom{d}{n} \Big/ \left(\frac{d}{n}\right)^n\right) \geq \log_2\left(\frac{n^n}{n!} \cdot (1 - o(1))\right) \\
&= \log_2\left(\frac{n^n}{(n/e)^n \cdot \Theta(\sqrt{n})}\right) = \log_2(e^n) - \log_2(\Theta(\sqrt{n})) \\
&= n \cdot \log_2(e) - \tfrac{1}{2}\log_2(n) \pm \Theta(1) = n \cdot \log_2(e) - O(\log n).
\end{aligned}$$

Karlsruhe Institute of Technology

Stefan Walzer, Stefan Hermann
ITI Prof. Sanders
winter term 2025/2026

# Exercise Sheet X – Mixed

## Probability and Computing

## Exercise 1 – It's all connected

We consider random graphs in the Gilbert model. For $n \in \mathbb{N}$ and $p \in [0,1]$ let

$$z(n,p) := \Pr[G(n,p) \text{ is connected}].$$

Show: For all $n \in \mathbb{N}$, $z(n,p)$ is monotonically increasing in $p$.

## Solution 1

Let $n \in \mathbb{N}$ be arbitrary and let $p_1, p_2 \in [0,1]$ with $p_1 < p_2$. We have to show that $z(n,p_1) \leq z(n,p_2)$. We use a coupling. For two vertices $u,v \in [n]$ with $u < v$ let $X_{u,v} \sim \mathcal{U}([0,1])$. Let $G_1 = ([n], E_1)$ and $G_2 = ([n], E_2)$ be the graphs with the following edge sets:

$$E_1 = \{\{u,v\} \mid u,v \in [n], u \neq v, X_{u,v} \leq p_1\}$$
$$E_2 = \{\{u,v\} \mid u,v \in [n], u \neq v, X_{u,v} \leq p_2\}$$

It should be clear that $G_1 \overset{\text{d}}{=} G(n,p_1)$ and $G_2 \overset{\text{d}}{=} G(n,p_2)$. Thus we have constructed a coupling between $G(n,p_1)$ and $G(n,p_2)$. From the definition of $E_1$ and $E_2$ it furthermore follows that $E_1 \subseteq E_2$ and therefore the implication "$G_1$ is connected $\Rightarrow$ $G_2$ is connected" holds. Hence:

$$\begin{aligned} z(n,p_1) &\overset{\text{def}}{=} \Pr[G(n,p_1) \text{ is connected}] \overset{\text{Coup.}}{=} \Pr[G_1 \text{ is connected}] \\ &\leq \Pr[G_2 \text{ is connected}] \overset{\text{Coup.}}{=} \Pr[G(n,p_2) \text{ is connected}] \overset{\text{def}}{=} z(n,p_2). \end{aligned}$$

## Exercise 2 – Sampling Gilbert Graphs

Let $p = \frac{\lambda}{n-1}$ for $\lambda = \Theta(1)$. Describe how one can sample $G(n,p)$ in expected time $O(n)$.

**Hint:** Let $X_1, X_2, \ldots \sim \text{Ber}(p)$ and $Y = \min\{i \in \mathbb{N} \mid X_i = 1\}$. Sample $Y$ in time $O(1)$.

## Solution 2

The $Y$ in the hint is the number of $\text{Ber}(p)$ trials until the first success, so $Y \sim \text{Geom}_1(p)$. On exercise sheet 3 (exercise 4) we described how to sample $Y$ in time $O(1)$.

To sample $G(n,p)$ we conceptually iterate through the pairs $(u,v)$ with $u,v \in [n]$ and $u < v$ in lexicographic order. The $n-1$ possible values of $u$ give rise to $n-1$ subsequences as follows:

$$\underbrace{(1,2),(1,3),\ldots,(1,n)}_{\text{subsequence } 1},\underbrace{(2,3),(2,4),\ldots,(2,n)}_{\text{subsequence } 2},\ldots\underbrace{(n-2,n-1),(n-2,n)}_{\text{subsequence } n-2},\underbrace{(n-1,n)}_{\text{subsequence } n-1}.$$

Each pair $(u,v)$ should lead to the edge $\{u,v\}$ with probability $p$. Instead of considering each pair individually, we jump ahead in the sequence by a $\mathrm{Geom}_1(p)$-distributed distance and thus directly to the next pair that yields an edge. If one works this out completely, one arrives at the following code:

**Algorithm** sampleGilbert$(n,p)$**:**

```
(u, v) ← (1, 1) // "to the left" of the first valid pair (1, 2)
E ← ∅
repeat
    g ~ Geom_1(p)
    v ← v + g
    while v > n do // next sublist?
        if u = n − 1 then
            return E
        u ← u + 1
        v ← v − n + u
    E ← E ∪ {{u, v}}
```

The repeat loop is executed $|E|+1$ times, the while loop is executed $n-1$ times. Since $\mathbb{E}[|E|] = p\binom{n}{2} = \Theta(\frac{\lambda}{n}n^2) = \Theta(n)$, this yields an expected running time of $O(n)$.

## Exercise 3 – Concentration around an unknown expectation

For two strings $S$ and $T$ let $\mathrm{lcs}(S,T)$ denote the maximum possible length of a string $L$ that can be obtained from both $S$ and $T$ by deleting characters. For example, for $S =$ ELEFANT and $T =$ ELLENLANG we have $\mathrm{lcs}(S,T) = 5$ because $L =$ ELEAN. One calls $L$ a *Longest Common Subsequence*.

In the following let $n \in \mathbb{N}$ and let $S,T \sim \mathcal{U}(\{0,1\}^n)$ be random bit strings.

(a) Show that $\mathbb{E}[\mathrm{lcs}(S,T)] \geq n/2$.

Remark: The number $\gamma_2 = \lim_{n\to\infty} \frac{1}{n}\mathbb{E}[\mathrm{lcs}(S,T)]$ is a so-called Chvátal–Sankoff number. It is known that it exists and that $\gamma_2 \in [0.788071, 0.826280]$. The exact value is unknown (simulations suggest $\gamma_2 \approx 0.811$).

(b) Let $\varepsilon > 0.$ be an arbitrary constant. Show $\Pr[|\mathrm{lcs}(S,T) - \mathbb{E}[\mathrm{lcs}(S,T)]| \geq \varepsilon n] \leq \exp(-\varepsilon^2 n)$.

## Solution 3

(a) Let $I = \{i \in [n] \mid S_i = T_i\}$. Obviously there exists a common bit string of length $|I|$ of $S$ and $T$, which one obtains by keeping in both strings the characters at the indices from $I$ and deleting the rest. Therefore:

$$\mathbb{E}[\mathrm{lcs}(S,T)] \geq \mathbb{E}[|I|] = \frac{n}{2}.$$

(b) By changing a single character in $S$ or $T$, $\mathrm{lcs}(S,T)$ can decrease by at most 1, since the affected character can always simply be deleted. Conversely, such a change can increase $\mathrm{lcs}(S,T)$ by at most 1. Thus $\mathrm{lcs}(S,T)$ arises as a function of $2n$ independent random variables (the $2n$ characters of $S$ and $T$) with "bounded difference constants" $c_i = 1$. We can therefore apply the bounded differences method and obtain:

$$\Pr[|\mathrm{lcs}(S,T) - \mathbb{E}[\mathrm{lcs}(S,T)]| \geq \varepsilon n] \leq \exp\left(-\frac{2(\varepsilon n)^2}{\sum_{i=1}^{2n} c_i}\right) = \exp\left(-\frac{2\varepsilon^2 n^2}{2n}\right) = \exp(-\varepsilon^2 n).$$

## Exercise 4 – A delicacy to conclude

A circular cake stands on a turntable. A *random cut* means that we rotate the cake randomly and then cut from the center towards the north.

(a) We cut twice at random. What is the expected size of the smaller piece (as a fraction of the whole cake)?

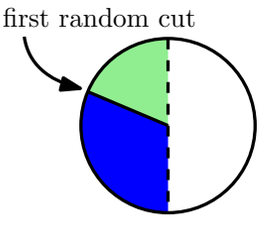
(b) There is a cherry in the cake (point-like, not exactly in the center). What is the probability that the cherry is in the smaller piece after the two random cuts?

## Solution 4

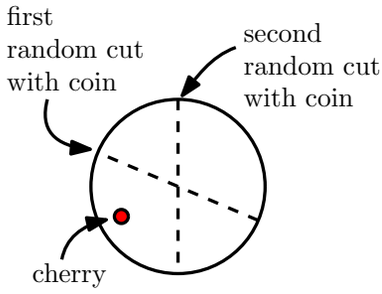
One can solve the problem using continuous probability spaces and integrals. However, it can also be done more elegantly. In each case we will shift a key aspect of the random experiment to the end, on which the result then exclusively depends.

By a *random cut with coin* we mean that we rotate the cake randomly and mark the north–south axis. Then we toss a fair coin. If it shows heads, we cut as before in the north. Otherwise in the south. It should be clear that this makes no difference for the distribution of the cut position.

(a) We first perform a random cut and then a random cut with coin. The situation before the coin toss is illustrated. Each of the two marked circular pieces will turn out to be the smaller piece with probability $1/2$. Since together they make up half of the cake, the expected fraction of the smaller piece is $1/4$.



(b) We perform two random cuts with coin, but shift both coin tosses to the very end. The situation before the coin tosses is illustrated. Each of the four pieces will turn out to be the smallest piece with probability 1/4. This applies in particular to the piece with the cherry.




**Remark:** You can also imagine that cherry is placed randomly in the end and use (a) to see that the probability of placing the cherry in the small piece is, on average, $\frac{1}{4}$.