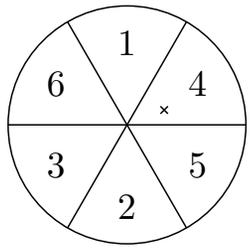
Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 1 – Grundbegriffe

Randomisierte Algorithmik

Aufgabe 1 – Wahrscheinlich weiß ich's noch ...

Eine Darts-Spielerin wirft einen Pfeil auf eine Dartscheibe. Diese ist in 6 gleichgroße Segmente unterteilt, welchen verschiedene Punkte aus $\{1, \dots, 6\}$ zugeordnet sind. Da sich die Spielerin noch im Training befindet, landet der Pfeil zufällig gleichverteilt auf der Scheibe (jedoch niemals daneben). Im Beispiel rechts hat der Wurf 4 Punkte erzielt.



(a) Modelliere das Zufallsexperiment des Wurfes (nicht das der resultierenden Punktzahl), indem du Ergebnismenge und Wahrscheinlichkeitsmaß eines kontinuierlichen Wahrscheinlichkeitsraumes beschreibst.

(b) Wir interessieren uns nun insbesondere für folgende Eigenschaften eines Wurfes:

- der Abstand der steckenden Pfeilspitze zum Zentrum der Scheibe
- die resultierende Punktzahl
- die resultierende Punktzahl ist gerade
- die resultierende Punktzahl modulo 2

Definiere die zugehörigen Zufallsvariablen und Ereignisse.

(c) Bestimme die Verteilungsfunktion des Abstands aus Teilaufgabe (b).

Im Folgenden interessieren wir uns jetzt nicht mehr für die Position der Pfeilspitze sondern lediglich für die resultierende Punktzahl.

(d) Modelliere dieses Zufallsexperiment, indem einen geeigneten *diskreten* Wahrscheinlichkeitsraumes angibst.

(e) Sei X die Zufallsvariable, die die Punktzahl widerspiegelt. Bestimme folgende Werte:

- Den Erwartungswert von X
- Die Varianz von X
- Den Erwartungswert von $X \cdot \mathbb{1}_{X \text{ is ungerade}}$. Das ist die erwartete Punktzahl einer Spielvariante wo nur ungerade Zahlen gewertet werden.
- Den Erwartungswert eines Wurfes der eine ungerade Punktzahl erzielt hat.

Lösung 1

(a) Die Ergebnismenge ist die Einheitskreisschreibe:

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

Die Wahrscheinlichkeitsverteilung ist die Gleichverteilung auf Ω. Mit anderen Worten: Das Wahrscheinlichkeitsmaß weist jeder Teilmenge von Ω mit Flächeninhalt A die Wahrscheinlichkeit A/π zu. Was ein Wahrscheinlichkeitsmaß formal ist, ist nicht Teil der Vorlesung.

(b) Gefragt ist nach folgenden Dingen.

- Die *Zufallsvariable* D, die den Abstand beschreibt. Es gilt $D((x, y)) = \sqrt{x^2 + y^2}$ für $(x, y) \in \Omega$.
- Die *Zufallsvariable* P, die die Punktzahl beschreibt. Wenn wir mit $A_1, A_2, \ldots, A_6 \subseteq \Omega$ die Segmente der Dartscheibe im mathematisch positiven Sinn durchnummerieren, dann gilt:

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

- Das Ereignis G, dass die Punktzahl gerade ist, können wir nun auf verschiedene äquivalente Arten hinschreiben. Hier sind drei Vorschläge:

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

Beachte: Im mittleren Fall haben die geschweiften Klammern nicht die gewöhnliche Semantik von Mengenklammern, sondern zeigen an, dass hier ein Ereignis definiert wird.

- Die Zufallsvariable $G(\omega) = P(\omega) \bmod 2$ ist besonders insofern als sie nur die Werte 0 und 1 annehmen kann. Solche Zufallsvariablen nennt man Indikatorzufallsvariablen. Es gilt $\mathbb{E}[G] = \Pr[G = 1]$.

(c) Für $d \leq 0$ gilt $\Pr[D \leq d] = 0$. Für $d \geq 1$ gilt $\Pr[D \leq d] = 1$. Für $0 \leq d \leq 1$ betrachten wir das Ereinis $E_d = \{(x, y) \in \Omega \mid x^2 + y^2 \leq d^2\}$. Dieses hat einen Flächeninhalt von $d^2\pi$. Also:

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

(d) $\Omega = \{1, 2, 3, 4, 5, 6\}$ mit Gleichverteilung.

(e) Gefragt war nach folgenden Größen:

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

Aufgabe 2 – Analogien zu den Rechenregeln

Sei Ω die Menge der Bewohner des fernen Landes Omegon. Betrachte folgende vier Aussagen und erkenne zu welcher der fünf Rechenregeln der Folien die Aussage analog ist. Überlege dir zur verbleibenden Rechenregel selbst eine Verbildlichung. Argumentiere (formal oder intuitiv, wie du möchtest) warum die Rechenregeln gelten.

1. Sei h der Anteil der Hundebesitzer, k der Anteil der Katzenbesitzer und t der Anteil der Bewohner mit Hund oder Katze. Dann gilt: $t \leq h + k$.

2. Angenommen 40% der Bewohner leben im Westen, der Rest im Osten. Wenn g_1 die Durchschnittsgröße der Wessis ist und g_2 die Durchschnittsgröße der Ossis dann ist $g_1 \cdot 0.4 + g_2 \cdot 0.6$ die Durchschnittsgröße in Omegon.

3. Angenommen 40% der Bewohner leben im Westen, der Rest im Osten. Sei k_1 der Anteil der Katzenbesitzer unter Wessis und k_2 der Anteil der Katzenbesitzer unter Ossis. Dann beträgt der Anteil der Katzenbesitzer insgesamt $k = k_1 \cdot 0.4 + k_2 \cdot 0.6$.

4. Wenn ein Bewohner pro Jahr im Schnitt w weiße und b braune Hühnereier verspeist, dann verspeist ein Bewohner pro Jahr im Schnitt $w + b$ Hühnereier.

Lösung 2

1. Vereinigungsschranke / Union Bound. Um sie zu zeigen brauchen wir, dass für disjunkte Ereignisse A und B gilt: $\Pr[A \cup B] = \Pr[A] + \Pr[B]$ und dass Wahrscheinlichkeiten nicht negativ sind. Für zwei Ereignisse E_1, E_2 gilt dann:

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

   Für mehr als zwei Ereignisse kann man Induktion verwenden.

2. Satz vom totalen Erwartungswert. Hier ein informeller Beweis:

$$\begin{aligned}
&\sum_{i=1}^{n} \mathbb{E}[X \mid E_i] \cdot \Pr[E_i] \\
&\quad = \sum_{i=1}^{n} \sum_{x} x \cdot \Pr[X = x \mid E_i] \cdot \Pr[E_i] && \text{(Definition bedingter Erwartungswert)} \\
&\quad = \sum_{i=1}^{n} \sum_{x} x \cdot \Pr[\{X = x\} \cap E_i] && \text{(Definition bedingte Wahrscheinlichkeit)} \\
&\quad = \sum_{x} x \cdot \sum_{i=1}^{n} \Pr[\{X = x\} \cap E_i] = \sum_{x} x \cdot \Pr[X = x] && \text{(Disjunktheit } E_1, \ldots, E_n\text{)} \\
&\quad = \mathbb{E}[X] && \text{(Definition Erwartungswert)}
\end{aligned}$$

3. Satz von der totalen Wahrscheinlichkeit. Das ist ein Spezialfall vom Satz vom totalen Erwartungswert für eine Indikatorzufallsvariable $X = \mathbb{1}_F$, denn dann ist $\Pr[F] = \mathbb{E}[X]$ und $\Pr[F \mid E_i] = \mathbb{E}[X \mid E_i]$.

4. Linearität des Erwartungswertes. Die Aussage wird durch einen Beweis kaum klarer. Hier trotzdem ein Versuch. Für diskrete Wahrscheinlichkeitsräume ergibt sich durch herunterbrechen auf alle möglichen Ergebnisse:

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

5. Es fehlte: Tail Sum Formula. Analogie:

   > Jeder Bewohner ω schafft eine bestimmte Anzahl ℓ_ω Liegestütze. Sei z_j die Anzahl der Bewohner, die mindestens z_j Liegestütze schaffen. Dann gilt: $\sum_{\omega \in \Omega} \ell_\omega = \sum_{j \geq 1} z_j$ und damit auch $\frac{1}{|\Omega|} \sum_{\omega \in \Omega} \ell_\omega = \sum_{j \geq 1} \frac{z_j}{|\Omega|}$. Die linke Summe ist die mittlere Anzahl Liegestütze und $\frac{z_j}{|\Omega|}$ ist der Anteil der Bewohner, die j Liegestütze schaffen.

   Die Formel lässt sich durch vertauschen von Summen herleiten. Für beliebige nichtnegative reelle Zahlen $(x_{i,j})_{i,j \in \mathbb{N}}$ gilt: $\sum_{j \geq 1} \sum_{i=1}^{j} x_{i,j} = \sum_{i \geq 1} \sum_{j \geq i} x_{i,j}$, denn in beiden Fällen wird $x_{i,j}$ genau dann gezählt wenn $j \geq i$ gilt. Wir wenden das an für $x_{i,j} = \Pr[X = i]$.

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

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 2 – The Power of Randomness

Randomisierte Algorithmik

Aufgabe 1 – Polynomgleichungen überprüfen

Sei $\mathbb{F}$ ein Körper, zum Beispiel $\mathbb{F} = \mathbb{Q}$. Gegeben sei eine Polynomgleichung, zum Beispiel:

$$(x+1)(x-2)(x+3)(x-4)(x+5)(x-6) \stackrel{?}{=} x^6 - 7x^3 + 25$$

(a) Argumentiere: Für den Fall, dass die Beispielgleichung *nicht* gilt, gibt es höchstens 6 Werte für x, sodass auf beiden Seiten das gleiche herauskommt.

(b) Beschreibe einen randomisierten Algorithmus der entscheidet ob eine Polynomgleichung gilt oder nicht. Dieser darf falsche Polynomgleichungen mit kleiner Wahrscheinlichkeit als korrekt akzeptieren. Was lässt sich über diese Wahrscheinlichkeit sagen?

Lösung 1

(a) Die Gleichung hat die Form $f(x) = g(x)$ und lässt sich umstellen zu $f(x) - g(x) = 0$. Falls die Gleichung nicht gilt ist $f(x) - g(x)$ ein Polynom von Grad $0 \leq d \leq 6$. Ein solches Polynom hat höchstens 6 Nullstellen. Also gibt es höchstens 6 Werte für x für die $f(x) - g(x) = 0$ und damit $f(x) = g(x)$ gilt.

(b) Betrachten wir zunächst den Fall $|\mathbb{F}| < \infty$. Wir wählen dann $X \sim \mathcal{U}(\mathbb{F})$ zufällig und setzen X in die Gleichung ein. Falls die Polynomgleichung allgemein stimmt, dann auch für dieses X. Falls sie nicht stimmt, dann können wir wie in (a) den maximalen Grad d betrachten, der auf beiden Seiten vorkommt. Es gibt höchstens d Elemente von $\mathbb{F}$, für die auf beiden Seiten das gleiche herauskommt, und die Wahrscheinlichkeit, dass wir eines davon gewählt haben ist höchstens $d/|\mathbb{F}|$.

Falls $|\mathbb{F}| = \infty$ gibt es keine Gleichverteilung auf $\mathbb{F}$. Das ist einerseits etwas lästig, andererseits können wir X gleichverteilt aus einer großen Teilmenge $S \subseteq \mathbb{F}$ wählen. Dann wird die obere Schranke für die Fehlerwahrscheinlichkeit mit $d/|S|$ sogar beliebig klein.

Bemerkung: Es handelt sich um einen false-biased Monte-Carlo-Algorithmus (siehe Vorlesung zu Probability Amplification).

Aufgabe 2 – Matrixprodukte überprüfen[1]

Seien $\mathbb{F}$ ein Körper, $n \in \mathbb{N}$.

(a) Zeige: Falls $C, C' \in \mathbb{F}^{n \times n}$ zwei verschiedene Matrizen sind und $v \in \{0,1\}^n$ uniform zufällig gewählt wird, dann gilt: $\Pr[C \cdot v \neq C' \cdot v] \geq \frac{1}{2}$.

(b) Beschreibe einen Algorithmus der bei Eingabe $A, B, C \in \mathbb{F}^{n \times n}$ ein Bit X ausgibt mit $X = 1$ falls $A \cdot B = C$ und $\Pr[X = 1] \leq 1/2$ falls $A \cdot B \neq C$. Der Algorithmus soll nur $O(n^2)$ Körperoperationen durchführen.

Lösung 2

(a) Seien $C_1, \ldots, C_n \in \{0,1\}^n$ und $C'_1, \ldots, C'_n \in \{0,1\}^n$ die Spalten von C und C' sowie $v_1, \ldots, v_n \in \{0,1\}$ die Einträge von v. Sei ferner $i \in [n]$ ein Index mit $C_i \neq C'_i$, der nach Vorraussetzung existieren muss. Wir können nun schreiben:

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

Wir stellen uns nun vor, dass zunächst alle Einträge von v bis auf den $i$ten Eintrag gewählt werden und nur v_i noch zufällig ist. Nun gibt es zwei Fälle.

Fall 1. $w = 0$. In dem Fall führt $v_i = 1$ zu $D = C_i - C'_i \neq 0^n$.

Fall 2. $w \neq 0$. In dem Fall führt $v_i = 0$ zu $D = w \neq 0^n$.

In beiden Fällen führt also mindestens eine der beiden Möglichkeiten für v_i zu $D \neq 0^n$. Also gilt auch insgesamt $\Pr[C \cdot v \neq C' \cdot v] = \Pr[D \neq 0^n] \geq \frac{1}{2}$.

(b) Wir verwenden (a) mit $C' = A \cdot B$.

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

Offenbar ist $X = 1$ garantiert falls $A \cdot B = C$. Falls $A \cdot B \neq C$ gilt nach (a) $\Pr[X = 1] = \Pr[C \cdot v = C' \cdot v] \leq \frac{1}{2}$.

Die drei Matrix-Vektor Produkte lassen sich jeweils in $O(n^2)$ Körperoperationen ausführen. Insbesondere ist das schneller als ein Matrix-Matrix Produkt das (mit einem naiven Algorithmus) $\Omega(n^3)$ Körperoperationen benötigt.

Bemerkung: Es handelt sich um einen false-biased Monte-Carlo-Algorithmus (siehe Vorlesung zu Probability Amplification).

[1] Bekannt als Algorithmus von Freivald.

Aufgabe 3 – Deterministische Auswertung von $\bar{\wedge}$-Bäumen

Sei A ein deterministischer Algorithmus der einen Bitvektor $I \in \{0,1\}^n$ als Eingabe erhält (mit $n = 2^d$) und den Wert des vollständigen balancierten $\bar{\wedge}$-Baums berechnet, dessen Blätter gemäß I beschriftet sind. Zeige: Es gibt eine Eingabe $I_A \in \{0,1\}^n$, sodass A jede Blattbeschriftung inspizieren muss.

Lösung 3

Wir sind ein Widersacher des Algorithmus und legen den Wert eines Blattes erst dann fest, wenn es angefragt wird. Wir stellen sicher, dass er alle Blätter anfragen muss. Die Belegung der Blätter die sich ergibt ist dann die Worst-Case Anfrage I_A für A.

Wir zeigen die Aussage per Induktion. Für $d = 1$ (Blatt = Wurzel) ist nichts zu zeigen. Der Algorithmus muss das einzige Blatt natürlich anfragen.

Für $d > 1$ liegen zwei Unterbäume der Tiefe $d-1$ vor. In beiden benutzen wir die Strategie, die sich aus der Induktion ergibt. Damit kann sich das Ergebnis eines Unterbaumes erst dann herausstellen, wenn alle 2^{d-1} Blätter angefragt wurden. Weil das letzte Blatt eine Rolle für das Ergebnis des Unterbaums gespielt hat, können wir sogar im letzten Schritt noch aussuchen, dass der Unterbaum insgesamt zu 1 auswertet. Damit ist das Gesamtergebnis an der Wurzel $1 \bar{\wedge} b$ wobei b das Ergebnis des anderen Unterbaumes ist. Das muss der Algorithmus also auch noch bestimmen, und damit auch im zweiten Unterbaum alle Blätter anfragen.

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 3 – Important Random Variables and How to Sample Them

Randomisierte Algorithmik

Aufgabe 1 – Ber(1/3) aus Ber(1/2)

Entwerfe einen Algorithmus der, gegeben eine Folge $B_1, B_2, \ldots \sim \text{Ber}(1/2)$ von Zufallsbits in erwarteter Zeit $O(1)$ ein Sample $B \sim \text{Ber}(1/3)$ berechnet.

Lösung 1

Wir interpretieren $B_1, B_2, B_3, \ldots$ als Binärdarstellung einer Zahl $U = (0.B_1B_2B_3\ldots)_2$. Für diese gilt dann $U \sim \mathcal{U}([0,1])$. Wir definieren $B := \mathbb{1}_{U<1/3}$. Daraus folgt unmittelbar $B \sim \text{Ber}(1/3)$ wie gewünscht. Die Binärdarstellung von $1/3$ ist $1/3 = (0.01010101\ldots)_2$. So ergibt sich folgender Algorithmus, der immer die nächsten zwei Ziffern der Binärdarstellung von U hernimmt und schaut ob diese eine Entscheidung zulassen:

for $i = 1$ **to** ∞ **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

Jede Runde führt mit Wahrscheinlichkeit 3/4 zu einer Entscheidung. Wenn R die Anzahl von Runden ist, dann gilt

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

Bemerkung: In der Praxis würde man das nicht so machen, sondern wie in der nächsten Aufgabe davon ausgehen, dass man $U \sim \mathcal{U}([0,1])$ direkt sampeln kann (so genau wie es Floating-Point Zahlen eben hergeben).
Bemerkung: Die Laufzeit ist unbeschränkt und das ist auch unvermeidbar. Das können wir durch einen Widerspruchsbeweis zeigen. Angenommen ein Algorithmus kommt für ein festes $C \in \mathbb{N}_0$ stets mit dem Präfix $B_1, \ldots, B_C$ der Zufallsfolge aus. Dann ist seine Ausgabe B eine Zufallsvariable $B : \Omega \to \{0,1\}$, die auf dem Wahrscheinlichkeitsraum $\Omega = \{0,1\}^C$ mit Gleichverteilung definiert ist (dessen Ergebnisse sind alle Möglichkeiten für $B_1, \ldots, B_C$). Jedes Ergebnis hat eine Wahrscheinlichkeit von 2^{-C}. Jedes Ereignis – und damit auch das

Ereignis $\{B = 1\}$ ist eine Menge von Ergebnissen und hat damit eine Wahrscheinlichkeit, die ein ganzzahliges Vielfaches von 2^{-C} ist. Das steht im Widerspruch dazu, dass diese Wahrscheinlichkeit $1/3$ sein soll.

Aufgabe 2 – $\mathrm{Ber}(p)$ und $\mathcal{U}(\{1, \dots, n\})$ aus $\mathcal{U}([0, 1])$

Wir nehmen nun ein Maschinenmodell an in dem reelle Zahlen verarbeitet werden können und in dem wir $U \sim \mathcal{U}([0, 1])$ sampeln können. Zeige, dass wir auch $B \sim \mathrm{Ber}(p)$ für $p \in [0, 1]$ und $X \sim \mathcal{U}(\{1, \dots, n\})$ für $n \in \mathbb{N}$ sampeln können.

Hinweis: Für den Rest des Blattes und des Semesters nehmen wir Voraussetzung und Ergebnis dieser Aufgabe als gegeben an.

Lösung 2

Gegeben $U \sim \mathcal{U}([0, 1])$ definieren wir $B := \mathbb{1}_{U<p}$ und $X := \lceil U \cdot n \rceil$. Dann gilt wie gewünscht:

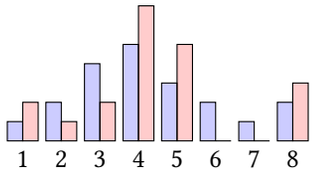
$$\Pr[B = 1] = \Pr[U < p] = p \text{ sowie}$$
$$\text{für } 1 \le i \le n\text{: } \Pr[X = i] = \Pr\left[U \cdot n \in (i-1, i]\right] = \Pr\left[U \in (\tfrac{i-1}{n}, \tfrac{i}{n}]\right] = \tfrac{1}{n}.$$

Bemerkung: Streng genommen ist für $U \sim \mathcal{U}([0, 1])$ das Ergebnis $U = 0$ möglich was zu $X = 0$ führt, obwohl eigentlich $X \in \{1, \dots, n\}$ gelten soll. Das passiert allerdings mit Wahrscheinlichkeit 0. Man kann das reparieren indem man zum Beispiel sagt, dass 0 einfach auch zu 1 aufgerundet wird. Oder man kehrt solche nervigen Spitzfindigkeiten einfach unter den Teppich.

Aufgabe 3 – Rejection Sampling Allgemein

Seien $\mathcal{D}_1$ und $\mathcal{D}_2$ Verteilungen auf einer endlichen Menge D. Wir nehmen an:

1. Wir können in Zeit $O(1)$ ein Sample $X \sim \mathcal{D}_1$ erzeugen.
2. Wir können wir ein gegebenes $x \in D$ in Zeit $O(1)$ die Wahrscheinlichkeiten $p_1(x) := \Pr_{X \sim \mathcal{D}_2}[X = x]$ und $p_2(x) := \Pr_{X \sim \mathcal{D}_1}[X = x]$ berechnen.
3. Es existiert $C > 0$ sodass für alle $x \in D$ gilt:
$$p_2(x) \le C \cdot p_1(x).$$



Mögliches Histogramm für $\mathcal{D}_1$ (blau, links) und $\mathcal{D}_2$ (rot, rechts). Es gilt stets "rot $\le 2 \cdot$ blau", also ist Bedingung (3) mit $C = 2$ erfüllt.

Entwerfe einen Algorithmus, der in erwarteter Zeit $O(C)$ ein Sample $Y \sim \mathcal{D}_2$ erzeugt.

Lösung 3

Der Algorithmus geht folgendermaßen:

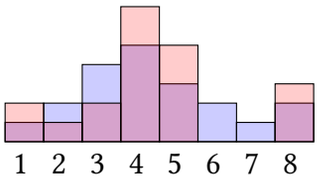
while True **do**
  sample $X \sim \mathcal{D}_1$ // $O(1)$
  sample $U \sim \mathcal{U}([0,1])$ // $O(1)$
  if $U < \frac{p_2(X)}{C \cdot p_1(X)}$ **then** // $O(1)$
    return X

Wir überprüfen zunächst seine Korrektheit. Wichtig ist hierbei, dass $\frac{p_2(X)}{p_1(X) \cdot C} \in [0,1]$ gilt, was aus Vorraussetzung (3) folgt. Sei Y das Ergebnis eines Schleifendurchgangs, das heißt $Y = X$ falls X zurückgegeben wird und $Y = \perp$ falls die Schleife ohne Ergebnis endet. Dann gilt für $x \in D$:

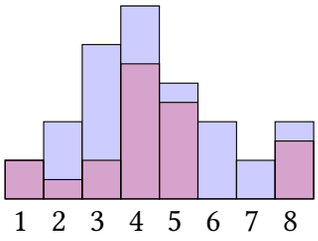
$$\Pr[Y = x] = \Pr[X = x] \cdot \Pr\left[U < \frac{p_2(x)}{C \cdot p_1(x)}\right] = p_1(x) \cdot \frac{p_2(x)}{C \cdot p_1(x)} = \frac{p_2(x)}{C}.$$

Insbesondere ist die Wahrscheinlichkeit $\Pr[Y = x]$ *proportional* zu $p_2(x)$. Damit gilt $\Pr[Y = x \mid Y \neq \perp] = p_2(x)$. In anderen Worten: Wenn etwas zurückgegeben wird, dann liegen die Wahrscheinlichkeiten von $\mathcal{D}_2$ zugrunde, wie gewünscht. Die Erfolgswahrscheinlichkeit einer Runde ist $\Pr[Y \neq \perp] = \sum_{x \in D} \Pr[Y = x] = \frac{1}{C}$. Damit sind im Erwartungswert C Runden nötig bis ein Erfolg eintritt.

Intuition: Man kann sich den Algorithmus auch verbildlichen. Wenn wir die Balken der Histogramme übereinander malen ergibt sich folgendes:



Wenn wir nun die blauen Balken hochskalieren, dann erhalten wir ein Bild in dem die roten Balken stets kleiner sind als die blauen.



Um aus der roten Verteilung zu ziehen genügt es, einen zufälligen roten Punkt zu ziehen und den Index des Balkens zurückzugeben in dem er liegt. Dazu ziehen wir einen zufälligen blauen Punkt (in der Darstellung: blau oder lila ist) und behalten ihn, falls er rot ist.

Im Algorithmus wird ein zufälliger blauer Punkt gezogen indem wir einen Balken X wählen und dann eine zufällige Höhe $U \cdot C \cdot p_1(X)$ auf diesem Balken. Das wird dann mit der Höhe des roten Balkens verglichen.

Aufgabe 4 – $G \sim \mathrm{Geom}_1(p)$ mit Inverse Transform Sampling

Entwerfe einen Algorithmus der für gegebenes $p \in (0, 1]$ eine Zufallsvariable $G \sim \mathrm{Geom}_1(p)$ in Zeit $O(1)$ sampelt.

Lösung 4

Die kumulative Verteilungsfunktion von G ist:

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

Für die (verallgemeinerte) Inverse gilt dann für $u \in (0, 1]$:

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

Nach der Methode sollte also folgendes funktionieren:

sample $U \sim \mathcal{U}([0, 1])$
return $G = \left\lceil \frac{\log(1-U)}{\log(1-p)} \right\rceil$

Dass alles geklappt hat können wir auch nochmal überprüfen in dem wir nachrechnen, dass das G aus dem Algorithmus die gewünschte Verteilungsfunktion hat:

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

Aufgabe 5 – Ziehen ohne Zurücklegen

Wir betrachten Algorithmen, die für $k, n \in \mathbb{N}$ mit $0 \leq k \leq n/2$ eine Menge $S \subseteq [n]$ der Größe k berechnen, die aus allen Teilmengen von $[n]$ der Größe k uniform zufällig gewählt ist.

(a) Warum dürfen wir $k \leq n/2$ ohne Beschränkung der Allgemeinheit annehmen?

(b) Beschreibe einen Algorithmus, der eine erwartete Laufzeit von $O(k \log k)$ hat.
Hinweis: Rejection Sampling und Suchbaum.

(c) **Bonus:** Entwerfe einen Algorithmus, der eine Worst-Case Laufzeit von $O(k \log k)$ hat.

(d) **Bonus:** Recherchiere, wie man eine Worst-Case Laufzeit von $O(k)$ erreichen kann:

`https://stackoverflow.com/a/67850443`

Lösung 5

(a) $S \subseteq [n]$ ist eine zufällige Menge der Größe k genau dann, wenn $[n] \setminus S$ eine zufällige Menge der Größe $n - k$ ist.

(b) Von der Idee her zieht der Algorithmus *mit* Zurücklegen, speichert sich die Ergebnisse in einem Suchbaum und ignoriert Ergebnisse, die schonmal aufgetreten sind. Das macht er bis er k verschiedene Ergebnisse gesehen hat. Das ist eine Form von Rejection Sampling und es ist ziemlich klar, dass das korrekt ist.

Algorithm ZieheOhneZurücklegen(n, k)**:**
- $S \leftarrow \emptyset$ // als Suchbaum
- **while** $|S| < k$ **do**
  - sample $X \sim \mathcal{U}(\{1, \ldots, n\})$
  - **if** $X \notin S$ **then**
    - $S \leftarrow S \cup \{X\}$
- **return** S

Nach Annahme aus (a) und der Schleifenbedingung gilt am Anfang jedes Schleifendurchlaufs $|S| < k \leq n/2$. Damit ist die Wahrscheinlichkeit, etwas zu ziehen, was wir schon haben, stets höchstens $1/2$. Daraus folgt, dass die Anzahl F erfolgloser Schleifendurchläufe erwartet höchstens der Anzahl erfolgreicher Schleifendurchläufe entspricht, also gilt $\mathbb{E}[F] \leq k$.

Die Gesamtlaufzeit beträgt $T = (k + F) \cdot O(\log k)$ weil es $k + F$ Schleifendurchläufe gibt und jede mit Suchbaumoperationen in $O(\log k)$ durchführbar ist. Somit ergibt sich $\mathbb{E}[T] = O(k \log k)$.

(c) Die Idee ist, dass wir die Menge der noch ziehbaren Elemente explizit verwalten. Im Folgenden wird Array$[1..n]$ verwendet, was stets eine Permutation der Menge $\{1, \ldots, n\}$ enthält. Am Anfang von Schleifendurchlauf i enthält Array$[1..i-1]$ die bereits gezogenen Elemente und Array$[i..n]$ die noch ziehbaren Elemente.

Algorithm ZieheOhneZurücklegen(n, k)**:**
- Array $= [1, 2, \ldots, n]$ // alles noch ziehbar
- **for** $i = 1$ **to** k **do**
  - sample $j \sim \mathcal{U}(\{i, \ldots, n\})$
  - swap Array$[j]$ and Array$[i]$ // tut nichts falls $j = i$
- **return** Array$[1..k]$

Dummerweise ergibt sich so eine Laufzeit von $O(n + k)$ wegen der Initialisierung des Arrays. Das lässt sich aber reparieren. Offenbar kann stets für höchstens $2k$ Indizes i gelten, dass Array$[i] \neq i$ ist. Es genügt also sich diese Ausnahmepositionen in einem Suchbaum zu speichern. Dann erhält man Laufzeit $O(k \log k)$.

(d) Siehe `https://github.com/ciphergoth/sansreplace/blob/master/cardchoose.md`

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 4 – Probability Amplification

Randomisierte Algorithmik

Aufgabe 1 – Probability Amplification mit zweiseitigem Fehler

Angenommen ein Monte-Carlo Algorithmus A beantwortet ein Entscheidungsproblem mit Wahrscheinlichkeit $1/2 + \varepsilon$ korrekt und sonst falsch (für ein $\varepsilon > 0$). Sei A' der Algorithmus der t mal unabhängig A ausführt und sich für die häufigere Antwort entscheidet. Zeige, dass die Fehlerwahrscheinlichkeit von A' höchstens $e^{-2t\varepsilon^2}$ ist.

Hinweise: Für die Rechnung könnte nützlich sein: $\sum_{i=0}^{t} \binom{t}{i} = 2^t$.

Lösung 1

Sei $I \in \{0, \dots, t\}$ die Anzahl der Ausführungen, die das richtige Ergebnis liefern. Damit A' falsch antwortet muss $I \leq t/2$ gelten. Die Lösung ergibt sich durch geschicktes Abschätzen:

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

Aufgabe 2 – Pfadfinder

Gegeben sei ein ungerichteter Graph G mit n Knoten und m Kanten. Gesucht ist ein Pfad der Länge k, der keinen Knoten mehrfach besucht. Der naive Brute-Force Ansatz hat eine Laufzeit von $O(n^k)$. Wir betrachten einen einfachen, randomisierten Ansatz, der wie folgt funktioniert. Im ersten Schritt wird jedem Knoten v ein Label $L(v)$ zufällig gleichverteilt aus $\{1, \dots, k\}$ zugewiesen. Im zweiten Schritt wird von jedem Knoten v mit $L(v) = 1$ eine modifizierte Breitensuche gestartet, bei der ein Knoten w nur dann von einem Knoten u entdeckt werden kann, falls $L(u) = L(w) + 1$ gilt. Wird ein Knoten mit Label k entdeckt wird ein Pfad der Länge k konstruiert und ausgegeben.

(a) Zeige, dass der Algorithmus mit Wahrscheinlichkeit $1/k^k$ einen Pfad der Länge k findet, falls einer existiert.

(b) Verbessere die Erfolgswahrscheinlichkeit auf $1 - 1/n$ mit Probability Amplification. Welche Gesamtlaufzeit ergibt sich?

Bemerkung: Die Idee heißt "Color Coding". Es gibt noch raffiniertere Varianten.

Lösung 2

(a) Sei $P = (v_1, \ldots, v_k)$ ein Pfad der Länge k in G. Mit Wahrscheinlichkeit $1/k^k$ wird für jedes $i \in [k]$ der Knoten v_i mit Farbe i gefärbt. In dem Fall wird die Breitensuche die Knoten $v_1, \ldots, v_k$ alle erreichen (entlang von P oder entlang eines andere Pfades) und somit einen Pfad der Länge k finden.

(b) Wir führen den Algorithmus $t = k^k \cdot \ln n$ mal aus. Die Erfolgswahrscheinlichkeit beträgt dann wie gewünscht:

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

Da n Breitensuchen in Zeit $O(nm)$ durchgeführt werden können, ergibt sich insgesamt eine Laufzeit von $O(nm \cdot k^k \cdot \ln n)$.

Aufgabe 3 – Bonus: Random-Walk Löser für 3-SAT

Sei $\varphi(x_1, \ldots, x_n)$ eine 3-SAT Formel mit n Variablen und m Klauseln. Wir suchen eine Lösung mit folgendem Algorithmus:

Algorithm randomWalkSolver(φ)**:**
- 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**
  - sei C eine beliebige unerfüllte Klausel von φ
  - sample $j \sim \mathcal{U}(\{1, 2, 3\})$
  - sei x_i die $j$te 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** $\perp$

Falls φ unerfüllbar ist, gilt offenbar randomWalkSolver(φ) $= \perp$. Andernfalls sei x^* eine Lösung. Zeige:

(a) Mit Wahrscheinlichkeit $\geq 1/2$ stimmt die initial gesampelte Belegung $(x_1, \ldots, x_n)$ an mindestens $n/2$ Stellen mit x^* überein.

(b) Falls $\varphi(x_1, \ldots, x_n) = 0$ so führt eine Iteration der Schleife mit Wahrscheinlichkeit $1/3$ dazu, dass eine weitere Variable von $(x_1, \ldots, x_n)$ so wie in x^* gesetzt wird.

(c) Verwende Probability Amplification um einen Algorithmus mit Erfolgswahrscheinlichkeit $1 - 1/n$ zu erhalten. Was ist die Gesamtlaufzeit?

Lösung 3

(a) Für jede "schlechte" Belegung mit weniger als $n/2$ Übereinstimmungen hat die invertierte Belegung mehr als $n/2$ Übereinstimmung. Damit machen die "schlechten" Belegungen höchstens einen Anteil von $1/2$ aus. (Die Belegungen mit exakt $n/2$ Übereinstimmungen führen bei geradem n zu einem Bias zu unseren Gunsten).

(b) In der unerfüllten Klausel C, die im Schleifendurchlauf betrachtet wird, kommt mindestens eine Variable vor, die in x^* anders belegt ist als in $(x_1, \ldots, x_n)$, schließlich erfüllt x^* ja C. Mit Wahrscheinlichkeit $1/3$ wählen wir diese Variable aus.

(c) Zunächst ergibt sich aus (a) und (b), dass die Erfolgswahrscheinlichkeit bei einmaliger Ausführung mindestens $\frac{1}{2} \cdot 3^{-n/2}$ beträgt (intuitiv: wir starten in der Nähe von x^* und bewegen uns nur darauf zu). Wiederholen wir den Algorithmus $t = 2 \cdot 3^{n/2} \cdot \ln n$ mal, so ergibt sich eine Erfolgswahrscheinlichkeit von

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

Wenn m die Anzahl von Klauseln in φ ist, dann hat randomWalkSolver eine Laufzeit von $O(nm)$. Insgesamt ergibt sich $O(3^{n/2} \cdot nm)$. Das ist potentiell besser als ein naiver Algorithmus mit Laufzeit $O(2^n)$.

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 5 – Randomised Complexity Classes

Randomisierte Algorithmik

Aufgabe 1 – Beziehungen zwischen Komplexitätsklassen

Begründe folgende Inklusionen:

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

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

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

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

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

Lösung 1

(i) Gilt nach Definition von $\mathbf{ZPP} := \mathbf{RP} \cap \text{co}-\mathbf{RP}$.

(ii) Sei $L \in \mathbf{P}$. Zu zeigen ist $L \in \mathbf{ZPP}$. Sei T eine $\mathbf{P}$-DTM für L. Wir nutzen das „typecasting“ der Vorlesung. Aus T wird somit formal eine RTM (die nicht mehr wirklich randomisiert ist), die immernoch L entscheidet und immernoch polynomielle Laufzeit hat. Insbesondere bezeugt diese RTM, dass $L \in \mathbf{RP}$ ist. Weil $\mathbf{P} = \text{co}-\mathbf{P}$ gilt, ist $\bar{L} \in \mathbf{P}$ und nach dem Argument von eben daher auch $\bar{L} \in \mathbf{RP}$. Also gilt $L \in \text{co}-\mathbf{RP}$. Zusammen ergibt sich $L \in \mathbf{RP} \cap \text{co}-\mathbf{RP} = \mathbf{ZPP}$.

(iii) Sei $L \in \mathbf{RP}$ und T eine $\mathbf{RP}$-PTM für L. Durch „Vergessen“ der Wahrscheinlichkeiten entsteht eine NTM T'. Weil für jedes $w \in L$ galt, dass $\Pr[T(w) = \text{YES}] \geq \frac{1}{2}$, existiert insbesondere eine akzeptierende Berechnung für w. Also gilt $T'(w) = \text{YES}$. Für jedes $w \notin L$ galt, dass $\Pr[T(w) = \text{YES}] = 0$. Also existiert keine akzeptierende Berechnung für w. Also gilt $T'(w) = \text{NO}$. Also wird L von T' entschieden. Also gilt $L \in \mathbf{NP}$. Weil L beliebig war folgt $\mathbf{RP} \subseteq \mathbf{NP}$.

Für den symmetrischen Fall können wir nun auch schlussfolgern:

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

(iv) Sei $L \in \mathbf{RP}$ und T eine **RP**-PTM für L. Wenn T' die PTM ist, die T dreimal hintereinander ausführt (mit unabhängigem Zufall) und akzeptiert, wenn mindestens eine der Berechnungen von T akzeptiert, dann gilt für alle Wörter $w \in L$:

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

Für $w \in L$ ergibt sich:

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

Für $w \notin L$ ergibt sich:

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

Also ist T' eine **BPP**-PTM für L. Somit gilt $L \in \mathbf{BPP}$. Weil L beliebig war folgt $\mathbf{RP} \subseteq \mathbf{BPP}$.

Der symmetrisch Fall ist wieder analog weil $\mathbf{BPP} = \text{co}-\mathbf{BPP}$ gilt.

(v) Hier ist überhaupt nichts zu tun. Jede **BPP**-PTM ist auch eine **PP**-PTM, weil die Anforderungen lediglich heruntergeschraubt werden. Also gibt es für jede Sprache L, für die es eine **BPP**-PTM gibt, auch eine **PP**-PTM.

Aufgabe 2 – Las Vegas Algorithmus für L impliziert $L \in$ ZPP

Sei **LV** (für Las Vegas) die Klasse aller Sprachen L, für die eine probabilistische Turingmaschine T mit folgenden Eigenschaften existiert:

- T entscheidet L. (Das heißt L liefert für alle $x \in L$ stets die Ausgabe 1 und für alle $x \notin L$ stets die Ausgabe 0.)
- Es gibt ein Polynom $p(n)$, sodass die *erwartete* Laufzeit von T bei Eingabe x durch $p(|x|)$ beschränkt ist.

In der Vorlesung haben wir bewiesen, dass $\mathbf{ZPP} \subseteq \mathbf{LV}$ gilt. Zeige nun, dass auch $\mathbf{LV} \subseteq \mathbf{ZPP}$ gilt. die beiden Klassen sind also identisch.

Hinweis: Definition von **ZPP**, Markov-Ungleichung.

Lösung 2

Sei $L \in \mathbf{LV}$. Wir zeigen zunächst $L \in \mathbf{RP}$. Sei nun also T eine **LV**-TM für L mit zugehörigem Polynom $p(n)$. Wir betrachten folgende Turingmaschine T':

Algorithm $T'(w)$**:**

$t_{\max} \leftarrow 2p(|w|)$
simuliere T auf Eingabe w für $t_{\max}$ Schritte
if *T hat mit Ausgabe r terminiert* **then**
  return r
else // T hat noch nicht terminiert
  return NO

Wir zeigen nun, dass T' eine **RP**-TM für L ist. Durch die Verwendung von $t_{\max}$ ist klar, dass sich die Laufzeit von $T'(w)$ durch ein Polynom $q(|w|)$ beschränken lässt. Zu den Ausgaben von T' is zu sagen:

- Für $w \notin L$ ist die Ausgabe von $T'(w)$ stets NO, entweder, weil T das so entschieden hat oder weil wir im else-Fall angekommen sind. Also: $\Pr[T'(w) = \text{YES}] = 0$.
- Für $w \in L$ betrachten wir die zufällige Laufzeit $t(w)$ von T auf w. Nach Vorraussetzung gilt $\mathbb{E}[t(w)] \leq p(|w|)$. Mit der Markov Ungleichung folgt:

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

  Also terminiert T auf w mit Wahrscheinlichkeit mindestens $1/2$ innerhalb des gesetzten Zeitlimits (mit dem richtigen Ergebnis). Also gilt $\Pr[T'(w) = \text{YES}] \geq \frac{1}{2}$.

Also ist T' eine **RP**-TM für L. Also gilt $L \in \mathbf{RP}$. Analog folgt auch $L \in \text{co}-\mathbf{RP}$ (indem man die Ausgabe bei nicht-Terminierung auf YES setzt) und somit $L \in \mathbf{ZPP}$. Weil L beliebig war gilt $\mathbf{LV} \subseteq \mathbf{ZPP}$.

Aufgabe 3 – Bonus: Probability Amplification für BPP

Recherchiere auf Wikipedia, was es mit der Komplexitätsklasse **P/poly** auf sich hat. Zeige, dass **BPP** $\subseteq$ **P/poly** gilt. Diese Einsicht heißt auch Adlemans Satz.

Hinweis: Mache die Fehlerwahrscheinlichkeit kleiner als 2^{-n}. „Caste“ die PTM dann in eine DTM und zeige, dass es einen Zufallsstrings gibt, der für alle Eingaben zum richtigen Ergebnis führt.

Lösung 3

Keine Musterlösung für Bonusaufgabe.

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 6 – Concentration Bounds

Randomisierte Algorithmik

Aufgabe 1 – Rechenregeln für Varianz

Seien X, Y unabhängige Zufallsvariablen deren Varianz existiert. Seien ferner $s, t > 0$. Zeige, dass gilt:

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

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

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

Hinweis: Nutze $\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$. Wenn du Lust hast, kannst du das auch nochmal aus der Definition von Unabhängigkeit (für diskrete Zufallsvariablen) herleiten, die lediglich $\Pr[X = i \wedge Y = j] = \Pr[X = i] \cdot \Pr[Y = j]$ für alle i, j garantiert.

Lösung 1

Wir zeigen zunächst nochmal $\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$. Seien hierfür $R_X, R_Y \subseteq \mathbb{R}$ abzählbare Mengen, die alle möglichen Werte für X und Y enthalten. Dann gilt:

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

Wir beweisen nun die drei einfachen Rechenregeln (und machen das sehr ausführlich).

(a) Hier ist neben der Definition der Varianz nur die Einsicht $\mathbb{E}[sZ] = s\mathbb{E}[Z]$ nötig (folgt aus Linearität des Erwartungswertes). Letzteres gilt für jede Zufallsvariable Z, deren Erwartungswert existiert und jedes $s \in \mathbb{R}$. Also:

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

(b) Zunächst haben zentrierte Zufallsvariablen Erwartungswert 0:

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

Wenn zwei Zufallsvariablen X und Y unabhängig sind, dann sind auch $X - c$ und $Y - d$ unabhängig für beliebige Konstanten $c, d \in \mathbb{R}$. Wenn wir $c = \mathbb{E}[X]$ und $d = \mathbb{E}[Y]$ setzen folgt, dass $X - \mathbb{E}[X]$ unabhängig von $Y - \mathbb{E}[Y]$ ist. Also folgt:

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

Wir betrachten nun den Term $(X + Y - \mathbb{E}[X + Y])^2$, dessen Erwartungswert die gesuchte Varianz ist.

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

Wenn wir nun den Erwartungswert auf beiden Seiten nehmen erhalten wir:

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

wobei wir im letzten Schritt die Definition von Varianz sowie Gleichung (2) verwenden.

(c) Wir verwenden, dass für unabhängige Zufallsvariablen X und Y auch sX und tY unabhängig sind. Damit folgt unmittelbar aus (a) und (b):

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

Aufgabe 2 – Chernoff in noch einfacher für starke Abweichung

Sei $X = X_1 + \cdots + X_n$ eine Summe von unabhängigen Bernoulli-Zufallsvariablen mit $\mu = \mathbb{E}[X]$ und sei $b \geq 6\mu$. Zeige

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

Hinweis: Benutze die Chernoff Schranke $\Pr[X \geq (1 + \delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$.

Lösung 2

Um dem Hinweis zu folgen setzen wir $\delta = b/\mu - 1$. Es ergibt sich:

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

Aufgabe 3 – Vergleich von Konzentrationsschranken

Für $n \in \mathbb{N}$ sei X_n die Anzahl der Sechsen bei n-maligem Würfeln eines gewöhnlichen Würfels. Sei p_n die Wahrscheinlichkeit, dass X_n seinen Erwartungswert um mindestens 10% übersteigt. Finde jeweils eine Schranke an p_n mit…

(a) … der Markov-Ungleichung.

(b) … der Tschebyscheffschen Ungleichung (engl. „Chebyshev's Inequality").

(c) … der Chernoff-Ungleichung (oder einer Variante davon).

(d) Vergleiche die asymptotische Stärke der Schranken.

Lösung 3

Vorbereitungen:

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

(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) Asympototisch ist die Chernoff-Schranke die stärkste und die Markov-Schranke die schwächste.
Bermerkung: Während Markov und Chernoff für alle $n \in \mathbb{N}$ eine Schranke mit Wert < 1 liefert, greift Tschebyscheff erst ab $n = 501$.

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 7 – Classic Hash Tables

Randomisierte Algorithmik

Aufgabe 1 – 2–Unabhängigkeit vs. 1-Universalität

Sei $\mathcal{H} \subseteq [m]^D$ eine Familie von Hashfunktionen von D nach $[m]$. Zeige oder widerlege, dass folgende Implikationen gelten:

(a) $\mathcal{H}$ ist 2-unabhängig $\Rightarrow$ $\mathcal{H}$ ist 1-universell.

(b) $\mathcal{H}$ ist 1-universell $\Rightarrow$ $\mathcal{H}$ ist 2-unabhängig.

Hinweis: In einem Fall lässt sich die Implikationen leicht zeigen. Im anderen Fall kann man dämliche Gegenbeispiele finden.

Lösung 1

(a) Die Implikation ist richtig. Für jedes $x \neq y \in D$ gilt nach Definition von 2-Unabhängigkeit:

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

Damit können wir für $\mathcal{H}$ die Kollisionswahrscheinlichkeit von x und y folgendermaßen abschätzen:

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

Damit ist die Bedingung für 1-Universalität gezeigt.

(b) Die Implikation gilt nicht. Allerdings vor allem aus „dummen“ Gründen.

Beispiel 1. Man nehme $D = [m]$ und $\mathcal{H} = \{\mathrm{id}\}$. Natürlich erzeugt die Identitätsfunktion niemals Kollisionen, also ist $\mathcal{H}$ sogar 0-universell (und damit auch 1-universell). Natürlich ist $\mathcal{H}$ aber nicht 2-unabhängig, weil die Hashwerte nicht gleichverteilt in $[m]$ liegen (die sind ja nicht mal zufällig).

Beispiel 2. Man nehme die Klasse $\mathcal{H} = \mathcal{H}^{\text{lin}}_{p,m}$ aus der Vorlesung mit Parametern p und m, sodass m kein Teiler von $p \cdot (p-1)$ ist. Nach Vorlesung ist $\mathcal{H}$ eine 1-universelle Klasse. Weil $|\mathcal{H}| = p \cdot (p-1)$ ist, sind alle relevanten Wahrscheinlichkeiten (alle Zahlen der Form $\Pr_{h \sim \mathcal{H}}[\ldots]$) aber Vielfache von $\frac{1}{p \cdot (p-1)}$. Nun ist aber $\frac{1}{m}$ kein solches Vielfaches. Also kann $\Pr_{h \sim \mathcal{H}}[h(x) = 0] = \frac{1}{m}$ für kein x gelten. Der Hashwert von x ist also nicht gleichverteilt in $[m]$.

Aufgabe 2 – d-Unabhängigkeit ohne Unabhängigkeit

Alice und Bob drehen beide an einem Glücksrad, das 10 gleich große Segmente mit den Zahlen von 0 bis 9 hat. Seien A und B die Ergebnisse von Alice und Bob. Sei $C = (A + B) \bmod 10$.

(a) Zeige: A, B, C sind paarweise unabhängig.

(b) Zeige: A, B, C sind nicht unabhängig.

(c) Finde für beliebiges $d \in \mathbb{N}$ eine Familie von Zufallsvariablen, die d-unabhängig ist aber nicht unabhängig ist.

Lösung 2

Wir lösen die Aufgabe direkt für beliebiges $d, m \in \mathbb{N}$ (statt für $d = 2$ und $m = 10$). Das heißt wir haben d unabhängige Zufallsvariablen $A_1, A_2, \ldots, A_d \sim \mathcal{U}([m])$ und eine weitere Zufallsvariable $C := (A_1 + \cdots + A_d) \bmod m$. Es ist leicht zu sehen, dass auch $C \sim \mathcal{U}([m])$ gilt indem man sich vorstellt, dass $A_1, \ldots, A_d$ nacheinander gewählt werden: Ganz egal was für $A_1, \ldots, A_{d-1}$ herausgekommen ist, die m Möglichkeiten die noch für A_d bleiben machen noch jeden Wert in $[m]$ für C mit gleicher Wahrscheinlichkeit möglich. Wir zeigen zwei Eigenschaften:

Die Familie $\{A_1, \ldots, A_d, C\}$ ist nicht unabhängig. Wir haben $\Pr[\forall i \in [d] : A_i = 0] = m^{-d}$ sowie $\Pr[C = 0] = m^{-1}$. Gleichzeitig impliziert aber das erste Ereignis das zweite (wenn alle $A_i = 0$ sind, dann auch $C = 0$). Also gilt $\Pr[C = 0 \wedge \forall i \in [d] : A_i = 0] = m^{-d}$. Wäre $\{A_1, \ldots, A_d, C\}$ eine unabhängige Familie von Zufallsvariablen hätte $m^{-(d+1)}$ herauskommen müssen.

Die Familie $\{A_1, \ldots, A_d, C\}$ ist d-fach unabhängig. Wir betrachten eine beliebige Auswahl von d der Variablen und das Ereignis, dass diese eine bestimmte Kombination von Werten annehmen. Es ist zu zeigen, dass die Wahrscheinlichkeit für dieses Ereignis sich wie erwartet als Produkt von Wahrscheinlichkeiten ergibt. Vernachlässigt man symmetrische Fälle (die Variablen $A_1, \ldots, A_d$ haben alle die gleiche Rolle) hat man nur zwei Fälle zu unterscheiden. Entweder C ist nicht ausgewählt oder C ist ausgewählt:

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

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

Die „$\overset{!}{=}$“ sind zu zeigen. Im ersten Fall ist das klar, weil $A_1, \ldots, A_d$ schon als unabhängig definiert sind. Es bleibt der zweite Fall. Wir betrachten also ein Ereignis der Form:

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

wobei $a_1, \ldots, a_{d-1}, c$ feste Zahlen sind. Dieses Ereignis lässt sich aber nach Definition von C auch äquivalent schreiben als:

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

Weil wir aber die Werte von A_1 bis A_{d-1} bereits vorschreiben ist das aquivalent zu:

$$E = \{A_1 = a_1 \wedge \ldots \wedge A_{d-1} = a_{d-1} \wedge A_d = c - a_1 - a_2 - \ldots - a_{d-1}\}.$$

In dieser Form ist klar, dass E Wahrscheinlichkeit m^{-d} hat, weil $A_1, \ldots, A_d$ nach Definition unabhängig und gleichverteilt sind.

Ein Nachtrag. Schaut man sehr genau auf unsere Definitionen, könnte man noch eine Sorge haben. Wir haben zwar oben gezeigt, dass egal welche d Variablen wir wählen, diese d Variablen zulassen, dass wir Wahrscheinlichkeiten von zusammengesetzten Ereignisse als Produkte auseinanderziehen. Die Definition von d-Unabhängigkeit spricht aber von „bis zu“ d Variablen. Was ist also, wenn wir k Variablen wählen, für ein $k < d$? Folgt dann automatisch dass diese k Variablen auch unabhängig sind? Die Antwort ist „ja“.

Betrachten wir beispielsweise das Ereignis:

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

Wir müssen zeigen, dass für dieses E gilt $\Pr[E] \stackrel{!}{=} \Pr[C = c] \cdot \prod_{i=1}^{k-1} \Pr[A_i = a_i] = m^{-k}$. Das geht hier indem man einfach zusätzliche Fallunterscheidungen über andere Zufallsvariablen macht und das Ergebnis von vorher verwendet:

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

Aufgabe 3 – Konzentrationsschranken für Summen d-unabhängiger Zufallsvariablen

Sei $d \in \mathbb{N}$ gerade und $\{X_1, \ldots, X_n\}$ eine d-unabhängige Familie von Zufallsvariablen, die alle Verteilung $\mathrm{Ber}(p)$ mit $p = \Omega(1/n)$ haben. Wir betrachten die Summe $X = \sum_{i=1}^{n} X_i$. Beachte: Weil $X_1, \ldots, X_n$ nicht unabhängig sind ist X nicht unbedingt binomialverteilt!

Ziel dieser Aufgabe ist es, eine Konzentrationsschranke für X zu zeigen, nämlich, dass für beliebige $\delta > 0$ gilt:

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

Dazu schauen wir die „zentrierten“ Zufallsvariablen $Y_i := X_i - p$ an, deren Summe $Y = \sum_{i=1}^{n} Y_i$ und den Erwartungswert $\mathbb{E}[Y^d]$.

(i) Zum Aufwärmen: Sei $d \geq 3$ und $n \geq 3$. Überzeuge dich, dass Folgendes gilt und erkläre kurz warum.

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

Im weiteren Verlauf darfst du die zugrundeliegenden Einsichten ohne weitere Begründung verallgemeinert verwenden.

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

(iii) Seien $i_1, \ldots, i_d \in [n]$ (nicht notwendig verschieden) sowie $S = \{i_1, \ldots, i_d\}$. Zeige:

- Falls $|S| > d/2$ dann gilt $\mathbb{E}[Y_{i_1} \cdot \ldots \cdot Y_{i_d}] = 0$.
- Andernfalls gilt $\mathbb{E}[Y_{i_1} \cdot \ldots \cdot Y_{i_d}] \leq p^{|S|}$.

(iv) Zeige $\mathbb{E}[Y^d] = O((np)^{d/2})$. Du darfst annehmen, dass $d = O(1)$ gilt.
Hinweis: Multipliziere $(\sum_{i=1}^n Y_i)^d$ aus. Ja, das ergibt n^d Terme.

(v) Beweise das ursprüngliche Ziel dieser Aufgabe indem du die Markov Ungleichung auf Y^d anwendest.

Lösung 3

(i) Wegen $d \geq 3$ gilt für beliebige verschiedene $i_1, i_2, i_3 \in [n]$, dass $X_{i_1}, X_{i_2}, X_{i_3}$ unabhängig sind. Insbesondere sind X_1, X_2, X_3 unabhängig. Weil sich Y_1^5, Y_2^{42} und Y_3 jeweils als Funktion aus X_1,X_2 bzw. X_3 ergeben sind auch Y_1^5, Y_2^{42}, Y_3 unabhängig.

(a) Der Erwartungswert des Produktes unabhängiger Zufallsvariablen ist das Produkt der Erwartungswerte dieser Zufallsvariablen nach Definition von Unabhängigkeit.

(b) Auch hier kann die Erwartungswerte auseinanderziehen und dann $\mathbb{E}[Y_3] = \mathbb{E}[X_3 - p] = \mathbb{E}[X_3] - p = p - p = 0$ verwenden.

(c) Weil $|Y_1| \leq 1$ ist und x^i für $x \in [0, 1]$ monoton fallend in i ist folgt:

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

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

(iii) Die zentrale Frage für diesen Aufgabenteil ist, ob es einen Index gibt, der in der Liste $i_1, \ldots, i_d$ nur einmal vorkommt.

- $|S| > d/2$ bedeutet, dass die Liste der d Indizes mehr als $d/2$ verschiedene Werte abdeckt, also muss es mindestens einen Index j geben, der nicht doppelt vorkommt. Dann hat der Erwartungswert $\mathbb{E}[Y_{i_1} \cdot \ldots \cdot Y_{i_d}]$ die Form wie in Aufgabenteil (i) (b). Nach dem Auseinanderziehen kommt also $\mathbb{E}[Y_j] = 0$ als Faktor vor.

- $|S| \leq d/2$ heißt, dass höchstens $d/2$ verschiedene Variablen im Produkt vorkommen. Die kann man wie in Aufgabenteil (i) (a) auseinanderziehen. Wenn eine Variable mit Potenz 1 vorkommt, bekommt man wieder 0 als Gesamtergebnis. Andernfalls ist jede Potenz mindestens 2. Man kann dann Aufgabenteil (i) (c) sowie (ii) verwenden, um jeden der $|S|$ Terme mit p abzuschätzen.

(iv) Wir rechnen drauf los und kommentieren unten die einzelnen Schritte:

$$\begin{aligned}
\mathbb{E}[Y^d] &= \mathbb{E}\left[\left(\sum_{i=1}^{n} Y_i\right)^d\right] = \mathbb{E}\left[\sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\cdots\sum_{i_d=1}^{n} Y_{i_1}\cdot Y_{i_2}\cdot\ldots\cdot Y_{i_d}\right] \\
&\overset{(1)}{=} \sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\cdots\sum_{i_d=1}^{n}\mathbb{E}[Y_{i_1}\cdot Y_{i_2}\cdot\ldots\cdot Y_{i_d}] \\
&\overset{(2)}{=} \sum_{i_1,\ldots,i_d}\mathbb{E}[Y_{i_1}\cdot Y_{i_2}\cdot\ldots\cdot Y_{i_d}] \overset{(3)}{=} \sum_{r=1}^{d}\sum_{\substack{S\subseteq[n]\\|S|=r}}\sum_{i_1,\ldots,i_d}\mathbb{1}_{\{i_1,\ldots,i_d\}=S}\cdot\mathbb{E}[Y_{i_1}\cdot Y_{i_2}\cdot\ldots\cdot Y_{i_d}] \\
&\overset{(4)}{\leq} \sum_{r=1}^{d/2}\sum_{\substack{S\subseteq[n]\\|S|=r}}\sum_{i_1,\ldots,i_d}\mathbb{1}_{\{i_1,\ldots,i_d\}=S}\cdot p^{|S|} \overset{(5)}{=} \sum_{r=1}^{d/2}\sum_{\substack{S\subseteq[n]\\|S|=r}} p^{|S|}\sum_{i_1,\ldots,i_d}\mathbb{1}_{\{i_1,\ldots,i_d\}=S} \\
&\overset{(6)}{\leq} \sum_{r=1}^{d/2}\sum_{\substack{S\subseteq[n]\\|S|=r}} p^{|S|}|S|^d \overset{(7)}{=} \sum_{r=1}^{d/2}\binom{n}{r}p^r r^d \overset{(8)}{\leq} (d/2)^d\sum_{r=1}^{d/2} n^r p^r \overset{(9)}{\leq} O(n^{d/2}p^{d/2}).
\end{aligned}$$

(1) Linearität des Erwartungswertes.

(2) Kompaktere Schreibweise.

(3) Gruppierung der Terme nach der Menge $S = \{i_1, \ldots, i_d\}$.

(4) Nach (iv) sind die Terme mit $|S| > d/2$ gleich 0, können also weggelassen werden. Die übrigen sind höchstens $p^{|S|}$.

(5) Ausklammern.

(6) Damit die $\mathbb{1}$-Variable 1 liefern kann müssen alle $i_1, \ldots, i_d$ in S gewählt werden (das ist notwendig, aber nicht hinreichend). Dafür gibt es $|S|^d$ Möglichkeiten.

(7) Die innere Summe hängt nicht mehr von S ab sondern nur noch von $r = |S|$. Es gibt $\binom{n}{r}$ Summenglieder.

(8) Wir verwenden $\binom{n}{r} \leq n^r$ und $r \leq d/2$.

(9) Wenn $d = O(1)$ dann ist auch $(d/2)^d = O(1)$ und fällt weg. Wegen $p = \Omega(1/n)$ ist $pn = \Omega(1)$. Also dominiert der Summand mit $r = d/2$ die anderen (konstant vielen) Summanden.

(v) Wieder erst eine Rechnung, dann die Begründungen:

$$\Pr[X - \mathbb{E}[X] \geq \delta\mathbb{E}[X]] \overset{(1)}{=} \Pr[Y \geq \delta np] \leq \Pr[|Y| \geq \delta np] = \Pr[|Y|^d \geq (\delta np)^d]$$
$$\overset{(2)}{=} \Pr[Y^d \geq (\delta np)^d] \overset{(3)}{\leq} \frac{\mathbb{E}[Y^d]}{(\delta np)^d} \overset{(4)}{\leq} O\left(\frac{n^{d/2}p^{d/2}}{(\delta np)^d}\right) = O\left(\delta^{-d}(np)^{-d/2}\right).$$

(1) Verwendet Definition von Y und Erwartungswert von X.

(2) Weil d gerade ist, können die Betragsstriche weggelassen werden.

(3) Markov Ungleichung angewendet auf Y^d. Beachte: Weil d gerade ist gilt $Y^d \geq 0$.

(4) Das Resultat der letzten Teilaufgabe.

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 8 – Bounded Differences and Bloom Filters

Randomisierte Algorithmik

Aufgabe 1 – Bälle, Behälter und beschränkte Differenzen

Sei $\lambda > 0$ eine Konstante, $m = \lambda n$ eine Anzahl von Bällen und n eine Anzahl von Behältern. Der j-te Ball werde in Behälter X_j platziert wobei $X_1, \dots, X_m \sim \mathcal{U}([n])$ unabhängige Zufallsvariablen sind. Die Kollisionszahl ist definiert als $C = |\{(i,j) \mid 1 \le i < j \le n, X_i = X_j\}|$. Ziel dieser Aufgabe ist es, eine Konzentrationsschranke an C zu bestimmen.

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

Für $i \in [n]$ sei $L_i = |\{j \in [m] \mid X_j = i\}|$ der Füllstand von Behälter i. Es ist weder schwer noch interessant zu zeigen, dass $\Pr[\max_{i\in[n]} L_i \le \log n] \ge 1 - O(1/n)$. Nimm das im Folgenden als gegeben hin (Beweis in der Musterlösung).

(ii) Definiere $C_{\text{cap}} := \sum_{i\in[n]} \binom{\min(\log n, L_i)}{2}$. Zeige $\Pr[C_{\text{cap}} = C] = 1 - O(1/n)$.

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

(iv) Zeige $\Pr[C - \mathbb{E}[C] \ge n^{2/3}] = O(1/n)$.

Lösung 1

(i) Je zwei verschiedene Bälle kollidieren mit Wahrscheinlichkeit $1/n$. Also gilt $\mathbb{E}[C] = \binom{m}{2} \cdot \frac{1}{n} = \frac{m(m-1)}{2n} = \frac{\lambda(m-1)}{2} = \Theta(n)$.

Wir schauen uns nun die behauptete Schranke an die Füllstände an. Seien zunächst $i \in [n]$ und $k \in \mathbb{N}$ beliebig aber fest. Für eine Menge $S \subseteq [m]$ der Größe k sei $E_{S,i}$ das Ereignis, dass alle Bälle aus S in Behälter i platziert werden. Es gilt $\Pr[E_{S,i}] = n^{-k}$. Dann folgt:

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

Jetzt muss man nur noch $k = \log n$ einsetzen und verwenden, dass dann $\alpha^k = \text{poly}(n)$ sowie $m = \text{poly}(n)$ gilt, aber $(\log n)!$ schneller als jedes Polynom wächst.

(ii) Falls $\max_{i\in[n]} L_i \leq \log n$ gilt so folgt

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

Letzeres ist aber eine alternative Definitionsmöglichkeit für die Kollisionszahl C (summiere Kollisionen innerhalb jedes Behälters).

(iii) C_{cap} ergibt sich als Funktion aus den m unabhängigen Zufallsvariablen $X_1, \ldots, X_m$. Verändert sich eines der X_j, so kann sich C_{cap} um höchstens $\log n$ verändern. Das Ergebnis ergibt sich durch direkte Anwendung von McDiarmid's Ungleichung.
Bemerkung: Auf C selbst lässt sich die Strategie nicht gut anwenden. Der Extremfall ist, dass $X_1 = \ldots = X_{m-1} = 1$ und $X_m = 2$ gilt. Ändert man X_m auf 1, so entstehen $m - 1$ zusätzliche Kollsionen.

(iv) Wir setzen die vorherigen beiden Teilergebnisse zusammen:

$$\begin{aligned}
\Pr[C - \mathbb{E}[C] \geq n^{2/3}] &\leq \Pr[C \neq C_{\text{cap}} \vee C_{\text{cap}} - \mathbb{E}[C] \geq n^{2/3}] \\
&\overset{\text{UB}}{\leq} \Pr[C \neq C_{\text{cap}}] + \Pr[C_{\text{cap}} - \mathbb{E}[C] \geq n^{2/3}] \\
&\overset{C \geq C_{\text{cap}}}{\leq} \Pr[C \neq C_{\text{cap}}] + \Pr[C_{\text{cap}} - \mathbb{E}[C_{\text{cap}}] \geq n^{2/3}] \\
&\leq O(1/n) + \exp(-2n^{4/3}/(m \log^2 n)) = O(1/n).
\end{aligned}$$

Folgende Aufgabe hat nicht wirklich mit randomisierten Algorithmen zu tun, außer dass wir die Abschätzung in der Vorlesung gebraucht haben. Daher "Bonus".

Aufgabe 2 – Bonus: Approximationen von e

Du weißt sicher, dass $e = \lim_{n\to\infty}(1 + \frac{1}{n})^n$ gilt (das ist sogar eine gängige *Definition* von e). Leite daraus ab, dass die folgenden beiden Gleichungen für alle $n \in \mathbb{N}$ gelten:

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

Folgende Schritte bieten sich an.

(i) Zeige die linken beiden Ungleichungen.
Hinweis: Benutze mal wieder $1 + x \leq e^x$.

(ii) Zeige, dass die rechten Seiten monoton in n fallen.

(iii) Folgere aus (ii) und einer Grenzwertbetrachtung die rechten beiden Ungleichungen.

Lösung 2

(i) Es gilt jeweils:

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

(ii) Zunächst erhalten wir durch Logarithmieren des Hinweises für (i):

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

Um zu zeigen, dass $f(n) = (1+\frac{1}{n})^{n+1}$ monoton in $n \in \mathbb{N}$ fällt genügt es zu zeigen, dass $\ln(f(x)) = (x+1)\ln(1+\frac{1}{x})$ monoton in $x \in (0,\infty)$ fällt. Dazu betrachten wir die Ableitung und zeigen, dass diese überall kleinergleich 0 ist:

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

Analog argumentieren wir für $g(x) = (1-\frac{1}{n})^{n-1}$ dass die Ableitung von $\ln(g(x))$ kleinergleich 0 ist:

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

(iii) Es gilt durch Separierung eines Faktors:

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

Den Grenzwert von $(1-\frac{1}{n})^{n-1}$ können wir folgendermaßen bestimmen:

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

Also konvergieren die rechten Seite gegen e bzw. e^{-1}. Das tun sie nach (iii) „von oben“. Also gelten die Ungleichungen wie behauptet.

Aufgabe 3 – Counting Bloomfilter (a.k.a.: Count-Min-Sketch)

Ein Counting Bloomfilter ist zunächst wie ein Bloomfilter: Er verwaltet n Schlüssel in einem Array der Größe m, der *load factor* ist $\alpha := \frac{n}{m}$ und es gibt k Hashfunktionen. Die Parameter seien wie in der Vorlesung gewählt, sodass ein (gewöhnlicher) Bloomfilter eine Falsch-Positiv-Wahrscheinlichkeit von ε hätte. Das Array enthält nun keine Bits mehr sondern natürliche Zahlen. Neben insert und query gibt es nun als weitere Operation delete.

Algorithm insert(x):
```
for i ∈ [k] do
    A[h_i(x)] + +
```

Algorithm delete(x):
```
for i ∈ [k] do
    A[h_i(x)] − −
```

Algorithm query(x):
```
for i ∈ [k] do
    if A[h_i(x)] = 0 then
        return false
return true
```

Wir lassen zu, dass ein Schlüssel *mehrfach* in den Counting Bloomfilter eingefügt wird. Insofern ist das verwaltete Objekt S nun eine *Multimenge*, deren n Elemente $\{x_1, \ldots, x_n\}$ jeweils mit einer Vielfachheit $a_1, \ldots, a_n$ vorliegen. Die Operation insert(x) soll der Multimenge S eine Kopie von x hinzufügen, das heißt die Vielfachheit von x erhöhen, falls x bereits in S enthalten war oder ein weiteres Element mit Vielfachheit 1 in S aufnehmen, falls x noch nicht in S enthalten war. Die Bedeutung von delete ist analog.

Wie zuvor auch darf query falsch-positive, aber keine falsch-negativen Antworten liefern.

(a) Wir fordern, dass delete nur für solche Elemente aufgerufen wird, die auch wirklich in S enthalten sind (mit Vielfachheit mindestens 1). Was geht kaputt wenn wir das nicht fordern?

Mit Counting Bloomfiltern lassen sich noch zusätzliche nützliche Operationen implementieren.

(b) Implementiere eine Operation count, die für $x \in D$ einen Schätzwert count(x) für die Vielfachheit a von x in S angibt. Zeige, dass $\Pr[a \neq \text{count}(x)] \leq \varepsilon$ gilt.

(c) Das wichtigste Argument dafür (Counting-) Bloomfilter zu verwenden, ist der geringe Speicherplatz im Vergleich zu einer exakten Datenstruktur. Wir sollten daher besser nicht große Integer Datentypen (etwa 64 Bit) für die Zähler verwenden. Stellen wir uns einen Anwendungsfall vor, in dem die „meisten“ Zähler niemals 8 Bit überschreiten. Wir nutzen daher ein Array A von 8-Bit Zählern. Diskutiere (kurz) die folgenden Vorschläge zum Umgang mit Zählerüberläufen. Was sind die Vorteile und welche Kompromisse werden eingegangen?

- Alice verhindert lediglich Zählerüberläufe, passt also die $A[h_i(x)] + +$ Operation in insert und die $A[h_i(x)] - -$ Operation in delete so an, dass der Zähler nur inkrementiert bzw. dekrementiert wird, wenn der maximale darstellbare Wert bzw. der minimale darstellbare Wert noch nicht erreicht ist.
- Bob schlägt vor, einen Zähler, der den Wert $(11111111)_2 = 255$ erreicht hat „einzufrieren“, das heißt zukünftige insert oder delete Operationen werden den Zähler nie mehr anpassen.
- Carol schlägt vor, die Bitfolge $(11111111)_2 = 255$ als Markierung dafür zu verwenden, dass der wahre Zählerwert Größer als 254 ist. Der wahre Zählerwert wird in diesem Sonderfall in einer Hashtabelle gespeichert.

Lösung 3

(a) Ein delete(y) für ein y, das niemals eingefügt reduziert unkontrolliert k Zähler im Counting Bloomfilter. So können Zähler 0 werden. Dadurch könnte es sein, dass für ein $x \in S$ anschließend query(x) = false gilt. Also wird für x eine falsch-negative Antwort gegeben, was wir nicht wollen.

(b) Wir geben das Minimum der mit x assoziierten Zähler zurück:

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

Zunächst fällt auf, dass $\text{count}(x) < a$ unmöglich ist, denn jedes Vorkommen von x wird von jedem mit x assoziierten Zähler gezählt. Es könnte allerdings sein, dass $\text{count}(x) > a$ gilt, wenn alle k Zähler zusätzlich von anderen Schlüsseln verwendet werden.

Um dies zu verstehen sei $A'[1..m] \in \{0,1\}^m$ der gewöhnliche Bloomfilter, in den die Elemente aus S mit Ausnahme von x eingefügt wurden (mit den selben Hashfunktionen, wie für den Counting Bloomfilter). Dann gilt:

$$
\begin{aligned}
\text{count}(x) \neq a &\Leftrightarrow \text{count}(x) > a \\
&\Leftrightarrow \min_{i \in [k]} A[h_i(x)] > a \\
&\Leftrightarrow \forall i \in [k] : A[h_i(x)] > a \\
&\Leftrightarrow \forall i \in [k] : \text{es gibt einen Schlüssel in } S \text{ außer } x \text{ der } A[h_i(x)] \text{ verwendet} \\
&\Leftrightarrow \forall i \in [k] : A'[h_i(x)] = 1 \\
&\Leftrightarrow x \text{ ist falsch-positives Element für } A'
\end{aligned}
$$

Das letzte Ereignis hat nach Wahl der Konfigurationsparameter höchstens Wahrscheinlichkeit ε. Also gilt dies auch für das dazu äquivalente erste Ereignis.

(c) Zu den Vorschlägen ist folgendes zu sagen:

- Der Vorschlag von Alice ist einfach zu implementieren, allerdings werden falsch-negative Antworten möglich. Am einfachsten sieht man das an der Operationsfolge, die 256 mal den selben Schlüssel x einfügt und diesen anschließend 255 mal wieder löscht. Die letzte Einfügung geht verloren und die Löschungen machen alle relevanten Zähler wieder zu 0. Nun würde query(x) als Ergebnis falsch zurückgeben, obwohl der Schlüssel noch einmal in der Datenstruktur sein müsste. Doof.
- Bei Bobs Vorschlag kann ein Zähler niemals fälschlich zurück auf Null fallen, also kann es keinen falsch-negativen Antworten geben. Ein Nachteil ist, dass man eingefrorene Zähler niemals wieder los wird. Langfristig könnte daher die falsch-positiv Wahrscheinlichkeit steigen.

- Carols Vorschlag macht keine Kompromisse bei der Funktionalität. Die zusätzliche Hashtabelle kostet aber selbstverständlich Platz und Zugriffe darauf brauchen Zeit.

Aufgabe 4 – Schnitte Schätzen mit Bloomfiltern

Seien $n, m, k \in \mathbb{N}$. Alice und Bob wollen abschätzen, wie ähnlich ihr Musikgeschmack ist. Seien X die n Lieblingslieder von Alice und Y die n Lieblingslieder von Bob. Zu schätzen ist $\gamma := \frac{|X \cap Y|}{n} \in [0, 1]$. Beide gehen folgendermaßen vor.

- Alice konstruiert einen Bloomfilter $A[1..m] \in \{0,1\}^m$ für X unter Verwendung von k Hashfunktionen $h_1, \ldots, h_k$.
- Bob konstruiert einen Bloomfilter $B[1..m] \in \{0,1\}^m$ für Y unter Verwendung *derselben* k Hashfunktionen.
- Alice und Bob tauschen ihre Filter aus und berechnen $\delta := \frac{|\{i \in [m] | A[i] \neq B[i]\}|}{m}$.
- Alice und Bob berechnen basierend auf δ eine Schätzung $\bar{\gamma}$ für γ.

Löse folgende Teilaufgaben:

(a) Diskutiere: Welche Vor- und Nachteile könnte das Verfahren im Vergleich zum direkten Austausch von X und Y haben?

(b) Gewinne Intuition: Welche Werte von δ erwartest du (in etwa) für die Extremfälle, in denen $\gamma = 1$ bzw. $\gamma = 0$ gilt?
Hinweis: Du darfst hier und im Folgenden davon ausgehen, dass den Bloomfiltern eine „optimale“ Konfiguration mit $\alpha k = \ln(2)$ zugrundegelegt wurde.

(c) Berechne $\mathbb{E}[\delta]$ als Funktion von γ. Du darfst hierbei Terme niederer Ordnung unter den Tisch fallen lassen, also z.B. $(1 - \frac{1}{m})^m \approx e^{-1}$ schreiben, ohne ein $o(1)$ mitzuführen.
Hinweis: Zunächst scheint es, als könnten andere Parameter (z.B. $n, m, k, \alpha, \varepsilon$) auch eine Rolle spielen. Deren Einfluss verschwindet aber in Termen niederer Ordnung.

(d) Diskutiere: Welche Konzentrationsschranke eignet sich, um zu beweisen, dass δ mit hoher Wahrscheinlichkeit nahe an $\mathbb{E}[\delta]$ liegt?

(e) Stelle die Gleichung aus (c) um, sodass ersichtlich wird, wie eine Schätzung $\bar{\gamma}$ für γ aus δ berechnet werden kann.

(f) Spekuliere: Welche Rolle spielt die Wahl von k (bzw. von ε) im vorliegenden Kontext?

Lösung 4

(a)
- Vorteil: Der Platzverbrauch ist unter Umständen geringer.
- Nachteil: Wir können zwar erreichen, dass $\bar{\gamma}$ mit hoher Wahrscheinlichkeit in der Nähe von γ liegt, aber wir können γ nicht fehlerfrei berechnen.

- Vorteil: Die Elemente der Mengen X und Y werden nicht bekanntgegeben, d.h. Alice kann für jedes $x \in X$ abstreiten dass $x \in X$ gilt, ohne dass jemand sie der Lüge überführen kann.

(b) Für $\gamma = 1$ gilt offensichtlich $A[i] = B[i]$ für alle i und damit $\delta = 0$. Für $\gamma = 0$ haben die beiden Bloomfilter nichts miteinander zu tun und sind unabhängig. Laut Vorlesung sind in einem Bloomfilter mit $\alpha k = \ln(2)$ etwa die Hälfte der Einträge 1 und die andere Hälfte ist 0. Plausibel ist also, dass für alle $i \in [m]$ gilt: $\Pr[A[i] \neq B[i]] \approx \Pr_{C,D\sim\mathrm{Ber}(1/2)}[C \neq D] = 1/2$. Wir erwarten also $\delta \approx 1/2$.

(c) Wir schreiben kurz $h(z) := \{h_1(z), \ldots, h_k(z)\}$. Beachte: Ereignisse, die sich auf verschiedene Schlüssel oder verschiedene Hashfunktionen beziehen können wir wegen Unabhängigkeit „auseinanderziehen“. Im Folgenden nutzen wir x_0 für ein beliebiges Element in X und y_0 für ein beliebiges Element in $Y \setminus X$.

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

(d) Die Methode der beschränkten Differenzen bzw. McDiarmids Ungleichung ist hier geeignet. Die $k \cdot |X \cup Y|$ relevanten Hashwerte sind alle unabhängig. Verändert man einen davon dann ändert sich der Wert von δ um höchstens $\pm\frac{1}{m}$. Die Analyse der Vorlesung bzgl. der Konzentration von Z (Anzahl Nuller) überträgt sich problemlos.

(e) Aus (c) haben wir $\mathbb{E}[\delta] = 1 - (\frac{1}{2})^{(1-\gamma)}$. Wir entfernen das „$\mathbb{E}$“ (weil wir nur δ zur Verfügung haben, nicht $\mathbb{E}[\delta]$) und ersetzen γ durch $\bar{\gamma}$ (weil wir also nicht γ ausrechnen können sondern nur eine Schätzung dafür). Umstellen von $\delta = 1 - (\frac{1}{2})^{(1-\bar{\gamma})}$ ergibt: $\bar{\gamma} = 1 - \log_2(1/(1-\delta))$.

(f) Je größer k ist, desto stärker wird die Konzentrationsschranke und desto besser wird entsprechend die Schätzung $\bar{\gamma}$. Es spricht aber wenig dagegen einfach $k = 1$ zu verwenden. Es ist sogar denkbar $k \in (0, 1)$ zu wählen mit der Bedeutung, dass ein Schlüssel nur mit Wahrscheinlichkeit k eine Position zugeteilt bekommt und mit Wahrscheinlichkeit $1 - k$ einfach verworfen wird. Diese Zufallsentscheidungen müssten widerum durch eine Hashfunktion getroffen werden, die Alice und Bob beide kennen. Mit $k = \Theta(\frac{1}{n})$ könnte man

einen Speicherbedarf erreichen der nicht mehr von n abhängt. Ähnlich wie bei Approximationsalgorithmen könnte man einen relativen Fehler und eine Fehlerwahrscheinlichkeit einführen und ausrechen wie groß k in Abhängigkeit dieser beiden Parameter gewählt werden müsste.

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 9 – Coupling, Balls into Bins and Poissonisation

Randomisierte Algorithmik

Aufgabe 1 – Coupling einer Irrfahrt

Seien $X_1, X_2, \ldots \sim \mathcal{U}(\{-1, 1\})$ unabhängige Zufallsvariablen. Für $n \in \mathbb{N}_0$ sei $W_n := \sum_{i=1}^{n} X_i$. Man nennt $(W_n)_{n\in\mathbb{N}_0}$ eine Irrfahrt (engl. *random walk*).

Wir können auch eine verschobene Irrfahrt $(V_n)_{n\in\mathbb{N}_0}$ betrachten mit $V_n := W_n + 42$, die also Startpunkt $V_0 = 42$ statt $W_0 = 0$ hat. Wir wollen zeigen, dass die Wahl des Startpunktes typischerweise langfristig keine Rolle spielt.

Wir wollen dabei ohne Beweis verwenden, dass die Irrfahrt mit Wahrscheinlichkeit 1 jede ganze Zahl mindestens einmal besucht. Insbesondere gilt $\lim_{n\to\infty} \Pr[\max\{W_1, \ldots, W_n\} < c] = 0$ für alle $c \in \mathbb{N}$.

(i) Seien $S_1, S_2, \ldots \subseteq \mathbb{Z}$ beliebige Mengen. Zeige $\lim_{n\to\infty} |\Pr[W_n \in S_n] - \Pr[V_n \in S_n]| = 0$.
Hinweis: Konstruiere ein Coupling $(W'_n, V'_n)_{n\in\mathbb{N}_0}$ von $(W_n)_{n\in\mathbb{N}_0}$ und $(V_n)_{n\in\mathbb{N}_0}$ für das $\lim_{n\to\infty} \Pr[W'_n = V_{n'}] = 1$ gilt.

(ii) Zeige, dass das Ergebnis von Aufgabenteil (i) für eine Verschiebung von 43 statt 42 nicht in dieser Form gilt.

Lösung 1

(i) Wir verwenden $(W'_n) = (W_n)$ und beschreiben (V'_n) in natürlicher Sprache. Zunächst verhalte sich (V'_n) genau gegenläufig zu (W_n), verwendet also die invertierten Beiträge $-X_1, -X_2, -X_3 \ldots$ usw. Sei nun $T = \min\{t \in \mathbb{N} \mid W_n = 21\}$. Dann gilt $W_T = 21$ und $V'_T = 42 - 21 = 21$, das heißt die Irrfahrten begegnen sich zum Zeitpunkt T. Ab diesem Zeitpunkt verhalte sich (V'_n) genau wie (W_n), verwende also die selben Beiträge $X_{T+1}, X_{T+2}, \ldots$.

Es ist ziemlich klar, dass $(V'_n)_{n\in\mathbb{N}_0} \overset{\mathrm{d}}{=} (V_n)_{n\in\mathbb{N}_0}$ gilt, denn die Beiträge, die wir akumulieren sind nach wie vor unabhängige Zufallsvariablen und gleichverteilt in $\mathcal{U}(\{-1, 1\})$ (ob wir X_i addieren oder subtrahieren legen wir fest noch bevor wir den Wert X_i kennen). Also haben wir ein gültiges Coupling. In diesem Coupling gilt die Implikation $W_n \neq V'_n \Rightarrow T \geq n$.

Wir machen nun eine Hilfsrechnung für beliebige Zufallsvariablen X, Y und beliebige Mengen S.

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

Wenden wir dies nun auf $S = S_n$, $X = W_n$ und $Y = V'_n$ an, so ergibt sich:

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

wobei der letzte Schritt den Hinweis für $c = 21$ verwendet.

(ii) So wie wir die Irrfahrt definiert haben, "vergisst" diese zwar ihren konkreten Startpunkt aber nicht die Parität dieses Startpunktes. Mit anderen Worten, wenn wir $S_n := S := 2 \cdot \mathbb{Z}$ definieren, dann sind Irrfahrten immer abwechselnd in S und nicht in S. Für eine Verschiebung von 23 hätten wir dann $|\Pr[W_n \in S_n] - \Pr[V_n \in S_n]| = 1$ für alle $n \in \mathbb{N}$.

Aufgabe 2 – Coupling und Total Variation Distance

Seien X und Y zwei Zufallsvariablen mit Werten in $\mathbb{N}$. Der Totalvariationsabstand (engl. *total variation distance*) von X und Y (bzw. deren Verteilungen) ist definiert als[1]

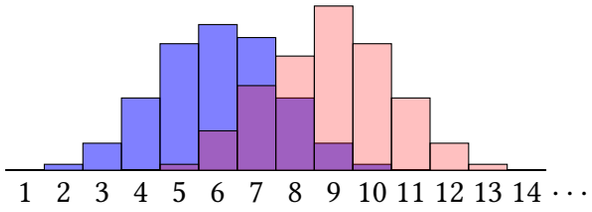
$$d(X, Y) = \frac{1}{2} \sum_{i \in \mathbb{N}} |\Pr[X = i] - \Pr[Y = i]|.$$

(i) Zeige: Es gibt ein Coupling (X', Y') von X und Y sodass $\Pr[X' \neq Y'] = d(X, Y)$.

(ii) Zeige: Kein Coupling (X', Y') von X und Y erfüllt $\Pr[X' \neq Y'] < d(X, Y)$.

Lösung 2

Zur Vorbereitung stellen wir uns ein gemeinsames Histogramm von X (blau) und Y (rot) vor.



[1] Eine allgemeine Definition, die auch für kontinuierliche Wahrscheinlichkeitsräume funktioniert, findet man auf Wikipedia.

Alle Balken mögen Breite 1 habe. Wir bezeichnen mit rot, blau und lila die Menge der Punkte der entsprechenden Farben und mit A_{rot}, A_{blau} und A_{lila} die zugehörigen Flächeninhalte. Weil die Balken Verteilungen beschreiben gilt $A_{\text{blau}} + A_{\text{lila}} = 1$ sowie $A_{\text{rot}} + A_{\text{lila}} = 1$, also folgt $A_{\text{blau}} = A_{\text{rot}}$. Beide sind jeweils genau der Totalvariationsabstand $d(X, Y)$. Das sieht man so:

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

(i) Wir sampeln zunächst ein Paar (P, Q) von Punkten folgendermaßen

- sample $P \sim \mathcal{U}(\text{blau} \cup \text{lila})$
- falls $P \in \text{lila}$ setze $Q = P$
- andernfalls sample $Q \sim \mathcal{U}(\text{rot})$.

Es sollte klar sein, dass damit $Q \sim \mathcal{U}(\text{rot} \cup \text{lila})$ gilt. Wir definieren nun X' als den Index des Balkens in dem P liegt und Y' als den Index des Balkens in dem Q liegt. Nun sollte klar sein, dass $X' \overset{\text{d}}{=} X$ und $Y' \overset{\text{d}}{=} Y$ gilt. Die nützliche Eigenschaft, die wir verwenden werden, ist $\Pr[X' = Y'] = \Pr[P = Q] = A_{\text{lila}}$. Daraus folgt nämlich wie gewünscht:

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

(ii) Sei $S = \{i \in \mathbb{N} \mid \Pr[X = i] > \Pr[Y = i]\}$. Sei nun (X', Y') irgendein Coupling von X und Y. Dann gilt:

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

Aufgabe 3 – Eigenschaften der Poissonverteilung

Sei $X \sim \text{Pois}(\lambda)$. Zeige:

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

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

(iii) Für $Y \sim \text{Pois}(\rho)$ unabhängig von X gilt $X + Y \sim \text{Pois}(\lambda + \rho)$.

(iv) Für $X' \sim \text{Bin}(X, p)$ gilt $X' \sim \text{Pois}(\lambda p)$.
Beachte: Hier wird also ein zweistufiges Zufallsexperiment durchgeführt. Das Ergebnis X des ersten ist ein Parameter des zweiten.

Lösung 3

Im folgenden verwenden wir die Definition der e-Funktion ständig, d.h. $e^t = \sum_{i=0}^{\infty} \frac{t^i}{i!}$.

(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) Wir bestimmen zunächst das zweite *un*zentrierte Moment:

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

Wir wissen zudem $\mathbb{E}[X]^2 = \lambda^2$. Es folgt

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

(iii) Sei $k \in \mathbb{N}$. Wir betrachten alle $k+1$ Möglichkeiten wie $X + Y$ zur Summe k führen kann und verwenden dann den binomischen Lehrsatz.

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

(iv) Sei $k \in \mathbb{N}$. Damit am Ende k herauskommt muss $X \geq k$ gegolten haben. Wir betrachten

alle Möglichkeiten.

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

Aufgabe 4 – Poissonisierte Bloom-Filter

Wir betrachten ein Poisson-Modell von Bloom-Filtern, gehen also davon aus, dass jede Position im Array unabhängig von andere Positionen $\mathrm{Pois}(\alpha k)$-verteilt oft als Hashwert vorkommt.

(i) Wir wählen wieder $\alpha k = \ln 2$. Wie lässt sich zeigen, dass der Anteil $\frac{Z}{m}$ der Nuller mit hoher Wahrscheinlichkeit nahe an $\frac{1}{2}$ liegt?

(ii) Wie ließe sich das Ergebnis in ein nicht-Poissonisiertes Modell übertragen?

Lösung 4

(i) Wenn $X \sim \mathrm{Pois}(\ln 2)$ so gilt $\Pr[X = 0] = e^{-\ln 2} = \frac{1}{2}$. Da jede Position nun unabhängig von allen anderen leer bzw. nicht leer ist, gilt also $Z \sim \mathrm{Bin}(m, \frac{1}{2})$. Es folgt $\mathbb{E}[\frac{Z}{m}] = \frac{1}{2}$ und auf Z sind nun direkt Chernoff Schranken anwendbar.

(ii) Die Anzahl $m - Z$ ist eine monotone Funktion im Sinne des Poissonisierungs-Theorems der Vorlesung. Entsprechend kann man das exakte "nk Bälle in m Behälter" Modell zwischen zwei Poissonisierten Modellen einsperren wie besprochen.

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 10 – Approximation Algorithms

Randomisierte Algorithmik

Aufgabe 1 – Die Jensensche Ungleichung

Sei $D \subseteq \mathbb{R}$ ein zusammenhängender Definitionsbereich und $f : D \to \mathbb{R}$ eine Funktion. Die Funktion f heißt konvex, wenn sie „linksgekrümmt“ ist und konkav wenn sie „rechtsgekrümmt“ ist.[1] Eine Funktion ist konvex genau dann wenn ihre Negation konkav ist. Für eine formale Definition siehe: Wikipedia

(a) Entscheide (ohne Beweis) für folgende Funktionen, ob diese auf ihrem jeweiligen Definitionsbereich konvex ist, konkav ist, beides ist oder weder noch.

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

(b) Sei f eine konvexe Funktion mit Definitionsbereich D. Argumentiere geometrisch, dass es für jedes $x_0 \in D$ eine lineare Funktion g gibt, sodass gilt:

(i) $f(x) \geq g(x)$ für alle $x \in D$

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

(c) Schlussfolgere, dass für jede konvexe Funktion f und für jede Zufallsvariable X mit Werten im Definitionsbereich D von f gilt:

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

Hinweis: Betrachte $x_0 = \mathbb{E}[X]$ und das zugehörige g aus der vorherigen Teilaufgabe.

(d) Zeige, dass analog für jede konkave Funktion f mit Definitionsbereich D und für jede Zufallsvariable X mit Werten in D gilt:

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

Die Ungleichung aus (c) sowie Varianten wie in (d) nennt man Jensensche Ungleichung.

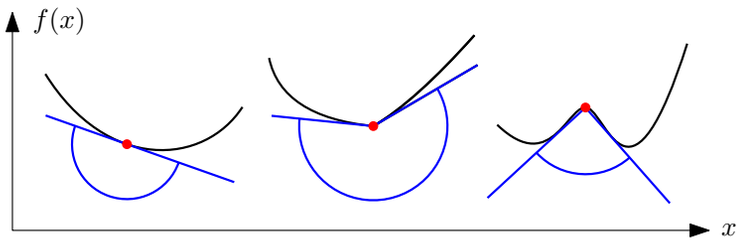
[1] Das „linksgekrümmt“ in Anführungszeichen lässt neben Linkskrümmungen (einer zweimal stetig differenzierbaren Funktion) auch Linksknicke und geradlinige Verläufe zu.

Lösung 1

(a) Die Funktionen sind alle zweimal stetig differenzierbar. Wenn die zweite Ableitung überall nicht-negativ ist, dann ist die Funktion konvex, wenn sie überall nicht-positiv ist, dann ist die Funktion konkav.

- f_1 ist konvex und konkav
- f_2 ist konvex
- f_3 ist weder konvex noch konkav
- f_4 ist konkav
- f_5 ist weder konvex noch konkav

(b) Weil f linksgekrümmt ist, kann man am Funktionsgraphen von f an Stelle $(x_0, f(x_0))$ eine Tangente g anlegen, sodass f vollständig oberhalb von g verläuft.



Dass das geht sieht man folgendemaßen (illustriert im Bild): Man betrachtet unterhalb des Punktes $(x_0, f(x_0))$ (rot) auf dem Funktionsgraphen von f (schwarz) den Winkelbereich derjenigen Richtungen, die niemals über den den Funktionsgraphen hinausgehen (blau). Ist dieser Bereich kleiner als 180° (rechts im Bild), dann hat man einen Widerspruch zur Konvexität von f. Ist dieser Bereich größer 180° (Mitte) oder gleich 180° (links), dann gibt es mindestens eine Gerade durch $(x_0, f(x_0))$, die vermeidet über den Funktionsgraphen von f hinauszugehen.

(c) Sei $g(x)$ die Funktion aus (b) für $x_0 = \mathbb{E}[X]$. Dann gilt $f(x) \geq g(x)$ für alle $x \in D$ und $f(\mathbb{E}[X]) = g(\mathbb{E}[X])$. Weil g eine Gerade ist gibt es $a, b \in \mathbb{R}$ sodass $g(x) = ax + b$. Es folgt

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

(d) Weil $-f$ konvex ist folgt direkt aus (c):

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

Aufgabe 2 – Analyse von Lossy Counting

Erinnerung: Lossy Counting ist ein einfacher Streaming Algorithmus, der die Länge m eines Streams approximativ zählt. Darin kommt ein Parameter $p \in (0, 1]$ vor. Der Algorithmus selbst sowie die Art und Weise, wie er verwendet wird, ist rechts nochmal zu sehen. Beweise:

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

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

(c) $\mathbb{E}[\text{space}] \leq \log(1 + mp) + 1$.
Hinweis: Mit space bezeichnen wir den maximalen Speicherverbrauch, der für den Zustand Z von LossyCounting benötigt wird. Eine Zahl $i \in \mathbb{N}$ lässt sich mit $\lceil \log_2(i + 1) \rceil$ Bits kodieren. Verwende die Jensensche Ungleichung aus Aufgabe 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

Verwendung:

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

Lösung 2

(a) Seien $X_1, \ldots, X_m \sim \text{Ber}(p)$ unabhängige Zufallsvariablen, wobei X_i angibt, ob das $i$te Element des Streams zu einer Erhöhung des Zählers Z führt. Dann ist $X := \sum_{i=1}^{m} X_i$ der Wert von Z nach dem letzten Update. Die Schätzung des Algorithmus für m ist somit result $= X/p$. Daher gilt:

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

(b) Mit der genannten Chernoff Schranke ergibt sich

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

Die Behauptung ergibt sich durch Betrachten der Gegenwahrscheinlichkeit.

(c) Da Z monoton wächst ist der Speicherbedarf für Z ganz am Ende am größten, nämlich $\lceil \log_2(1 + X) \rceil$. Weil $f(x) = \log(1 + x)$ konkav auf $[0, \infty)$ ist, folgt mit der Jensenschen Ungleichung

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

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 11
Game Theory & Yao's Principle
Randomisierte Algorithmik

Aufgabe 1 – Untere Schranken für randomisiertes Sortieren

Sei $n \in \mathbb{N}$ und **Inputs** die Menge aller Permutation von $\{1, \dots, n\}$. Sei **Algos** die Menge aller vergleichsbasierten *deterministischen* Sortieralgorithmen.

(i) Nenne ein $A \in$ **Algos** (ohne Beweis) mit $\max_{I \in \textbf{Inputs}} C(A, I) = O(n \log n)$.

(ii) Fingerübung: Für $n = 3$ gibt es einen randomisierten Algorithmus $\mathcal{A}$ der jeden deterministischen Algorithmus in folgendem Sinne schlägt:

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

Unser Plan im Folgenden ist zu zeigen, dass dieser Vorteil in O-Notation verschwindet.

(iii) Zeige, dass es keine „schwierige Eingabe" gibt, dass nämlich gilt:

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

Um gleich die bestmöglichen Kosten für eine Eingabe*verteilung* in den Blick nehmen, benötigen wir etwas Vorbeitung:

(iv) Zeige, dass in einem Binärbaum mit k Blättern die durchschnittliche Tiefe eines Blattes mindestens $\lfloor \log_2(k) \rfloor$ beträgt.
Hinweis: Zeige dafür, dass die durchschnittliche Blatttiefe bei balancierten Bäumen minimal ist, genauergesagt, dass sich jeder Baum, in dem sich die Blatttiefen um mindestens 2 unterscheiden, so umbauen lässt, dass die durchschnittliche Blatttiefe sinkt, die Anzahl der Blätter aber gleichgeblieben ist.

(v) Folgere nun mithilfe von Yaos Prinzip, dass gilt:

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

Lösung 1

(i) Mergesort.

(ii) Wir zeigen zunächst die linke Gleichung. Sei $A \in \textbf{Algos}$ beliebig. Bei Eingabe $I = (x, y, z)$ können wir aus Symmetriegründen (zwischen den Elementen) annehmen, dass A zunächst x und y vergleicht und $x < y$ herausfindet. Aus Symmetriegründen (zwischen $<$ und $>$) können wir annehmen, dass er als nächstes x und z vergleicht. Nun könnte es sein, dass $x < y$ und $x < z$ gilt, dann ist die Reihenfolge von y und z noch ungeklärt und ein weiterer Vergleich ist nötig. Also gibt es einen Eingabe I sodass $C(A, I) = 3$. Damit gilt $\max_{I \in \textbf{Inputs}} C(A, I) = 3$. Weil A beliebig war gilt:

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

Ein randomisierter Algorithmus kann "raten" welches Element das mittlere Element ist und die beiden anderen Elementen mit diesem Vergleichen. Wenn der Algorithmus richtig geraten hat ist nach zwei Vergleichen die vollständige Reihenfolge geklärt. Anderfalls ist der dritte Vergleich nötig. Dieses $\mathcal{A}$ erfüllt $\mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \frac{1}{3} \cdot 2 + \frac{2}{3} \cdot 3 = 2 + \frac{2}{3}$ für jede Eingabe I wie gewünscht.

(iii) Die Reihenfolge von „min" und „max" erlaubt es uns, den Algorithmus auf die Eingabe I zuzuschneiden. Sei dafür π die Permutation[1], die I sortiert. Wir definieren A in Abhängigkeit von I so:

```
Algorithm A(I'):
    vertausche Elemente von I' gemäß π
    fail ← FALSE
    for i = 1 to n − 1 do // prüfe ob sortiert
        if I'[i] > I'[i + 1] then
            fail ← TRUE
            break
    if fail then
        MergeSort(I')
```

Wir müssen uns drei Dinge überlegen:

Der Algorithmus ist korrekt, also $A \in$ Algos. Das ist klar: Der Algorithmus ordnet seine Eingabe I' zunächst gemäß π um und überprüft, ob I' dadurch sortiert ist. Falls ja, tut er nichts mehr, sonst benutzt er MergeSort.

Der Algorithmus ist schnell für I. Wenn die Eingabe I' mit I übereinstimmt (der Eingabe, auf die A zugeschnitten ist), dann ist I' nach den Vertauschungen gemäß π sortiert. Dann sind nur die $n - 1$ Vergleiche der Schleife nötig, um dies zu verifizieren.

[1] Mit „Permutation" meint man manchmal Anordnungen einer Menge (wie am Anfang der Aufgabe), manchmal einen Operator, der eine gegebene Folge umordnet (so ist es jetzt gemeint).

Es geht nicht noch schneller. Angenommen es wurden weniger als $n-1$ Vergleiche angestellt. Wir betrachten den Graphen $G = ([n], E)$ der eine Kanten zwischen i und j enthält falls das $i$te Elemente der Eingabe mit dem $j$ten Element der Eingabe verglichen wurde. Wegen $|E| < n-1$ ist G unzusammenhängend. Damit ist die relative Reihenfolge dieser Zusammenhangskomponenten nicht geklärt. (Formal: Wenn eine Eingabe mit der Bauminformation konsistent ist, dann auch eine weitere Eingabe. Also "weiß" der Algorithmus noch nicht was die Eingabe war, was aber nötig ist um die Ausgabe zu konstruieren.)

(iv) Sei T ein Binärbaum mit k Blättern und minimaler durchschnittlicher Blatttiefe. Dann ist T ein voller Binärbaum, das heißt jeder Knoten hat 0 oder 2 Kinder (Grund: Innere Knoten mit nur einem Kind kann man herausbasteln und die durchschnittliche Blatttiefe reduzieren). Seien L und L' die größte bzw. kleinste Tiefe eines Blattes in T. Wir zeigen nun $L' \geq L-1$. Angenommen das ist nicht so, dann gilt $L' \leq L-2$. Sei ℓ_1 ein Blatt auf Tiefe L und ℓ_2 sein Geschwisterknoten (existiert weil T voll ist), der ebenfalls ein Blatt sein muss (nach Wahl von L als maximale Tiefe). Sei ℓ' ein Blatt auf Tiefe L'. Wir hängen nun ℓ_1 und ℓ_2 um und machen sie zu Kindern von ℓ'. Dadurch entsteht wieder ein Binärbaum. Die Summe der Blatttiefen verändert sich aus folgenden Gründen:

- Die Blätter ℓ_1 und ℓ_2 haben nun Tiefe $L'+1$ anstatt L.
- Der Knoten p, der Vater von ℓ_1 und ℓ_2 war, ist nun ein Blatt der Tiefe $L-1$.
- Der Knoten ℓ' auf Tiefe L' ist jetzt kein Blatt mehr.

Die Summe der Blatttiefen verändert sich damit um

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

ist also kleiner geworden. Damit ist die durchschnittliche Blatttiefe kleiner geworden, was ein Widerspruch zur Wahl von T ist.

Also gilt $L' \geq L-1$, d.h. die Tiefen der Blätter unterscheiden sich um höchstens 1. *Wäre* die durchschnittliche Blatttiefe von T weniger als $\lfloor \log_2(k) \rfloor$, dann hätte mindestens ein Blatt eine Tiefe von weniger als $\lfloor \log_2(k) \rfloor$ und alle Blätter hätten Tiefe höchstens $\lfloor \log_2(k) \rfloor$. Damit gibt es weniger als $2^{\lfloor \log_2(k) \rfloor} \leq k$ Knoten – ein weiterer Widerspruch.

Also hat T durchschnittliche Blatttiefe mindestens $\lfloor \log_2(k) \rfloor$. Weil T nach Wahl minimale durchschnittliche Blatttiefe hat, gilt dies auch für alle anderen Binärbäume.

(v) Betrachten wir den Entscheidungsbaum T_A der ein beliebiges $A \in$ **Algos** beschreibt. Ohne Einschränkung schauen wir nur solche Algorithmen an, die einen Vergleich „$x < y$" nur dann anstellen, wenn sowohl „true" als auch „false" als Ergebnis noch möglich sind, also jeder Berechnungspfad auch von mindestens einer Eingabe beschritten wird. Das bedeutet, dass T_A ein voller Binärbaum ist. Für jede der $n!$ möglichen Eingaben ist eine andere Umordnung der Elemente nötig, also muss jede Eingabe zu einem anderen Blatt in A führen. Also hat A exakt $n!$ Blätter. Nach (iii) ist die durchschnittliche Blatttiefe von A mindestens $\lfloor \log_2(n!) \rfloor$. Wenn wir nun die Gleichverteilung $\mathcal{I}$ auf den Eingaben anschauen, dann ist die erwarte Anzahl von Vergleichen, die A für $I \sim \mathcal{I}$ anstellt exakt

die durchschnittliche Blatttiefe von T_A, also ebenfalls mindestens $\lfloor \log_2(n!) \rfloor$. Aus $n! \geq (n/2)^{n/2}$ folgt $\lfloor \log_2(n!) \rfloor \geq \lfloor (n/2) \log_2(n/2) \rfloor = \Omega(n \log n)$. Damit ist nach dem Satz von Yao

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

Aufgabe 2 – Yaos Prinzip ohne Schnick-Schnack(-Schnuck)

Beweise Yaos Prinzip ohne auf spieltheoretische Sätze zurückzugreifen (kein Satz von Nash, Loomis, etc). Beweise also, dass im Setting der Vorlesung für eine beliebige Verteilung $\mathcal{A}_0$ auf **Algos** und eine beliebige Verteilung $\mathcal{I}_0$ auf **Inputs** gilt:

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

Hinweis: Der spieltheoretische Vorbau der Vorlesung ist hier nicht erforderlich, weil wir nicht zeigen möchten, dass „=“ möglich ist.

Lösung 2

Es gilt:

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

In Worten: In der Mitte werden sowohl A als auch I *zufällig* gewählt. Links wird I nicht zufällig, sondern mit dem Ziel einer Maximierung gewählt, weshalb das Ergebnis größer wird. Rechts wird hingegen A nicht zufällig sondern mit dem Ziel einer Minimierung gewählt, weshalb das Ergebnis kleiner wird. Das löst bereits die Aufgabe.

Bemerkung. Formal könnte man den ersten Schritt (und den zweiten analog) noch kleinteiliger aufschreiben, indem man folgende zwei abstrakte Einsichten verwendet:

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

wobei f und g Funktionen sind, $\mathcal{X}$ eine Verteilung auf einer Menge X und $\mathcal{Y}$ eine Verteilung auf einer Menge Y.

Die erste Gleichung erlaubt ein Maximum durch einen Erwartungswert abzuschätzen, die zweite Gleichung erlaubt ein zweistufiges Zufallsexperiment und einen „erwarteten Erwartungswert“ in ein einstufiges Zufallsexperiment zu überführen. Nutzt man diese Gleichungen für $X = \textbf{Inputs}$, $Y = \textbf{Algos}$, $\mathcal{X} = \mathcal{I}_0$, $\mathcal{Y} = \mathcal{A}_0$, $f(I) = \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)]$ und $g(A, I) = C(A, I)$ so ergibt sich die erste Ungleichung von oben.

Aufgabe 3 – Empfehlung: Simulating the Evolution of Teamwork

Der Youtube-Kanal *Primer* beschäftigt sich mit evolutionärer Spieletheorie. In folgendem Video geht es unter anderem darum, alle möglichen 2-Spieler-Spiele mit zwei reinen Strategien danach zu klassifizieren, wieviele und welche Arten von Nash-Equilibria für diese existieren. Das Video ist unterhaltsam und lädt zum mitdenken ein, ist aber nur bedingt vorlesungsrelevant.

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

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 12 – Probabilistic Method

Randomisierte Algorithmik

Aufgabe 1 – Ein Kinderspiel

Alice und Bob spielen ein asymmetrisches Spiel auf einer Reihe von Feldern $0, 1, 2, \ldots, n$. Zu Beginn liegen k Tokens auf Feld 0. Jede Runde läuft wie folgt ab.

1. Alice wählt zwei disjunkte Menge T_1 und T_2 von Tokens.
2. Bob wählt daraufhin $i \in \{1, 2\}$.
3. Die Tokens aus T_i werden entfernt.
4. Die Tokens aus T_{2-i} bewegen sich jeweils ein Feld nach rechts.

Alice gewinnt, sobald ein Token Feld n erreicht. Sie verliert, sobald sie zwei leere Mengen wählt. Löse folgende Aufgaben.

(i) Gib eine Strategie für Alice, mit der sie für $k \geq 2^n$ gewinnt.

(ii) Benutze die probabilistische Methode, um zu zeigen, dass es eine Gewinnstrategie für Bob gibt wenn $k < 2^n$.

(iii) Bonus: Konstruiere eine Gewinnstrategie für Bob (ohne probabilistische Methode).

Lösung 1

(i) Ohne Einschränkung sei $k = 2^n$, denn Alice kann zusätzliche Tokens ignorieren. Alices Strategie ist es, die verbleibenden Tokens stets gleichmäßig auf T_1 und T_2 aufzuteilen. Dann wird stets die Hälfte der Tokens weiterwandern und die andere Hälfte wird entfernt. Eine einfache Induktion zeigt, dass nach $0 \leq i \leq n$ Runden genau 2^{n-i} Tokens auf Feld i liegen. Also hat Alice nach n Runden gewonnen.

(ii) Bob spielt uniform zufällig. Für Alice betrachten wir eine beliebige Strategie. Wir können das Spiel nun aus Sicht eines einzelnen Tokens t betrachten. Jedes mal, wenn t von Alice ausgewählt wird, wandert es mit Wahrscheinlichkeit $1/2$ einen Schritt nach rechts und wird mit Wahrscheinlichkeit $1/2$ entfernt. Wir können uns ohne Einschränkung vorstellen, dass Alice nie verliert (d.h. zwei leere Menge wählt) solange noch Tokens existieren und weiterspielt selbst wenn sie bereits gewonnen hat. Dann ist die Wahrscheinlichkeit, dass Token t jemals Position $i \in \mathbb{N}$ erreicht exakt 2^{-i}. Die Wahrscheinlichkeit, dass t jemals Position n erreicht ist damit 2^{-n}. Die erwartete Anzahl von Tokens die Position n

erreichen ist damit $2^{-n} \cdot k$, was nach Vorraussetzung < 1 ist. Damit ist es möglich, dass kein Token Position n erreicht. Damit ist die Wahrscheinlichkeit, dass Bob gewinnt positiv. Also ist Alices Strategie keine Gewinnstrategie. Weil Alices Strategie beliebig war, gibt es keine Gewinnstrategie für Alice. Also gibt es eine Gewinnstrategie für Bob.

(iii) Wir stellen uns vor, dass ein Token, das auf Feld i liegt, einen Wert von 2^i hat, das heißt der Wert eines Tokens verdoppelt sich wenn es ein Feld nach rechts wandert. Wenn Alice nun zwei Mengen T_1 und T_2 wählt, von Wert w_1 und w_2, dann sollte Bob stets die Menge der Tokens von größerem Wert entfernen. Ohne Einschränkung sei das T_1. Er lässt damit zu, dass sich der Wert der Tokens in T_2 verdoppelt. Weil für $w_1 \geq w_2$ aber $2w_2 \leq w_1 + w_2$ gilt kann der Gesamtwert nicht gewachsen sein. Weil am Anfang ein Wert von k vorhanden ist und ein Sieg von Alice einen Wert von mindestens 2^n vorraussetzt, kann Alice nicht gewinnen falls $k < 2^n$ gilt.

Aufgabe 2 – Größere[1] unabhängige Mengen

Sei $G = (V, E)$ ein Graph mit n Knoten und m Kanten. Zeige mithilfe der probabilistischen Methode, dass G eine unabhängige Menge der Größe $\sum_{v \in V} \frac{1}{\deg(v)+1}$ besitzt.

Hinweis: Zufällige Permutation der Knoten.

Lösung 2

Wir permutieren die Knoten zufällig. Sei anschließend

$$I = \{v \in V \mid v \text{ kommt in der Permutation vor all seinen Nachbarn}\}.$$

Es dürfte klar sein, dass I eine unabhängige Menge ist. Es ist ebenfalls klar, dass v mit Wahrscheinlichkeit $\frac{1}{\deg(v)+1}$ in I aufgenommen wird, denn dafür muss v in der zufälligen Permutation unter $\deg(v) + 1$ Knoten der Erste sein. Damit gilt

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

Nach dem Expectation Argument existiert also insbesondere eine unabhängige Menge der geforderten Größe.

[1] **Bemerkung:** Sei $d = \frac{2m}{n}$ der Durchschnittsgrad der Knoten. In der Vorlesung haben wir eine unabhängige Menge der Größe $\frac{n}{2d}$ konstruiert. Für die Größe U der unabhängigen Menge, die durch diese Aufgabe garantiert ist, gilt mithilfe einer Ungleichung über arithmetische und harmonische Mittel:

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

Das ist größer als $\frac{n}{2d}$ für $d > 1$.[2]

[2] "*Aber was ist wenn $d < 1$ gilt?*" Dann ist der Satz der Vorlesung gar nicht anwendbar.

Aufgabe 3 – Reprise: Unabhängige Regenbogenmengen

Sei $G = (V, E)$ ein Graph mit $|V| = kc$ Knoten, die mit c Farben gefärbt sind, wobei jede Farbe genau k mal vertreten ist. Der Maximalgrad sei Δ. Zeige: Falls $k \geq 8\Delta$ so gibt es eine unabhängige Regenbogenmenge.

Lösung 3

Bei dieser Aufgabe handelt es sich um eine einfache Verallgemeinerung der Perlenketten-Analyse der Vorlesung.

Wir wählen die Regenbogenmenge R wieder, indem wir aus jeder Farbklasse einen Knoten uniform zufällig ziehen. Ziel ist zu zeigen, dass R mit positiver Wahrscheinlichkeit eine unabhängige Menge ist.

Dafür definieren wir ein schlechtes Ereignisse $B_{\{u,v\}}$ für jede Kante $\{u, v\}$ des Graphen, das besagt, dass sowohl u als auch v in R gelandet sind. Es gilt wieder $\Pr[B_{\{u,v\}}] \leq \frac{1}{k^2} =: p$.

Ein anderes Ereignis $B_{\{u',v'\}}$ kann mit $B_{\{u,v\}}$ nur dann zu tun haben, wenn u' oder v' eine Farbe hat, die auch u oder v hat. Es gibt also höchstens $d = 2k\Delta - 2$ solche Ereignisse (2 relevante Farben, jeweils k relevante Knoten, jeweils Δ inzidente Kanten, wobei $\{u, v\}$ selbst von u und v aus jeweils nicht mitgezählt wird).

Damit gilt $4pd = 4\frac{1}{k^2}(2k\Delta - 2) < \frac{8\Delta}{k} \leq 1$. Damit können wir Lovász Local Lemma anwenden.

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 13 – Random Graphs

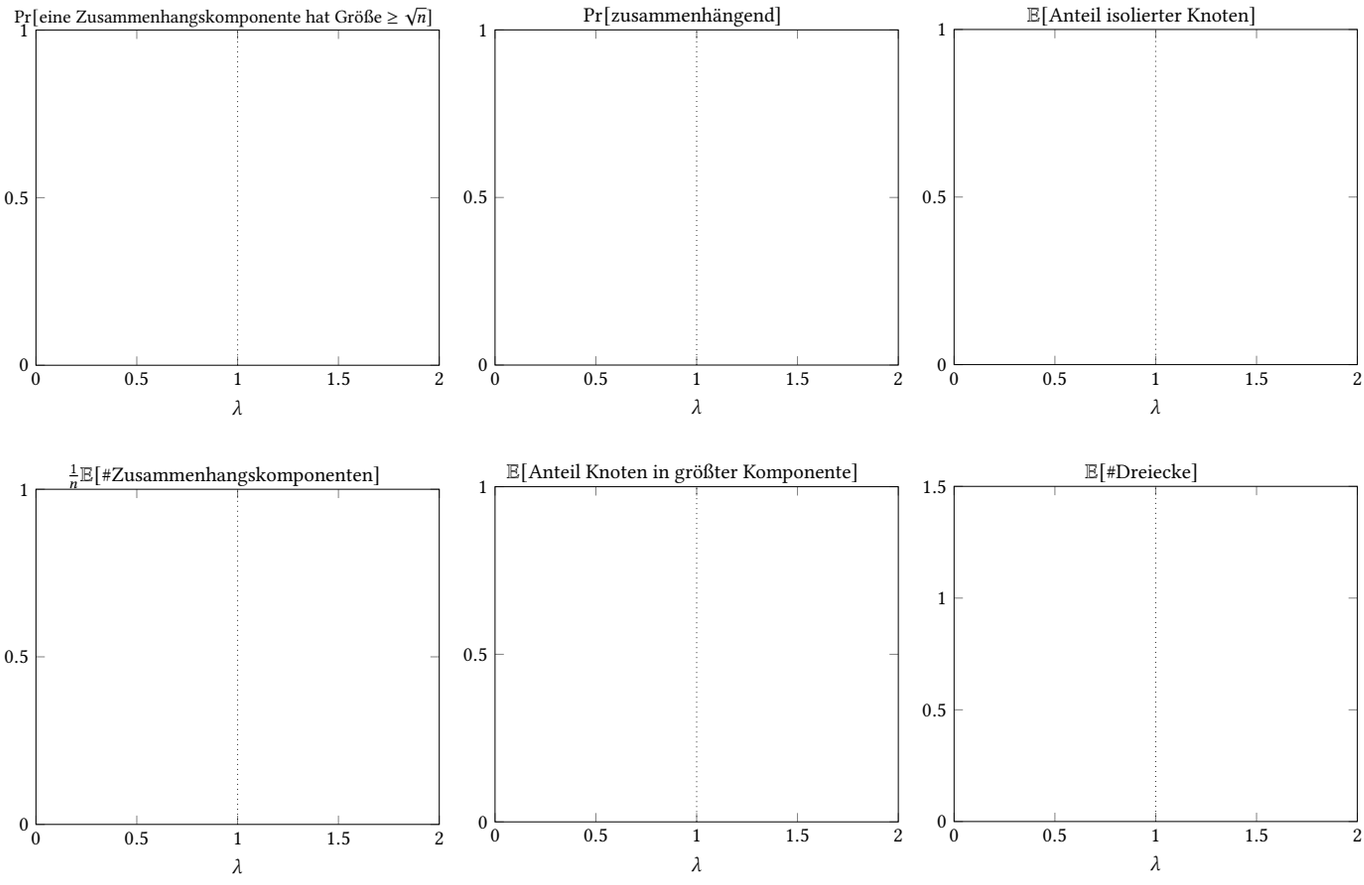
Randomisierte Algorithmik

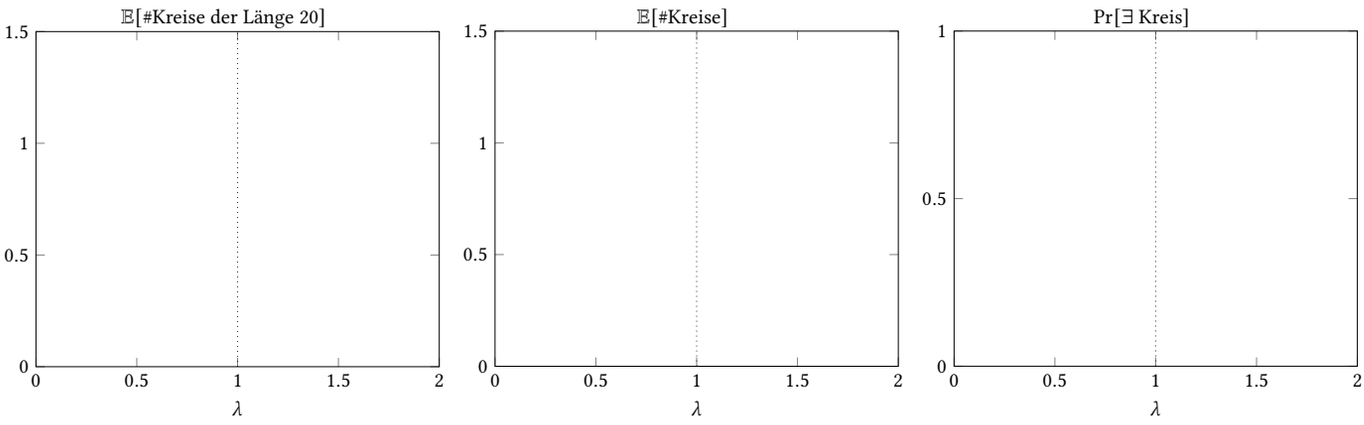
Aufgabe 1 – Intuition für Erdős-Renyi Graphen

Wir betrachten den Erdős-Renyi Graphen $G(n, \lambda n/2)$ oder den Gilbert Graph $G(n, \lambda/n)$ (beide führen zum selben Ergebnis). Der erwartete Knotengrad ist also $\lambda \pm O(1/n)$.

(i) Skizziere den Verlauf der folgender Wahrscheinlichkeiten und Erwartungswerte für $n \to \infty$ in Abhängigkeit von $\lambda \in [0, 2]$.

Hinweis: Es geht hier nicht um numerische Exaktheit sondern um den qualitativen Verlauf. Wo nimmt die Kurve den Wert 0,1 bzw. ∞ an? Passiert besonderes bei $\lambda = 1$?

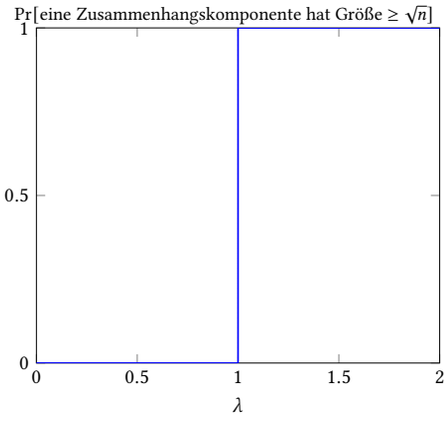




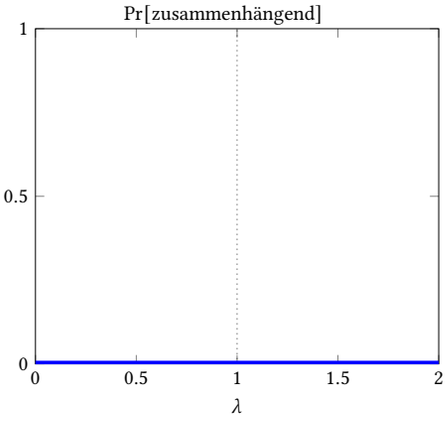
(ii) Sei $\lambda = \Theta(1)$. Rate: Was gilt mit hoher Wahrscheinlichkeit für (die Größenordnung) des minimalen und maximalen Knotengrads in Abhängigkeit von n?

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

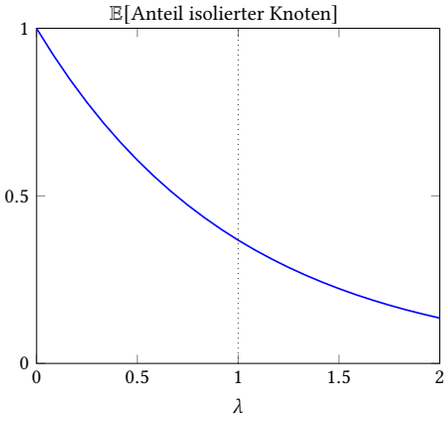
Lösung 1



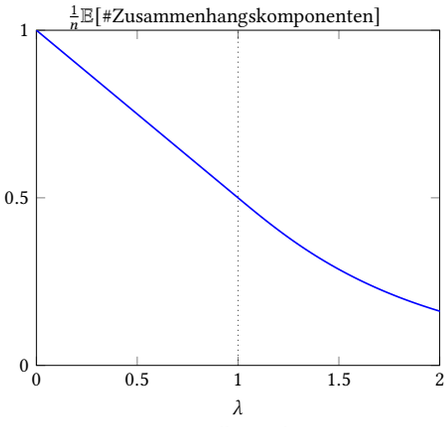
folgt aus „Sudden Emergence“ Ergebnis



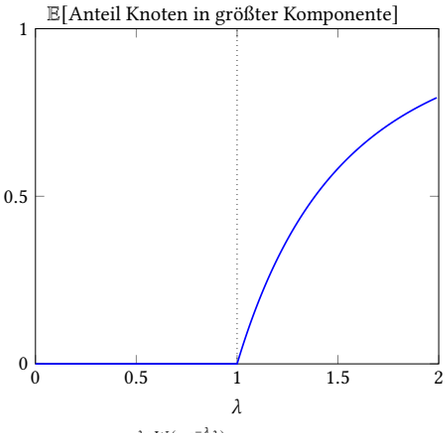
siehe nächste Frage



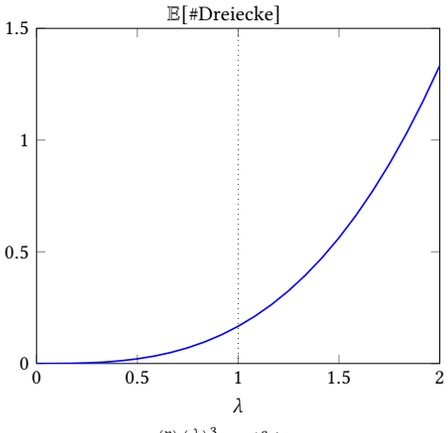
$\Pr_{X \sim \mathrm{Po}(\lambda)}[X = 0] = e^{-\lambda}$



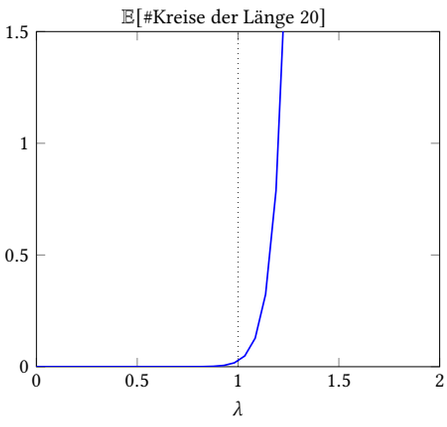
experimentelle Ergebnisse
für $n = 1000000$



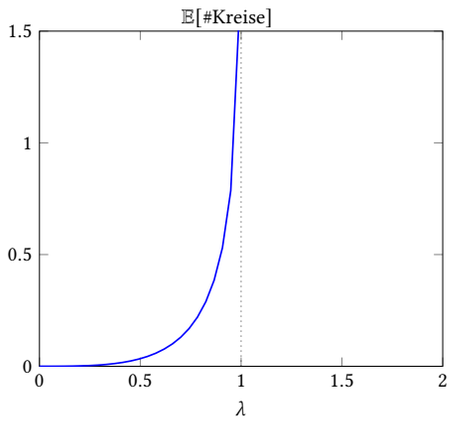
$\frac{\lambda + W(-e^{-\lambda}\lambda)}{\lambda}$ für $\lambda \geq 1$,
siehe Aufgabe 3



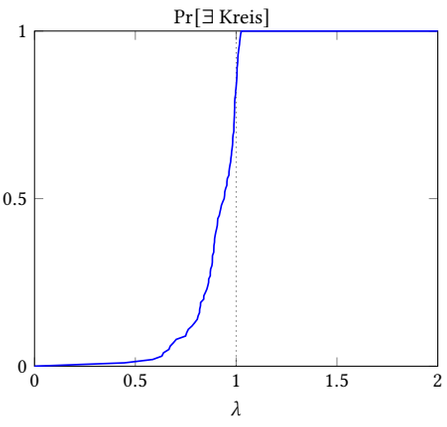
$\binom{n}{3}\left(\frac{\lambda}{n}\right)^3 \to \lambda^3/6$



$\frac{n(n-1)\cdot\ldots\cdot(n-19)}{2\cdot 20}\cdot\left(\frac{\lambda}{n}\right)^{20}\to\lambda^{20}/40$



$\to\sum_{k=3}^{\infty}\frac{\lambda^k}{2k}=\frac{1}{4}(-\lambda^2-2\lambda-2\ln(1-\lambda))$



experimentelle Ergebnisse
für $n = 1000000$

(ii) Der minimale Knotengrad ist mit hoher Wahrscheinlichkeit 0. In Aufgabenteil (a) haben wir ja sogar gesehen, dass es erwartet sogar $\Theta(n)$ isolierte Knoten gibt. Ein formaler Beweis könnte etwas trickreich sein.

Der maximale Knotengrad ist mit hoher Wahrscheinlichkeit $\Theta(\frac{\log n}{\log\log n})$. Darauf kommt man, wenn man berücksichtigt, dass die erwartete Anzahl von Knoten von Grad d etwa $\mathbb{E}[N_d] \approx n\cdot e^{-\lambda}\frac{\lambda^d}{d!}$ beträgt. Setzt man $\mathbb{E}[N_d] = 1$ so kriegt man $d = \Theta(\frac{\log n}{\log\log n})$. Die hauptsächliche ist hier welches d man braucht damit $d! \approx n$ gilt.

Aufgabe 2 – Knotengrade in Erdős-Renyi Graphen

Auf Folie 15 des Abschnitts über Balls-Into-Bins und Poissonisierung haben wir gezeigt, dass $\mathrm{Bin}(n, \frac{\lambda}{n})$ *für* $n\to\infty$ *gegen* $\mathrm{Pois}(\lambda)$ *konvergiert (bzw. die CDFs entsprechend konvergieren). Du darfst in folgender Aufgabe ohne Beweis verwenden:*

> **Lemma.** Sei $t_1, t_2, \ldots \in \mathbb{N}$ und $p_1, p_2, \ldots \in (0,1)$ sowie $\lambda \in \mathbb{R}_+$. Sei ferner $X_n \sim \mathrm{Bin}(t_n, p_n)$ für alle $n\in\mathbb{N}$ und $X\sim\mathrm{Pois}(\lambda)$. Falls $t_n\longrightarrow\infty$ und $t_n\cdot p_n\longrightarrow\lambda$ für $n\to\infty$ gilt dann folgt $X_n\xrightarrow{\mathrm{d}}X$ für $n\to\infty$.

Sei $\lambda > 0$ und $X\sim\mathrm{Pois}(\lambda)$. Wir betrachten Varianten von Erdős-Renyi Graphen aus der Vorlesung. Wir stellen den erwarteten Knotengrad auf ungefähr λ und wollen (mit obigem Lemma) zeigen, dass die Verteilung eines Knotens sich asymptotisch $\mathrm{Pois}(\lambda)$ nähert wenn $n\to\infty$ geht (während λ konstant bleibt).

(i) Sei X_n der Grad von Knoten 1 in $G(n,p)$ mit $p=\lambda/n$. Zeige $X_n\xrightarrow{\mathrm{d}}X$.

(ii) Sei X_n der Grad von Knoten 1 in $G^{\mathrm{UE}}(n,m)$ mit $m=\lfloor\lambda n/2\rfloor$. Zeige $X_n\xrightarrow{\mathrm{d}}X$.

(iii) Sei X_n der Grad von Knoten 1 in $G(n,m)$ mit $m=\lfloor\lambda n/2\rfloor$. Zeige $X_n\xrightarrow{\mathrm{d}}X$.

Hinweis: Der letzte Aufgabenteil ist mit Abstand der schwierigste. Es bietet sich an $G(n,m)$ mithilfe eines Couplings zwischen zwei Gilbert Graphen $G^- = (n,p^-)$ und $G^+=(n,p^+)$ „einzusperren“.

Lösung 2

(i) Weil jede der $n-1$ zu Knoten 1 inzidenten Kanten mit Wahrscheinlichkeit λ/n vorhanden ist gilt $X_n \sim \mathrm{Bin}(n-1, \lambda/n)$. Wir verwenden das Lemma mit $t_n = n-1$ und $p_n = \lambda/n$. Offenbar gilt $t_n \to \infty$ und $t_n p_n \to \lambda$. Also folgt $X_n \to X$ wie gewünscht.

(ii) Weil jeder der $2m$ Kantenendpunkte mit Wahrscheinlichkeit $1/n$ an Knoten 1 angeschlossen wird gilt $X_n = \mathrm{Bin}(2m, \frac{1}{n})$. Wir wenden das Lemma an mit $t_n = 2m = 2\lfloor \lambda n/2 \rfloor$ und $p_n = 1/n$. Offenbar gilt dann $t_n \to \infty$ und $t_n p_n \to \lambda$. Also folgt $X_n \to X$ wie gewünscht.

(iii) Sei $G = G(n, m)$. Wir betrachten außerdem die Gilbert Graphen $G^- = G(n, p^-)$ sowie $G^+ = G(n, p^+)$ wobei $p^- := \frac{\lambda}{n} - n^{-4/3}$ und $p^+ := \frac{\lambda}{n} + n^{-4/3}$. Weil sich p^- und p^+ nur geringfügig von $p = \lambda/n$ unterscheiden konvergiert der Grad X_n^- von Knoten 1 in G^- sowie der Grad X_n^+ von Knoten 1 in G^+ wie in Aufgabenteil (i), es gilt also $X_n^- \xrightarrow{\mathrm{d}} X$ sowie $X_n^+ \xrightarrow{\mathrm{d}} X$. Um den Grad X_n zwischen X_n^- und X_n^+ „einzusperren“, verwenden wir ein Coupling.

Sei dazu $\mathcal{E}$ die Menge der $\binom{n}{2}$ möglichen Kanten. Im gemeinsamen Wahrscheinlichkeitsraum gibt es unabhängig für jedes $e \in \mathcal{E}$ eine Zufallsvariable $Z_e \sim \mathcal{U}([0,1])$. Die Kantenmenge von E^-, E und E^+ von G^-, G und G^+ (bzw. formal von deren „Kopien“ $G^{-\prime}$, G' und $G^{+\prime}$ mit gleicher Verteilung) seien:

$$\begin{aligned} E^- &= \{e \in \mathcal{E} \mid Z_e \le p^-\} \\ E &= \{e \in \mathcal{E} \mid Z_e \text{ gehört zu den } m \text{ kleinsten Zahlen in } (Z_e)_{e\in\mathcal{E}}\} \\ E^+ &= \{e \in \mathcal{E} \mid Z_e \le p^+\} \end{aligned}$$

Es dürfte klar sein, dass sich so die richtigen Verteilungen ergeben, es sich also um ein gültiges Coupling handelt. Sei nun $m^- = |E^-|$ und $m^+ = |E^+|$. Es gilt:

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

Analog zeigt man, dass $\Pr[m^+ \le m] = \Theta(n^{-1/3})$. Wir betrachten nun das Ereignis $\mathrm{succ} = \{m^- \le m \le m^+\}$. Tritt succ ein, so gilt nach Konstruktion $E^- \subseteq E \subseteq E^+$ und damit $X_n^- \le X_n \le X_n^+$. Nach Rechnung von eben und Union Bound gilt $\Pr[\mathrm{succ}] = 1 - \Theta(n^{-1/3})$. Es folgt für $i \in \mathbb{N}_0$:

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

Sehr ähnlich folgt auch

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

Damit gilt $\Pr[X_n \le i] \longrightarrow \Pr[X \le i]$ für alle $i \in \mathbb{N}_0$ und es folgt $X_n \xrightarrow{\mathrm{d}} X$.

Aufgabe 3 – Aussterbewahrscheinlichkeit in Galton-Watson Bäumen

Sei GWT(λ) der Galton-Watson Baum, dem die Verteilung Pois(λ) zugrunde liegt.

(i) Sei n_i die Anzahl der Knoten in Ebene i von GWT(λ). Die Wurzelebene sei Ebene 0. Was ist $\mathbb{E}[n_i]$?

(ii) Für $i \in \mathbb{N}_0$ sei p_i die Wahrscheinlichkeit, dass GWT(λ) mindestens einen Knoten auf Ebene i hat. Drücke p_{i+1} in Abhängigkeit von p_i aus.
Hinweis: Verwende Aufgabe 3 (iv) von Blatt 9.

(iii) Bestimme für $\lambda \in \{0, 0.5, 1, 1.1, 1.5\}$ (oder für beliebige λ) jeweils eine Approximationen für die Wahrscheinlichkeit $s(\lambda)$, dass GWT(λ) unendlich ist. Ein Computeralgebrasystem könnte nützlich sein (z.B. Wolfram Alpha.

Zusatzüberlegung: Was ist die Erwartete Anzahl von Knoten auf Ebene i?

Lösung 3

(i) Jeder Knoten hat erwartet λ Kinder in der nächsten Ebene. Es ist recht intuitiv, dass daher $\mathbb{E}[n_i] = \lambda^i$ gilt. Formal kann man bedingte Erwartungswerte und Induktion verwenden. Kompakt aufgeschrieben ergibt sich:

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

(ii) Sei $\text{depth}(T) \in \mathbb{N}_0 \cup \{\infty\}$ die Tiefe eines Baumes T. Sei $X \sim \text{Pois}(\lambda)$ die Anzahl von Kindern der Wurzel von GWT(λ) und seien $T^{(1)}, T^{(2)}, \ldots, T^{(X)}$ die bei diesen Kindern startenden Unterbäume. Sei $Y = |\{i \in \{1, \ldots, X\} \mid \text{depth}(T^{(i)}) \geq i - 1\}|$ die Anzahl der Unterbäume mit Tiefe mindestens $i - 1$. Weil die Unterbäume die selbe Verteilung wie GWT(λ) haben, gilt $\Pr[\text{depth}(T^{(i)}) \geq i - 1] = p_{i-1}$ für alle $i \in \{1, \ldots, X\}$. Weil Unterbäume unabhängig voneinander sind, gilt ferner $Y \sim \text{Bin}(X, p_{i-1})$. Damit folgt $Y \sim \text{Pois}(\lambda p_{i-1})$ aus Aufgabe 3 (iv) von Blatt 9. Also gilt:

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

(iii) Es ist intuitiv, dass $s(\lambda) = \lim_{i \to \infty} p_i$ gilt, aber wir zeigen es trotzdem formal. Die Wahrscheinlichkeit, dass GWT(λ) *genau* i Ebenen hat ist $q_i = p_i - p_{i+1}$. Offenbar gilt dann $s(\lambda) + q_0 + q_1 + \ldots = 1$. Es folgt:

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

Nach (ii) gilt $p_0 = 1$ und $p_{i+1} = 1 - e^{-\lambda p_i}$ für alle $i \geq 0$. Es bietet sich an, die Funktion $f(x) = 1 - e^{-\lambda x}$ zu betrachten. Iteriertes Anwenden der Funktion startend bei $x = 1$

generiert die Folge $(p_i)_{i\in\mathbb{N}}$. Weil f monoton fallend ist und unser Startwert $f(p_0) < p_0$ erfüllt, folgt, dass die Folge monoton fallend ist. Damit ist $s(\lambda) = \lim_{i\to\infty} p_i$ der größten Fixpunkt von f. Für $\lambda \leq 1$ ist $f(x) = 1 - e^{-\lambda x} \leq 1 - e^{-x} \leq 1 - (1 - x) = x$ mit Gleichheit genau für $x = 0$. Für solche λ gilt also $s(\lambda) = 0$.

Für $\lambda > 1$ ist die größte Lösung von $x = 1 - e^{-\lambda x}$ verschieden von der trivialen Lösung $x = 0$. Ein Computeralgebrasystem kann diese als $s(\lambda) = \frac{\lambda + W(-\lambda e^{-\lambda})}{\lambda}$ bestimmen wobei W die sogenannte Labertsche W-Funktion ist. Numerisch kann man dann $s_{1.1} = 0.176134$ und $s_{1.5} = 0.582812$ berechnen. Ein Plot ist in Aufgabe 1 zu finden.

Karlsruher Institut für Technologie

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 14 – Cuckoo Hashing

Randomisierte Algorithmik

Aufgabe 1 – Cuckoo Hashing & Erdős-Renyi Graphen

Beachte: Für diese Aufgabe ist n eine Anzahl Kanten und m eine Ansatz Knoten.

Das „Sudden Emergence"-Resultat von Erdős und Renyi (Folie 19, Random Graphs Kapitel) gilt auch, wenn man $G(m, \lambda/m)$ durch $G^{\text{UE}}(m, \frac{\lambda m}{2})$ ersetzt. Beschreibe eine Variante von Cuckoo Hashing, der bei n Schlüsseln und m Tabellenplätzen der Graph $G^{\text{UE}}(m, n)$ zugrunde liegt.

(i) Wir wollen Schlüssel bis zu einer Beladung von $\frac{n}{m} = \alpha < \frac{1}{2}$ einfügen. Folgere aus dem „Sudden Emergence" Ergebnis, dass dies mit hoher Wahrscheinlichkeit möglich ist.

(ii) Es ergibt sich ein praktischer Nachteil bei der Implementierung von insert. Welcher?

Lösung 1

(i) Wir verwenden eine Tabelle der Größe m und zwei voll zufällige Hashfunktionen $h_1, h_2 : D \to [m]$. Jeder Schlüssel $x \in S$ entspricht einer Kante $\{h_1(x), h_2(x)\}$ im Graphen, dessen Knotenmenge die Tabellenpositionen sind. Für $n = |S|$ Schlüssel hat der Graph exakt die Verteilung von $G^{\text{UE}}(m, n)$ (auch Mehrfachkanten und Schleifen sind hier möglich). Falls die Beladung $\frac{n}{m} = \alpha < \frac{1}{2}$ gilt, so ist der durchschnittliche Knotengrad $\lambda = \frac{2n}{m} < 1$.

Aus dem „Sudden Emergence" Ergebnis folgt aus $\lambda < 1$, dass $G^{\text{UE}}(m, n)$ mit hoher Wahrscheinlichkeit nur aus Bäumen und Pseudobäumen besteht, also aus Komponenten mit höchstens einem Kreis. Für diese ist es stets möglich die Kanten kollisionsfrei zu richten und entsprechend die Schlüssel kollisionsfrei zu platzieren.

(ii) Beim Verdrängen eines Schlüssels ist zunächst nicht klar, ob dieser gemäß seiner ersten oder zweiten Hashfunktion platziert wurde. Man muss also im Zweifelsfall beide ausprobieren um die Ausweichposition zu finden. Dies bringt einen Branch oder doppelte Arbeit beim Hashen mit sich. In der Praxis ist daher die „bipartite" Variante wie in der Vorlesung gängig.

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 15 – Peeling

Randomisierte Algorithmik

Aufgabe 1 – Deterministisches Schälen in Linearzeit

(a) Der Algorithmus constructByPeeling ist nichtdeterministisch (in der Wahl von i in der While-Schleife). Zeige, dass dennoch für zwei mögliche Ausführungen von constructByPeeling stets die gleiche Menge von Schlüsseln unplatziert bleibt. (Insbesondere beeinflusst der Nichtdeterminismus nicht Erfolg/Misserfolg.)

(b) Verfeinere den Pseudocode so, dass ein deterministischer Algorithmus herauskommt, der erkennbar Laufzeit $O(n)$ hat. Nutze geeignete Hlifsdatenstrukturen deiner Wahl.

Lösung 1

(a) Sei $X = (x_1, \ldots, x_k)$ die Folge von Schlüsseln, die eine vollständige Ausführung von constructByPeeling platziert (in dieser Reihenfolge) bevor die While-Schleife verlassen wird. Sei $Y = (y_1, \ldots, y_\ell)$ eine andere mögliche Folge. Es genügt zu zeigen, dass jedes Element, das in X vorkommt, auch in Y vorkommt: Aus Symmetriegründen folgt dann, dass X und Y die selben Elemente (möglicherweise in anderer Reihenfolge) enthalten.

Wir machen einen Beweis durch Widerspruch und nehmen an, $j \in [k]$ ist der kleinste Index sodass x_j nicht in Y vorkommt. Von der X-Ausführung wird x_j an einem Index i_j platziert, der für keinen der Schlüssel aus $S \setminus \{x_1, \ldots, x_{j-1}\}$ in Frage kommt. Also kommt der Index i_j auch für keinen Schlüssel aus $S \setminus \{y_1, \ldots, y_\ell\}$ in Frage, denn $\{y_1, \ldots, y_\ell\} \supseteq \{x_1, \ldots, x_{j-1}\}$ nach Wahl von j. Nach Annahme ist x_j am Ende der Y-Ausführung noch in S. Dann wäre aber i_j ein Index der nur für x_j und keinen anderen verbleibenden Schlüssel in Frage kommt. Damit wäre ein weiterer Durchlauf der While-Schleife möglich. Damit beschreibt Y keine vollständige Ausführung von constructByPeeling. Widerspruch.

(b) Wir legen dem Algorithmus den bipartiten Cuckoo-Graphen wie in der Vorlesung definiert zugrunde. Wir benutzen eine Warteschlange Q. Invariante ist, dass Q alle Tabellenknoten von Grad 1 enthält (und eventuell weitere Tabellenknoten). Es spielt keine Rolle ob Q eine FIFO/LIFO oder andere Art von Warteschlange ist.

```
Algorithm constructByPeeling-Linear(S, h_1, h_2, h_3):
    T ← [⊥, ..., ⊥] // leere Tabelle
    G ← (S, [m], {(x, h_i(x)) | x ∈ S, i ∈ [3]}) // cuckoo graph
    Q ← ∅ // leere Warteschlange
    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 ← eindeutiger Nachbar von i
            T[i] ← x
            S ← S \ {x}
            for j ∈ {h_1(x), h_2(x), h_3(x)} do
                lösche die Kante (x, j) aus G
                if deg_G(j) = 1 then
                    Q.push(j)
    if S = ∅ then
        return T
    else
        return NOT-PEELABLE
```

Nach Annahme lässt sich G am Anfang in Zeit $O(m)$ konstruieren und jede weitere Einzeloperation lässt sich in $O(1)$ ausführen. Weil jeder Knoten nur einmal zu einem Grad 1 Knoten werden kann (weil wir nur Kanten löschen), wird für jedes $i \in [m]$ nur einmal Q.push(i) ausgeführt. Damit ist die Anzahl der While-Schleifendurchläufe auch höchstens m (weil da jeweils ein Element aus Q entfernt wird). Insgesamt ergibt sich $O(m)$.

Bemerkung. Anstatt eine Adjazenzlistendarstellung von G zu wählen, genügt im vorliegenden Fall auch eine sparsamere Darstellung. Für jede Tabellenposition speichern wir zwei Dinge: Ihren Grad (den sie in G hätte) sowie die *Summe* der mit ihr verbundenen Elemente aus S (dies nimmt an, dass das Universum D eine Gruppe ist, zum Beispiel $D = \mathbb{Z}_{2^{64}}$). Man kann an dieser Darstellung erkennen ob der Grad eines Tabellenknotens 1 ist und in diesem Fall den eindeutigen Nachbarn extrahieren. Wenn der Grad größer 1 ist kann man durch Subtrahieren einen gegebenen Nachbarn herauslöschen. Diese Datenstruktur kommt ohne Zeiger aus.

Aufgabe 2 – Peeling mit 2 Hashfunktionen

Angenommen wir verwenden den Schälalgorithmus constructByPeeling in einem Setting mit nur zwei Hashfunktionen h_1 und h_2. Wir gehen vereinfachend davon aus, dass $h_1(x) \neq h_2(x)$

für alle $x \in D$ gilt und unter dieser Einschränkung die Paare $(h_1(x), h_2(x))$ uniform zufällig und für verschiedene $x \in D$ unabhängig sind.[1] Zeige:

(a) Es werden alle Schlüssel platziert genau dann wenn folgender Graph kreisfrei ist:

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

Beachte: G ist als Multigraph zu verstehen.

(b) Sei $\alpha > 0$ eine Konstante und $n = \lfloor \alpha m \rfloor$. Zeige, dass es (für $m \to \infty$) mit Wahrscheinlichkeit $\Omega(1)$ einen Kreis gibt.
Hinweis: Es genügt nach Kreisen der Länge 2 (also Doppelkanten) zu fanden.
Siehe `https://en.wikipedia.org/wiki/Birthdayproblem#Arbitrarynumberofdays`.

Es gibt damit keinen Schälbarkeitsschwellwert, wie es ihn für $k \geq 3$ gibt, denn für kein $\alpha = \Theta(1)$ hat constructByPeeling mit hoher Wahrscheinlichkeit Erfolg.

Lösung 2

(a) Wir können constructByPeeling als Algorithmus auf G auffassen, der wiederholt Knoten von Grad 1 sucht. Ist v ein solcher Knoten, dessen inzidente Kante von einem Schlüssel x herkommt, dann wird x in v platziert und die Kante zu x wird gelöscht. Es werden also alle Schlüssel platziert, wenn das sukzessive Löschen von Kanten mit einem Endpunkt von Grad 1 mit dem leeren Graphen endet.

Starten wir mit einem Wald G, dann ist G in jedem Schritt ein Wald (löschen von Kanten aus einem Wald liefert wieder einen Wald). Solange der Wald nicht leer ist, also mindestens ein Baum mit mindestens einer Kante übrig ist, hat dieser ein Blatt von Grad 1 an dem weitergearbeitet werden kann. Ergo: Wenn G ein Wald ist, werden alle Schlüssel platziert.

Hat G dagegen mindestens einen Kreis, der von Schlüsseln $x_1, \ldots, x_k$ herkommt, dann kann keiner dieser Schlüssel der erste sein, der von constructByPeeling platziert wird, denn beide möglichen Positionen des Schlüsseln kommen auch für einen zyklisch benachbarten Schlüssel in Frage. Also werden nicht alle Schlüssel platziert.

(b) Wir können uns vorstellen, dass wir beim Bestimmen der Hashwerte aller Schlüssel n mal mit Zurücklegen aus einer Urne mit $\binom{m}{2}$ Bällen ziehen, die den möglichen Kanten

[1] Um das sicherzustellen können wir zum Beispiel $h_2(x) := (h_1(x) + h_{\text{diff}}(x)) \bmod m$ definieren für ein $h_{\text{diff}} : D \to [m-1]$.

entsprechen. Dass wir paarweise verschiedene Bälle ziehen hat Wahrscheinlichkeit:

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

Im ersten Schritt nutzen wir dass $1 + x \leq e^x$ für $x \in \mathbb{R}$ gilt. Später nutzen wir $1 + 2 + \cdots + n - 1 = n(n-1)/2$. Die Wahrscheinlichkeit, dass wir nicht verschiedene Bälle ziehen und entsprechend eine Doppelkante in G vorkommt, ist damit im Grenzwert mindestens $1 - e^{-\alpha^2}$ und damit $\Omega(1)$. Insbesondere gibt es mit Wahrscheinlichkeit $\Omega(1)$ einen Kreis.

Aufgabe 3 – Der Schälalgorithmus bleibt nicht erst spät stecken[2]

Für eine Schlüsselmenge $S \subseteq D$ der Größe $n = |S|$ und Hashfunktionen $h_1, h_2, h_3 \sim \mathcal{U}([m]^D)$ betrachten wir den Cuckoo Graphen wie in der Vorlesung:

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

Wir nehmen lediglich $\alpha = \frac{n}{m} \leq 1$ an. Sei $S' \subseteq S$ die Menge der Schlüssel, die vom Schälalgorithmus nicht entfernt werden können (es gilt $|S'| \in \{0, \ldots, n\}$). Wir wollen zeigen, dass $\Pr[|S'| \in \{1, \ldots, \delta m\}] = O(1/m)$ ist für eine Konstante $\delta > 0$ (die später gewählt wird).

Intution: Entweder gilt $S' = \emptyset$ oder $|S'|$ ist $\Omega(m)$; alles dazwischen ist unwahrscheinlich.

(a) Beobachte: $|S'| = 1$ ist nicht möglich.

(b) Zeige: $|N(S')| \leq \frac{3}{2}|S'|$. Hierbei ist $N(S')$ die Menge aller Nachbarn von S' in G.

(c) Zeige, dass es eine Konstante C gibt, sodass für $s \in \{2, \ldots, n\}$ folgendes gilt:

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

Hinweis: Einfach brutal Union-Bound über alle Möglichkeiten von X und Y verwenden. Nützlich ist außerdem die Abschätzung $\binom{n}{k} \leq (\frac{ne}{k})^k$ für Binomialkoeffizienten. Ignoriere die Gaussklammern, das machen alle so.

(d) Zeige (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) Wähle $\delta = 1/2C$ und zeige $\Pr[|S'| \in \{1, \ldots, \delta m\}] \leq \sum_{s=2}^{\delta m} p_s = O(1/m)$.

[2]Diese Aufgabe ist etwas aufwändig. Sie ist vor allem auf dem Blatt weil sonst der Beweis der Vorlesung unvollständig wäre.

Lösung 3

(a) Ein einzelner Schlüssel kann sich nicht selbst im Weg stehen.

(b) Je nachdem, ob man Doppelkanten zählt oder nicht, gehen von S' genau $3|S'|$ oder höchstens $3|S'|$ Kanten aus. Weil jeder Behälter in $N(S')$ von zwei verschiedenen Schlüsseln aus S' getroffen werden muss (sonst könnte man weiterschälen) hat jeder Knoten in $N(S')$ mindestens zwei Nachbarn in S'. Für die Anzahl e von Kanten zwischen S' und $N(S')$ gilt also $3|S'| \geq e \geq 2|N(S')|$. Umstellen liefert die Behauptung.

(c) Es gibt $\binom{n}{s}$ Möglichkeiten um X zu wählen und $\binom{m}{(3/2)s}$ Möglichkeiten um Y zu wählen. Für beliebiges $x \in X$ gilt $\Pr[N(\{x\}) \subseteq Y] = (|Y|/m)^3$, denn alle drei Hashwerte müssen (unabhängig voneinander) in der Menge Y landen. Wir erhalten somit folgendes (auf dem Weg fassen wir einige Konstanten durch ein $C = \Theta(1)$ zusammen):

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

(d) (i) Es gilt $p_2 = O(m^{-1})$, $p_3 = O(m^{-3/2})$, $p_4 = O(m^{-2})$, $p_4 = O(m^{-5/2})$. Passt also.

(ii) Jeder der Summanden in $\sum_{s=6}^{\sqrt{m}} \left(\frac{Cs}{m}\right)^{s/2}$ ist höchstens $\left(\frac{C\sqrt{m}}{m}\right)^{6/2} = O(m^{-3/2})$. Da es $O(m^{1/2})$ Summanden gibt, summieren diese sich zu höchstens $O(m^{-1})$.

(iii) Jeder der Summanden in $\sum_{s=\sqrt{m}}^{m/(2C)} \left(\frac{Cs}{m}\right)^{s/2}$ ist höchstens $\left(\frac{1}{2}\right)^{\sqrt{m}/2} = O(m^{-2})$ (*sehr* grob abgeschätzt). Da es $O(m)$ Summanden gibt, summieren diese sich zu höchstens $O(m^{-1})$.

(e) Wähle $\delta = \frac{1}{2C}$. Angenommen $|S'| = s$ für ein $s \in \{1, \ldots, \delta m\}$. Sei $X = S'$ und $Y' := N(S')$. Nach (b) gilt $|Y'| \leq \frac{3}{2}s$ also gibt es eine Menge $Y \supseteq Y'$ mit $|Y| = \lfloor \frac{3}{2}s \rfloor$ und $N(S') \subseteq Y$. Die Mengen X und Y erfüllen das Ereignis dessen Wahrscheinlichkeit wir mit p_s beschränkt haben. Zusammenfassend können wir also schlussfolgern:

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

Karlsruher Institut für Technologie

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 16 – Retrieval

Randomisierte Algorithmik

Aufgabe 1 – AMQ aus Retrieval

Sei $b \in \mathbb{N}$ und S eine Menge der Größe $n = |S|$. Verwende die Peeling-basierte Retrieval Datenstruktur der Vorlesung als Black-Box um einen statischen Filter (also eine Approximate-Membership-Query Datenstruktur) für S mit falsch-positiv Wahrscheinlichkeit $\varepsilon = 2^{-b}$ zu konstruieren.

Nenne Vor- und Nachteile der entstandenen Datenstruktur im Vergleich zu einem Bloom-Filter mit gleicher falsch-positiv Wahrscheinlichkeit. Du darfst annehmen, dass $b = O(\log n)$, sodass Bistrings der Länge b in Zeit $O(1)$ verarbeitet werden können.

Lösung 1

Sei $f \sim \mathcal{U}([2^b]^D)$ eine voll zufällige Hashfunktion. Gemäß der Simple Uniform Hashing Assumption können wir uns f merken ohne Speicher dafür aufzuwenden. Wir nennen $f(x)$ den *Fingerprint* von $x \in D$. Sei nun $f_S : S \to [2^b]$ die Einschränkung von f auf S („dieselbe Funktion“ aber kleinerer Definitionsbereich).

Wir konstruieren eine Retrieval Datenstruktur R für f_S. Wir fassen R als eine Filterdatenstruktur auf, die bei Anfrage $x \in D$ genau dann YES zurückgibt, wenn $\text{eval}(R, x) = f(x)$ gilt. Nach Konstruktion gibt es keine falsch-negativen Antworten. Nehmen wir nun an, dass $x \in D \setminus S$ gilt. Der Fingerprint $f(x)$ ist uniform zufällig in $[2^b]$ verteilt und unabhängig von allem was in der Konstruktion von R eine Rolle gespielt hat. Was auch immer $\text{eval}(R, x)$ ist, die Wahrscheinlichkeit, dass $f(x)$ damit übereinstimmt, und somit x ein falsch-positives Element ist, ist somit $2^{-b} = \varepsilon$ wie gewünscht.

Der Vergleich zu einem Bloom-Filter mit $\varepsilon = 2^{-b}$ fällt wie folgt aus:

- Geringerer Speicherbedarf: Rund $1.23kn$ statt rund $1.44kn$.
- Schnellere Konstruktion: $O(n)$ statt $O(nk)$.
- Schnellere Anfragen: $O(1)$ statt $O(k)$.
- Kein Unterstützung von `insert`.

Aufgabe 2 – Learned Data Structures

Sei S eine Menge von $n = |S|$ Namen mit eindeutig zuordenbaren Geschlecht $f : S \to \{\text{F}, \text{M}\}$. Eine findige Studentin bemerkt, dass die meisten $x \in S$ mit $f(x) = \text{F}$ auf einen Vokal enden und die meisten $x \in S$ mit $f(x) = \text{M}$ auf einen Konsonanten enden. Diese simple Regel funktioniert für alle außer δn der Namen, für ein kleines $\delta > 0$.

Konstruiere eine Datenstruktur mit erwartetem Speicherbedarf $O(\delta n \log(1/\delta))$, die für jedes $x \in S$ das korrekte Geschlecht $f(x)$ liefert.

Hinweis: Stöpsle dazu einen AMQ-Filter und eine Retrieval Datenstruktur geschickt zusammen.

Bemerkung: Unter *Learned Data Structures* versteht man eine Kombination aus klassischen Datenstrukturen und Machine Learning Techniken. Wie in dieser Aufgabe angedeutet, geht es darum, die Mustererkennungsfähigkeiten von Machine Learning Techniken mit den Verlässlichkeitsgarantien klassischer Datenstrukturen nutzbringend zu verheiraten.

Lösung 2

Sei $F \subseteq S$ die Menge der δn Namen, für die die Heuristik versagt. Sei B ein AMQ-Filter für F mit falsch-positiv Wahrscheinlichkeit δ. Sei nun S^+ die Menge derjenigen Namen aus S, für die der Filter eine positive Antwort gibt. Neben F enthält S^+ auch alle $x \in S$, die falsch-positive Elemente von B sind. Die erwartete Größe von S^+ beträgt höchstens $|F| + |S \setminus F| \cdot \delta = \delta n + (1-\delta)n \cdot \delta \leq 2\delta n$. Sei $f_{S^+} : S^+ \to \{\text{F}, \text{M}\}$ die Einschränkung von f auf S^+. Wir konstruieren eine Retrievaldatenstruktur R für f_{S^+}.

Die Intuition ist nun: Der AMQ-Filter identifiziert die Namen, für die die Heuristik fehlschlägt, sowie ein paar falsch-positive. Die Retrievaldatenstruktur hat dann zur Aufgabe echt-positive von falsch-positiven Elementen zu trennen. Formal können wir für unsere Heuristik-gestützte Retrievaldatenstruktur R_H definieren:

Algorithm eval(R_H, x)**:**
  if query$(B, x) = \text{NO}$ **then**
    return Heuristik(x)
  else
    return eval(R, x)

Der Gesamtspeicherbedarf beträgt $O(\delta n \log(1/\delta))$ Bits für B sowie erwartet höchstens $O(\delta n)$ Bits für R.

Aufgabe 3 – Retrieval mit Variabler Bitlänge

Gemäß Vorlesung können wir für jedes Universum D, jede Menge $S \subseteq D$ und jede Funktion $f : S \to \{0, 1\}$ eine Retrieval-Datenstruktur für f mit Speicherbedarf $1.23|S|$ konstruieren. Dies soll hier als Black-Box verwendet werden.

Konstruiere eine Retrieval Datenstruktur für den Fall in dem der Wertebereich der Funktion f Bitstrings variabler Länge enthält.

Genauergesagt, sei $C \subseteq \{0,1\}^*$ ein präfixfreier Code, D' ein Universum, $T \subseteq D'$ und $g : T \to C$ eine Funktion. Konstruiere eine Datenstruktur R mit Speicherbedarf $1.23 \cdot \sum_{x \in T} |g(x)|$ und einen zugehörigen Algorithmus eval sodass für jedes $x \in T$ gilt $\text{eval}(R, x) = g(x)$.
Hinweis: Führe für jedes $x \in T$ soviele Schlüssel ein, wie die Länge $|g(x)|$ von $g(x)$ beträgt.

Lösung 3

Die Idee ist es, für jedes $x \in T$ mit $g(x) = (b_1, \ldots, b_k)$ die Schlüssel $\{(x,1), (x,2), \ldots, (x,k)\} \subseteq D' \times \mathbb{N}$ einzuführen, wobei (x,i) der Wert b_i zugeordnet ist. Es ergibt sich somit eine Funktion $f : S \to \{0,1\}$ wobei $S \subseteq D' \times \mathbb{N}$ eine Menge von Paaren ist mit $|S| = \sum_{x \in T} |g(x)|$.

Wir können also gemäß Vorlesung für das Universum $D = D' \times \mathbb{N}$ sowie S und f wie beschrieben eine Retrieval Datenstruktur R mit Speicherbedarf $1.23|S|$ konstruieren. Um dieser Datenstruktur dann für $x \in T$ den Wert $g(x)$ zu entlocken, betrachten wir der Reihe nach $\text{eval}(R, (x,1)), \text{eval}(R, (x,2)), \text{eval}(R, (x,3)), \ldots$ bis die gesehene Folge einen Code aus C ergibt. Weil C präfixfrei ist, wissen wir, dass keine Fortsetzung der Folge in C liegt, also haben wir $g(x)$ vollständig bestimmt.

Bei einer Anfrage für $x \in D' \setminus T$ ist egal, welches Element von C zurückgegeben wird, wir müssen lediglich sicherstellen, dass eval nicht abstürzt oder in eine Endlosschleife gerät. Das ist leicht (und im Grunde fast automatisch der Fall).