Karlsruher Institut für Technologie

Maximilian Katzmann
Hans-Peter Lehmann
Stefan Walzer

Übungsblatt 15 – Aktivsession

Randomisierte Algorithmik – Wintersemester 2023/2024

Folgende Aufgaben werden in der Aktiv-Session am 13.2.2024 gemeinsam bearbeitet.
Eine Abgabe ist nicht erforderlich.

Aufgabe 1 – AMQ aus Retrieval

Sei $k \in \mathbb{N}$ und S eine Menge der Größe $n = |S|$. Verwende eine Retrieval Datenstruktur der Vorlesung als Black-Box um eine statische Approximate-Membership-Query Datenstruktur für S mit einer falsch-positiv Wahrscheinlichkeit von $\varepsilon = 2^{-k}$ und einem Speicherbedarf von $1.23nk$ Bits zu konstruieren. (Beachte: Ein Bloomfilter würde $1.44nk$ Bits benötigen.)
Hinweis: Verwende eine Hashfunktion $f \sim \mathcal{U}([2^k]^D)$. Gemäß der Simple Uniform Hashing Assumption kannst du dir f merken ohne Speicher dafür aufzuwenden. Wir nennen $f(x)$ den *Fingerprint* von $x \in D$.

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 Bloom 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.

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.

Aufgabe 4 – Minimal-Perfekte Hash Funktion mit Brute Force

Betrachte folgendes Algorithmenpaar zur Konstruktion und Auswertung von Minimal-Perfekten Hashfunktionen[1]. Dabei sei $S \subseteq D$ und $n = |S|$. Wir gehen gemäß der SUHA davon aus, dass $h_1, h_2, h_3, \ldots \sim \mathcal{U}([n]^D)$ unabhängige, voll zufällige Hashfunktionen sind, die selbst keinen Speicherplatz benötigen.[2]

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

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

(a) Argumentiere: Jeder Schleifendurchlauf in construct führt mit Wahrscheinlichkeit $\frac{n!}{n^n}$ zum Erfolg.

(b) Argumentiere: Der Rückgabewert seed von construct erfüllt $\mathbb{E}[\text{seed}] = \frac{n^n}{n!}$.

(c) Der Speicherbedarf beträgt space = $\lceil \log_2(\text{seed}) \rceil$ Bits. Zeige: $\mathbb{E}[\text{space}] \leq n \log_2(e) + 1$.
Hinweis: Verwende die Jensen-Ungleichung (Aufgabenblatt 10) sowie die Stirling Approximation der Fakultätsfunktion. (siehe en.wikipedia.org/wiki/Stirling's_approximation).

(d) Bewerte den erwarteten Platzbedarf und die erwartete Konstruktionszeit des vorgestellten Verfahrens auf einer Skala von großartig bis grottenschlecht.

Aufgabe 5 – Untere Platzschranken für MPHF

Wir wollen zeigen, dass eine minimal-perfekte Hashfunktion im Allgemeinen nicht mit weniger als $\log_2(e) \approx 1.44$ Bits pro Element auskommt.

Betrachte Vorlesungsfolie 9. Sei $\varepsilon = 0$, $n = m \in \mathbb{N}$, $d = |D|$. Sei $\mathcal{I} := \{S \subseteq D \mid |S| = n\}$ die Menge aller möglicher Eingaben der Größe n. Wir betrachten die Anzahl der Eingaben, für die eine gegebene Datenstruktur P gleichzeitig als perfekte Hashfunktion dienen kann. Sei hierzu:

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

(a) Zum Warmwerden: Angenommen $n = 3$, $D = \{\text{a}, \text{b}, \text{c}, \text{d}, \text{e}, \text{f}, \text{g}, \text{h}\}$ und P ist eine Datenstruktur auf der sich eval wie in folgender Tabelle dargestellt verhält. Begründe: P kann für $|\text{cov}(P)| = 12$ verschiedene $S \in \mathcal{I}$ als MPHF dienen.

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

[1] Beachte: Die „Datenstruktur“ besteht hier lediglich aus der Zahl seed.
[2] Alternativ kann man sich vorstellen, dass eine einzige Hashfunktion $h : \mathbb{N} \times D \to [n]$ voll zufällig ist, die neben dem eigentlichen Schlüssel eine natürliche Zahl als Seed akzeptiert.

(b) Seien nun n und d sowie P beliebig. Zeige $\mathrm{cov}(P) \leq (\frac{d}{n})^n$.
Hinweis: Verwende ohne Begründung, dass unter allen Quadern mit vorgebener Summe von Kantenlängen der Würfel das maximale Volumen besitzt. Im n-dimensionalen bedeutet das $\max_{\substack{0 \leq c_1, \ldots, c_n \leq d \\ c_1 + \cdots + c_n = d}} c_1 \cdot \ldots \cdot c_n = (d/n)^n$.

Sei nun $\mathcal{P} = \{\mathrm{construct}(S) \mid S \in \mathcal{I}\}$ die Menge aller verschiedener Datenstrukturen, die construct produziert.

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

(d) Zeige $\binom{d}{n} / (\frac{d}{n})^n \geq \frac{n^n}{n!} \cdot (1 - o(1))$ falls $d = \omega(n^2)$.
Hinweis: Überlege dir in einem Zwischenschritt, dass $(1 - \frac{n}{d})^n \geq 1 - \frac{n^2}{d} = 1 - o(1)$ gilt.

(e) Begründe: Werden die Datenstrukturen, die construct liefert, jeweils als Bitstring einer Länge ℓ kodiert, dann gilt $\ell \geq \lceil \log_2(|\mathcal{P}|) \rceil$.

(f) Zeige dass $\ell \geq \log_2(e) \cdot n - o(n)$ gilt falls $d = \omega(n^2)$.
Hinweis: Setze die vorherigen Teilaufgaben zusammen und verwende Stirlings Approximation der Fakultätsfunktion.