Karlsruher Institut für Technologie

Institut für Theoretische Informatik
T.T.-Prof. Dr. Thomas Bläsius
https://scale.iti.kit.edu

Übungsblatt 05

Algorithmen I - Sommersemester 2023

Abgabe im ILIAS bis 26.05.2023, 18:00 Uhr

Die Abgabe erfolgt als *eine* PDF-Datei über das Übungsmodul in der Gruppe deines Tutoriums im ILIAS.
Beachte bitte die Hinweise zum Bearbeiten auf der Webseite (neu).

Aufgabe 1 - Sublogarithmische Suche (3 Punkte)

Wir haben in der Vorlesung gelernt, dass vergleichsbasiertes Suchen in einer Menge mit n Elementen eine Laufzeit in $\Omega(\log(n))$ benötigt. Im Folgenden soll nun mithilfe stärkerer Annahmen an die Eingabe eine Datenstruktur entwickelt werden, in der das schneller geht.

Sei $m \in \mathbb{N}$ eine Zahl und $M = \{1, 2, \ldots, m\}$ (die ersten m natürlichen Zahlen) unsere Grundmenge. Gesucht ist nun eine Datenstruktur D, mit der man in konstanter Zeit die folgenden Operationen ausführen kann:

- `insert`: fügt ein Element aus M in D ein,
- `delete`: löscht ein Element aus D, und
- `find`: entscheidet, ob ein Element aus M in D enthalten ist.

Hinweis: Natürlich soll die Datenstruktur auch in der Lage sein Elemente mehrfach zu enthalten.

a) Beschreibe eine Datenstruktur, welche die gewünschten Anforderungen erfüllt. (1 Punkt)

b) Beschreibe, wie man mithilfe deiner Datenstruktur aus a) die Operationen `insert`, `delete` und `find` in konstanter Zeit ausführen kann. (1 Punkt)

c) Welches Problem entsteht bei der Verwendung deiner Datenstruktur aus a), wenn wir nur noch annehmen, dass die Grundmenge M eine beliebige geordnete Menge natürlicher Zahlen ist? (1 Punkt)

Aufgabe 2 - Unorganisierte Bauarbiber (5 Punkte)

Nachdem ausreichend Stämme für einen stabilen Damm gekauft wurden, hat Dr. Meta ausgerechnet die Biber damit beauftragt, das Holz zum Damm zu transportieren. Dabei wurde nicht bedacht, dass Biber zwar hervorragend im Verarbeiten von Holz sind, ihre organisatorischen Fähigkeiten aber nicht unterschätzt werden können.

Die Biber haben das Holz nun wild durcheinander aufgereiht, obwohl sie wissen, dass eine einzelne Schicht des Dammes nur Stämme mit derselben Stabilitätszahl enthalten darf und das die unterste Schicht (quasi das Fun*damm*ent) die meisten Stämme enthalten muss.

Um die Biber bei der super-effizienten Verarbeitung der Stämme nicht auszubremsen, musst du jetzt schnell rausfinden, mit welchem Holz gestartet werden soll.

In dieser Aufgabe darfst du davon ausgehen, mithilfe einer universellen Familie von Hashfunktionen Hashmaps verwenden zu können, die erwartet konstante Zugriffszeiten haben.

a) Beschreibe einen Algorithmus, der als Eingabe ein Array A mit n ganzzahligen Stabilitätszahlen erhält und in erwartet $O(n)$ Zeit ausgibt, welche Stabilitätszahl am häufigsten in A vorkommt. Begründe die Korrektheit deines Algorithmus und warum er das geforderte Laufzeitverhalten hat. (3 Punkte)

Sofort machen sich die Biber an die Arbeit! Schnell stellst du fest, dass der Damm mehr als nur eine Schicht haben wird und eigentlich die Häufigkeiten der Stabilitätszahlen in absteigender Reihenfolge gebraucht werden, um immer direkt entscheiden zu können, welches Holz für die nächste Schicht gebraucht wird.

b) Beschreibe eine Datenstruktur, mit der man, gegeben eine Zahl k, in konstanter Zeit die k-häufigste Stabilitätszahl bestimmen kann. Begründe, warum deine Datenstruktur in erwartet $O(n)$ Zeit erstellt werden kann. (2 Punkte)

Aufgabe 3 - Schlechte Hashfunktionen (5 Punkte)

Sei $M \in \mathbb{N}$, $U = \{0, 1, \ldots, M-1\}$ ein Universum an Schlüsseln und $m \ll M$ die Größe einer Hashtabelle. Gib für jede der folgenden „Hashfunktionen“ an, welche Nachteile bei der Verwendung auftreten können.

a) $h : x \mapsto (2x) \mod m$ für gerades m

b) $h : x \mapsto (m - \lfloor \frac{x}{m} \rfloor) \mod m$

c) $h : x \mapsto ((x-1) \mod m) + 4 \cdot \lfloor \frac{m}{x+2} \rfloor$

d) $h : x \mapsto (x + \text{RANDOM}(0,1)) \mod m$
wobei RANDOM bei jedem Aufruf jeweils mit Wahrscheinlichkeit $1/2$ entweder eine 0 oder eine 1 ausgibt.

e) $h : x \mapsto \lfloor \frac{M}{x+3} \rfloor \mod m$

Aufgabe 4 - Familien von Hashfunktionen (6 Punkte)

Sei $M \in \mathbb{N}$, $U = \{0, 1, \ldots, M-1\}$ ein Universum an Schlüsseln und $m \ll M$ die Größe einer Hashtabelle.

a) Wir sagen, eine Menge $\mathcal{H}$ von Hashfunktionen ist eine *c-fast-universelle* Familie, wenn es eine Konstante $c \geq 1$ gibt, sodass für alle $k_1, k_2 \in U$ mit $k_1 \neq k_2$ und ein zufälliges $h \in \mathcal{H}$ gilt, dass $\Pr[h(k_1) = h(k_2)] \leq c/m$.

   Sei nun $\mathcal{H}$ eine c-fast-universelle Hashfamilie für ein $c \geq 1$. Außerdem sei $(a_1, \ldots, a_n)$ eine Folge von Elementen, die nacheinander in eine Hashtabelle der Größe $m \in \Theta(n)$ eingefügt werden. Zeige, dass für eine zufällige Hashfunktion $h \in \mathcal{H}$ und jedes $i \in \{1, \ldots, n\}$ gilt, dass die erwartete Größe des Buckets, in das a_i eingefügt wird, in $\Theta(1)$ liegt. (2 Punkte)

b) Sei $\mathcal{H} = \{h : x \mapsto (23x + a) \mod m \mid a \in U\}$. Zeige, dass $\mathcal{H}$ keine universelle Familie ist. (2 Punkte)

c) Sei $\mathcal{H} = \{h : x \mapsto (a \cdot \lceil \log(x+1) \rceil) \mod m \mid a \in U\}$. Zeige, dass $\mathcal{H}$ keine universelle Familie ist. (2 Punkte)