Karlsruher Institut für Technologie

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

Übungsblatt 05

Algorithmen I - Sommersemester 2022

Abgabe im ILIAS bis 25.05.2022, 14:00 Uhr

Bitte beschrifte Deine Abgabe gut sichtbar mit Deinem Namen und Deiner Matrikelnummer. Achte insbesondere bei handschriftlichen Abgaben auf Lesbarkeit und genügend Platz für Korrektur-Anmerkungen. Die Abgabe erfolgt über das Übungsmodul in der Gruppe Deines Tutoriums im ILIAS. Gib Deine Ausarbeitungen in *einer* PDF-Datei ab.
Achte darauf, effiziente Algorithmen zu formulieren, also solche mit möglichst geringer asymptotischer Laufzeit!

Aufgabe 1 - RadixSort (3 Punkte)

Gib Pseudocode für den RADIXSORT Algorithmus aus Vorlesung 6 an, um n Zahlen aus dem Intervall $[0, n^c)$ in Linearzeit zu sortieren. Nutze dabei die folgende Signatur:

```
RADIXSORT(A: [ℕ; n], c: ℕ) : [ℕ; n]
```

Hinweis: Dir steht die Methode BUCKETSORT aus der sechsten Vorlesung zur Verfügung und du darfst beispielsweise über $A[i]$.key Keys für die Elemente in A setzen.

Lösung 1

```
1: RADIXSORT(A: [ℕ; n], c: ℕ) : [ℕ; n]
2:     for i ∈ {0, ..., c − 1} do
3:         for num ∈ A do
4:             num.key = (num div n^i) mod n
5:         end
6:         BUCKETSORT(A)
7:     end
8:     return A
```

Aufgabe 2 - Dr. Meta ermittelt (5 Punkte)

Noch immer fahndet Dr. Meta entschlossen nach dem Verräter in den eigenen Reihen. Mit deiner Hilfe konnte er mehrere verdächtige Personen unter seinen Handlangern ausmachen. Doch alle von ihnen behaupten, zu der Zeit, zu der in das Labor eingebrochen wurde, bei der Arbeit gewesen zu sein. Dabei sollen sie sogar von anderen Handlangern gesehen worden sein. Dr. Meta entschließt sich also, die angeblichen Zeugen zu befragen. Leider hat er sich dafür kein System überlegt, sondern notiert sich bei jeder Befragung nur, welcher Verdächtige gesehen wurde. Damit hat er nur eine Sammlung von n Verweisen auf je eine verdächtigte Person. Natürlich ist Dr. Meta nicht so naiv, einer einzigen Zeugenaussage zu vertrauen. Er betrachtet eine Person erst als entlastet, wenn k andere Handlanger für ihn oder sie bürgen. Doch auf welche der Verdächtigten trifft das zu? Du hast dich bisher als äußert kompetent in der Unterstützung von Dr. Meta erwiesen. Beweise dein Können erneut!
In dieser Aufgabe darfst du davon ausgehen, mithilfe einer universellen Familie von Hashfunktionen Hashmaps verwenden zu können, die erwartet konstante Zugriffszeiten haben.

1. Beschreibe einen Algorithmus, der für eine Zahl k und ein Array A mit n Einträgen in erwartet $O(n)$ Zeit entscheidet, ob es einen Eintrag $e \in A$ gibt, der mehr als k mal in A vorkommt. Begründe die Korrektheit deines Algorithmus und warum er das geforderte Laufzeitverhalten hat. (2 Punkte)

2. Beschreibe einen Algorithmus, der für eine Zahl k und ein Array A, welches n ganze Zahlen enthält, in erwartet $O(n)$ Zeit entscheidet, ob es Indizes i, j gibt, sodass $A[i] + A[j] = k$. Begründe die Korrektheit deines Algorithmus und warum er das geforderte Laufzeitverhalten hat. (3 Punkte)

Lösung 2

1. Zunächst legen wir eine Hashmap der Größe n an, welche natürliche Zahlen hält. Als Schlüssel werden Elemente aus A verwendet.
   Nun überprüfen wir für jeden Eintrag a in A, ob zum Schlüssel a bereits ein Eintrag in der Hashmap existiert. Wenn nicht, fügen wir der Wert 1 zum Schlüssel a in die Hashmap ein. Ansonsten erhöhen wir den gefundenen Wert um 1. Sollte der Wert zum Schlüssel a dann größer als k sein, haben wir einen Eintrag gefunden, der mehr als k mal in A vorkommt.
   Der Zähler für einen Wert a' in A wird genau dann erhöht, wenn zum den Schlüssel a' in die Hashabelle eingefügt werden soll. Damit zählen wir, wie oft der Eintrag in A vorkommt. Da wir jedes der n Elemente in A genau ein Mal in erwartet $\Theta(1)$ in der Hashmap suchen und maximal ein Mal in erwartet $\Theta(1)$ einfügen, liegt die Laufzeit dieses Algorithmus in erwartet $\Theta(n)$.

2. Zunächst legen wir eine Hashmap der Größe n an, welche natürliche Zahlen hält. Als Schlüssel werden auch hier die Elemente aus A verwendet.

Nun fügen wir alle Einträge von $a = A[l]$ den Wert l zum Schlüssel a in die Hashmap ein. Anschließend überprüfen wir für jeden Eintrag $a' = A[i]$, ob ein Eintrag j zum Schlüssel $k - a'$ in der Hashmap existiert. Wenn das der Fall ist, hat das Indexpaar i, j die gewünschte Eigenschaft. Es gilt in diesem Fall $A[i] + A[j] = k$, denn es muss $A[j] = k - A[i]$ sein. Der Algorithmus fügt n Einträge in jeweils erwartet $\Theta(1)$ in die Hashmap ein und fragt dann maximal n Mal in jeweils erwartet $\Theta(1)$ einen Wert in der Hashmap ab. Damit liegt die Gesamtlaufzeit in erwartet $\Theta(n)$.

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, wieso sie keine geeignete Hashfunktion ist.

1. $h : x \mapsto (2x + 1) \mod m$ für gerades m
2. $h : x \mapsto \lfloor \frac{x}{m} \rfloor \mod m$
3. $h : x \mapsto (x \mod m) + \lfloor \frac{m}{x+1} \rfloor$
4. $h : x \mapsto (x + \text{RANDOM}(1, m-1)) \mod m$
   wobei RANDOM bei jedem Aufruf eine zufällige Zahl ausgibt, die uniform zwischen 1 und $m - 1$ gewählt wird.
5. $h : x \mapsto \lfloor \frac{M}{x+1} \rfloor \mod m$

Lösung 3

1. Da m gerade ist, ist auch $(2x + 1) \mod m$ für beliebige x ungerade. Also bildet h auf keinen geraden Wert und damit nur auf maximal die Hälfte der Elemente von $\{0, \ldots, m-1\}$ ab.
2. Ist $x < m$, so ist $\lfloor \frac{x}{m} \rfloor = 0$. Damit ist die Kollisionswahrscheinlichkeit für alle Eingaben kleiner m gleich 1.
3. Es gilt $h(m-1) = m - 1 + 1 = m$, doch die Hashtabelle hat nur m Einträge mit Indizes $0, \ldots, m-1$.
4. Die Ausgabe von h für einen Wert x ist nicht reproduzierbar und damit für eine Hashmap ungeeignet.
5. Alle $x \in U$ mit $x \geq \frac{M}{2}$ und damit etwa die Hälfte der Elemente in U, werden von h auf den Wert 1 abgebildet. Ebenso werden für alle $i \in \{0, \ldots, \frac{M}{2} - 1\}$ alle $y \in U$ mit $\frac{M}{i+1} \leq y \leq \frac{M}{i} - 1$ auf den gleichen Wert abgebildet. Die Hashtabelle würde somit sehr ungleichmäßig gefüllt.

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.

1. Wir sagen eine Menge $\mathcal{H}$ von Hashfunktionen ist eine *fast universelle* Familie, wenn für alle $k_1, k_2 \in U$ mit $k_1 \neq k_2$ und ein zufälliges $h \in \mathcal{H}$ gilt, dass $\Pr[h(k_1) = h(k_2)] \leq 2/m$. Gegeben sei eine Hashtabelle auf Basis einer Hashfunktion aus einer fast universellen Familie. Nimm an, dass darin n Elemente eingefügt wurden und zeige, dass die erwartete Größe eines Buckets immer noch konstant ist, wenn $m \in \Theta(n)$. (2 Punkte)

2. Sei $\mathcal{H} = \{h : x \mapsto (42x + a) \mod m \mid a \in U\}$. Zeige, dass $\mathcal{H}$ keine universelle Familie ist. (2 Punkte)

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

Lösung 4

1. Sei A das Array der Hashtabelle. Wir bestimmen den Erwartungswert $\mathbb{E}[|A[h(k)]|]$ für einen beliebigen Schlüssel $k \in U$. Für die n eingefügten Schlüssel $k_1, \ldots, k_n$ definieren wir Indikatorvariablen X_i mit $i \in \{1, \ldots, n\}$, wobei

$$X_i = \begin{cases} 1 & \mid \text{ falls } h(k) = h(k_i) \\ 0 & \mid \text{ sonst} \end{cases}$$

   Da die Wahrscheinlichkeit für eine Kollision durch $\Pr[X_i = 1] = \Pr[h(k) = h(k_i)] = 2/m$ ist, ergibt sich die erwartete Anzahl Schlüssel die auf $A[h(k)]$ abgebildet werden als

$$\begin{aligned} \mathbb{E}[|A[h(k)]|] &= \mathbb{E}\Big[\sum_{i=1}^{n} X_i\Big] \\ &= \sum_{i=1}^{n} \mathbb{E}[X_i] \\ &= \sum_{i=1}^{n} \Pr[X_i = 1] \\ &= \begin{cases} \frac{2n}{m} & \mid \text{ falls } k \notin \{k_1, \ldots, k_n\} \\ 1 + \frac{2n-2}{m} & \mid \text{ sonst} \end{cases} \end{aligned}$$

   Also: Ist $m \in \Theta(n)$, dann gilt $\mathbb{E}[|A[h(k)]|] \in \Theta(1)$.

2. Es gilt zu zeigen, dass es $k_1, k_2 \in U$ mit $k_1 \neq k_2$ so gibt, dass für ein zufälliges $h \in \mathcal{H}$ gilt, dass $\Pr[h(k_1) = h(k_2)] > \frac{1}{m}$. Dazu wählen wir $k_1 \in U$ beliebig und

$k_2 = k_1 \pm m$. Sei $h \in \mathcal{H}$ beliebig. Dann:

$$\begin{aligned} h(k_2) &= (42(k_1 \pm m) + a) \mod m \\ &\equiv (42k_1 + a \pm 42m) \mod m \\ &\equiv (42k_1 + a) \mod m \\ &\equiv h(k_1) \end{aligned}$$

Also ist die Kollisionswahrscheinlichkeit für k_1 und k_2 gleich 1.

3. Auch hier suchen wir $k_1, k_2 \in U$ mit $k_1 \neq k_2$ so, dass für ein zufälliges $h \in \mathcal{H}$ gilt, dass $\Pr[h(k_1) = h(k_2)] > \frac{1}{m}$. Dazu wählen wir $k_1 \in U$ beliebig und $k_2 = k_1 + i \cdot m$ für ein $i \in \mathbb{Z}$ (Dies ist möglich, da $|U| \gg m$). Sei $h \in \mathcal{H}$ beliebig. Dann:

$$\begin{aligned} h(k_2) &= a \cdot k_2^2 \mod m \\ &\equiv a \cdot (k_1 + i \cdot m)^2 \mod m \\ &\equiv a \cdot (k_1^2 + 2im \cdot k_1 + (im)^2) \mod m \\ &\equiv ak_1^2 \mod m \\ &\equiv h(k_1) \end{aligned}$$

Also ist die Kollisionswahrscheinlichkeit für k_1 und k_2 gleich 1.