ETH Zürich
Institute of Theoretical Computer Science
Dr. Johannes Lengler, Prof. Emo Welzl
Dr. Anders Martinsson, Charlotte Knierim

FS 2021

Algorithmen und Wahrscheinlichkeit
Peer-Aufgaben 4

Abgabe in Moodle (https://moodle-app2.let.ethz.ch/mod/workshop/view.php?id=586806) bis zum 29.04.2021 um 16:15 Uhr.

Das Peergrading findet in diesem Semester zum ersten Mal auch über Moodle statt. Bitte lösen Sie die Aufgabe selbstständig und laden Sie Ihre Lösung in Moodle hoch. Ihnen wird dann eine Lösung zum korrigieren zugewiesen. Die Korrektur erfolgt Donnerstag nach der Übung bis spätestens um Mitternacht am folgenden Sonntag. Sie erhalten die Bonuspuknte für einen ehrlichen Versuch die Aufgabe bestmöglich zu lösen und die Ihnen zugeteilte Abgabe bestmöglich zu koregieren.

Aufgabe 1 – *Hash tables*

Das folgende Problem ist als 'Element-Uniqueness-Problem' bekannt: Gegeben ist ein Array von Zahlen $a_1, \dots, a_n$, zum Beispiel vom Typ `double`. Wir wollen herausfinden, ob zwei der Elemente des Arrays identisch sind, also ob es $1 \leq i < j \leq n$ gibt, so dass $a_i = a_j$.

Deterministisch lässt sich dieses Problem mithilfe von zwei for-loops in Zeit $O(n^2)$ lösen. Ein etwas geschickterer Ansatz, der einen schnellen Sortieralgorithmus verwendet, benötigt Zeit $O(n \ln n)$. In der Praxis lässt sich aber mit Hash-Tables ein noch schnellerer Ansatz finden.

Wir nehmen an, dass eine Hashfunktion h gegeben ist, die als Input Zahlen vom Typ `double` verwendet und als Ausgabe eine ganze Zahl zwischen (einschliesslich) 1 und n generiert. Die Idee ist, zunächst n Listen $L_1, \dots, L_n$ zu generieren, wobei die Liste L_k alle Indizes $i \in \{1, \dots n\}$ enthält sodass $h(a_i) = k$, und dann jede Liste L_k genauer zu untersuchen.

(a) Seien n, $a_1, \dots, a_n$ und die Hashfunktion h gegeben. Sei

$$C = \{(i, j) : a_i \neq a_j \text{ und } h(a_i) = h(a_j)\}.$$

Beschreiben Sie einen Algorithmus der das Element-Uniqueness-Problem in Zeit $O(n + |C|)$ löst und $O(n)$ Speicherplatz benötigt.

Wir nehmen hierbei an, dass $h(x)$ in Zeit $O(1)$ berechnet werden kann.

(b) Angenommen, h wird zufällig gewählt, sodass für alle $a \neq b$ gilt

$$\Pr[h(a) = h(b)] = \frac{1}{n}.$$

Zeigen Sie, dass die erwartete Laufzeit des in (a) konstruierten Algorithmus $O(n)$ ist.

ETH Zürich
Institute of Theoretical Computer Science
Dr. Johannes Lengler, Prof. Emo Welzl
Dr. Anders Martinsson, Charlotte Knierim

FS 2021

Algorithmen und Wahrscheinlichkeit
Peer-Aufgaben 4 — Lösung

Aufgabe 1 – *Hash tables*

(a) Wir betrachten den folgenden Algorithmus: Zuerst generieren wir n Listen $L_1, \dots, L_n$, wobei die Liste L_k alle Indizes $i \in \{1, \dots n\}$ enthält sodass $h(a_i) = k$. Für jede Liste L_k und für jedes Paar (i, j) von Indizes in L_k überprüfen wir nun, ob $a_i = a_j$. Wenn $a_i = a_j$ geben wir aus, dass wir zwei identische Elemente gefunden haben. Wenn $a_i \neq a_j$ für alle überprüften Paare, geben wir aus, dass alle Elemente verschieden sind.

Laufzeit: Mit der Annahme, dass $h(x)$ in konstanter Zeit berechent werden kann, können wir die Listen in Zeit $O(n)$ generieren. Das durchlaufen aller Listen braucht dann ebenfalls Zeit $O(n)$, da alle Listen zusammen insgesamt n Elemente haben. Die ausgeführten Vergleiche benötigen Zeit $O(|C|)$, da jeder Vergleich, bei dem nicht abgebrochen wird, einem Element von C entspricht. Die totale Laufzeit des Algorithmus ist also $O(n + |C|)$.

Korrektheit: Zuerst zeigen wir, dass wir zwei identische Elemente $a_i = a_j$, $i \neq j$, tatsächlich finden. Seien also i und j verschiedene Indizes mit $a_i = a_j$. Insbesondere ist auch $h(a_i) = h(a_j) := b$. Also werden sowohl i als auch j der Liste L_b hinzugefügt. Somit werden beim durchlaufen von L_b die Elemente a_i und a_j verglichen und der Algorithmus erkennt, dass sie gleich sind.

Es bleibt zu zeigen, dass der Algorithmus keine gleichen Elemente findet, wenn keine existieren. Dies folgt aber sofort aus der Tatsache, dass der Algorithmus nur dann zwei gleiche Elemente meldet, wenn er tatsächlich solche gefunden hat.

(b) Wir haben bereits gezeigt, dass die Lauzeit des Algorithmus $O(n + |C|)$ ist. Wir definieren nun eine Zufallsvariable $X = |C|$, die die Kardinalität von C angibt. Wir behaupten, dass $\mathbb{E}[X]$ in $O(n)$ ist. Darus folgt sofort, dass unser Algorithmus insgesamt in Zeit $O(n + |C|) = O(n)$ läuft.
Um zu beweisen, dass $\mathbb{E}[X] = O(n)$, definieren wir für jedes Paar (i, j) von Indizes eine Indikatorvariable $X_{(i,j)}$ die genau dann 1 ist, wenn $(i, j) \in C$. Insbesondere gilt also $X = \sum_{(i,j)\in I} X_{(i,j)}$. Weiter gilt aufgrund unserer Annahme über h, dass

$$\Pr[X_{(i,j)} = 1] = \begin{cases} \frac{1}{n} & \text{wenn } a_i \neq a_j \\ 0 & \text{wenn } a_i = a_j. \end{cases}$$

Also ist in jedem Fall $\mathbb{E}[X_{(i,j)}] = \Pr[X_{(i,j)} = 1] \leq \frac{1}{n}$. Mit der Linearität des Erwartungswertes erhalten wir nun

$$\mathbb{E}[X] = \sum_{(i,j)\in I} \mathbb{E}\left[X_{(i,j)}\right] \leq \sum_{(i,j)\in I} \frac{1}{n} = n(n-1)\frac{1}{n} \in O(n).$$