Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
P. Bamberger
P. Schneider

Algorithmen und Datenstrukturen
Sommersemester 2020
Übungsblatt 4

Abgabe: Mittwoch, 10.06.2020, 16:00 Uhr.

Aufgabe 1: Hashing mit offener Adressierung *(10 Punkte)*

Wir betrachten Hashtabellen mit offener Adressierung und zwei Methoden zur Auflösung von Kollisionen: *Lineares Sondieren* und *Doppel-Hashing*. Sei m die Größe der Hashtabelle wobei m Primzahl ist. Sei $h_1(x) := 53 \cdot x$ und $h_2(x) := 1 + (x \bmod (m-1))$. Wir definieren die folgenden beiden Hashfunktionen zur Kollisionsauflösung gemäß der Vorlesung:

- Lineares Sondieren: $h_\ell(x, i) := (h_1(x) + i) \mod m$.
- Doppel-Hashing: $h_d(x, i) := (h_1(x) + i \cdot h_2(x)) \mod m$.

(a) Implementieren Sie eine Hashtabelle mit den Operationen `insert` und `find` gemäß der Vorlesung und den genannten Strategien zur Kollisionsauflösung. Sie *können* dazu die Vorlage `HashTable.py` benutzen. *(5 Punkte)*

(b) Erstellen Sie eine Hashtabelle der Größe $m > 1000$ (m prim) und messen Sie die *durchschnittliche* Laufzeit für das Einfügen von k Schlüsseln für $k \in \{\lfloor \frac{m \cdot i}{50} \rfloor \mid i = 1, \ldots, 49\}$. Wiederholen Sie das Experiment in vier Variationen: Mit linearem Sondieren bzw. Doppel-Hashing; mit k zufälligen Schlüsseln[1] bzw. der vorgegebenen Schlüsselmenge $\{m \cdot i \mid i = 1, \ldots, k\}$. Stellen Sie die durchschnittlichen Laufzeiten für das Einfügen grafisch dar. Diskutieren Sie die Schaubilder in Ihren `erfahrungen.txt`[2]. *(5 Punkte)*

Aufgabe 2: Anwendung von Hashtabellen *(10 Punkte)*

Gegeben ist folgender Algorithmus:

Algorithm 1 `algorithm` ▷ Input: Array A of length n with integer entries

```
1: for i = 1 to n − 1 do
2:     for j = 0 to i − 1 do
3:         for k = 0 to n − 1 do
4:             if |A[i] − A[j]| = A[k] then
5:                 return true
6: return false
```

(a) Beschreiben Sie, was `algorithm` berechnet und analysieren Sie die asymptotische Laufzeit. *(3 Punkte)*

[1] Zufallswerte aus $\{0, \ldots, z\}$ ohne Duplikate mit $z \gg m$. Bspw mit `random.sample(range(z+1), k)`.

[2] Nutzen Sie die `erfahrungen.txt` gerne weiterhin um uns Feedback über Dauer und Schwierigkeiten der Bearbeitung des Übungsblattes mitzuteilen.

(b) Beschreiben Sie einen alternativen Algorithmus $\mathcal{B}$ für dieses Problem (d.h. $\mathcal{B}(A) = \texttt{algorithm}(A)$ für jede Eingabe A) mit einer Laufzeit von $\mathcal{O}(n^2)$, welcher Hashing verwendet. *(3 Punkte)*

Hinweis: Sie dürfen annehmen, dass das Einfügen und Finden von Schlüsseln in einer Hashtabelle $\mathcal{O}(1)$ Zeitschritte benötigt, wenn $\alpha = \mathcal{O}(1)$ (α ist der Load der Hashtabelle).

(c) Beschreiben Sie einen weiteren Algorithmus für dieses Problem ohne Verwendung von Hashing mit einer Laufzeit von $\mathcal{O}(n^2 \log n)$. *(4 Punkte)*

Hinweis: Nutzen Sie Sortierung.