TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Fachbereich Informatik
AG Algorithmen und Komplexität

Ausgabedatum: 18.12.2014
Version: 2014-12-17 15:59

8. Übungsblatt für Track AI zur Vorlesung Entwurf und Analyse von Algorithmen, WS 14/15

Abgabe: *Bis Freitag, 09.01.2015, 12:00 Uhr, Kasten im Treppenhaus 48-6.*

Basisaufgaben

B8.1: Einfügen in Hashtabellen

3 Punkte

Wir betrachten Hashing mit dem Universum $\mathbb{N}$ und der Hash-Funktion $h(x) = x^2 \bmod 17$. Unsere Hash-Tabelle habe entsprechend die Adressen $\{0, 1, 2, \ldots, 16\}$. Gegeben ist die Schlüsselfolge $16, 22, 5, 19, 4, 12, 1, 7, 10, 3$.

a) Wie sieht die Hash-Tabelle nach dem Einfügen der Schlüsselfolge in die anfangs leere Hash-Tabelle bei Verwendung von *linear probing* für $\gamma = 3$ aus?

b) Wie sieht die Hash-Tabelle nach dem Einfügen der Schlüsselfolge in die anfangs leere Hash-Tabelle bei Verwendung von *quadratic probing* aus?

c) Wie sieht die Hash-Tabelle nach dem Einfügen der Schlüsselfolge in die anfangs leere Hash-Tabelle bei Verwendung von *add to hash* aus?

B8.2: Hashing von Strings

3 Punkte

a) Sei Σ ein endliches Alphabet. Entwerfen Sie eine *fast uniforme* Hashfunktion für Wörter der Länge n über Σ, also

$$\text{hash} : \Sigma^n \to [0..m] \,,$$

und begründen Sie die Korrektheit Ihrer Funktion. „Fast uniform“ soll hier heißen, dass

$$|\{w \mid \text{hash}(w) = i\}| \;\in\; \left\{ \left\lfloor \frac{|\Sigma|^n}{m+1} \right\rfloor, \left\lceil \frac{|\Sigma|^n}{m+1} \right\rceil \right\}$$

für alle $i \in [0..m]$.

b) Entwerfen Sie eine fast uniforme Hashfunktion

$$\text{hash} : \{\texttt{A}, \texttt{B}, \ldots, \texttt{Z}\}^3 \to [1..5] \,,$$

für die $\text{hash}(\texttt{FCK}) = 2$, und begründen Sie die Korrektheit Ihrer Funktion.

B8.3: Universelles Hashing

3 Punkte

a) Zeigen Sie, dass für eine geeignete Konstante c und für alle $N, m \in \mathbb{N}$ die Menge $\mathcal{H}_{N,m}$ aller Hashfunktionen $h : [1 \ldots N] \to [0 \ldots m]$ c-universell ist. Bestimmen Sie den minimalen Wert von c.

b) Warum verwendet man in der Praxis nicht einfach $\mathcal{H}_{N,m}$, die Menge *aller* Hashfunktionen, wo sie doch nach a) c-universell ist?

B8.4: Rekursion – Analyse und Optimierung

3 Punkte

Betrachten Sie den folgenden Algorithmus:

```
1 procedure e(n) {
2   result = 1
3   for i = 0 to n-1 {
4     result = result + i + e(i)
5   }
6   return result
7 }
```

a) Weisen Sie nach, dass der Algorithmus (mindestens) exponentielle Zeit in `n` benötigt, um `e(n)` zu berechnen.

b) Geben Sie einen ergebnisäquivalenten Algorithmus an, der in polynomieller Zeit in `n` läuft. Bestimmen Sie Laufzeit und Speicherbedarf Ihres Algorithmus.

Aufbauaufgaben

A8.1: Wörterbuch mit Undo

5 Punkte

Entwerfen Sie eine Datenstruktur für ganze Zahlen, die die folgenden Operationen mit den gegebenen asymptotischen Laufzeiten (für n die Anzahl der gespeicherten Elemente) unterstützt:

`FIND(e)` prüft in erwarteter Zeit $\mathcal{O}(\log n)$, ob das gegebene Element e enthalten ist.

`INSERT(e)` fügt ein neues Element e in erwarteter Zeit $\mathcal{O}(\log n)$ hinzu.

`UNDO` entfernt in Zeit $\mathcal{O}(1)$ das Element aus der Datenstruktur, das (unter allen enthaltenen) zuletzt hinzugefügt wurde, und gibt es zurück.

Erwartete Zeit heißt hier, dass wir annehmen, dass

1. die untersuchte Operation nach n `INSERT`-Operationen mit paarweise verschiedenen Schlüsseln aus E, $|E| \geq n$ durchgeführt wird, wobei die Permutation dieser Schlüssel uniform zufällig aus allen möglichen gewählt ist, und dass außerdem
2. der Parameter `e` uniform zufällig aus E bzw. im Falle von `INSERT` aus E ohne die bereits enthaltenen Elemente gewählt wird.

Der Speicherbedarf der Datenstruktur soll in $\Theta(n)$ sein.

A8.2: Echtes Löschen in AVL-Bäumen

4 Punkte

Implementieren Sie die Operation `DELETE` für AVL-Bäume, sodass sie stets in Zeit $\mathcal{O}(\log n)$ läuft, und begründen Sie die Korrektheit.

Benutzen Sie die gleiche Baumrepräsentation wie `INSERT` im Buch, also mehrfache Verkettung ohne Vaterzeiger. Sie können eine geeignete Implementierung der Rotationsoperationen und der Neuberechnung des Balancegrades eines einzelnen Knotens als gegeben annehmen; bitte spezifizieren Sie deren Schnittstellen.

A8.3: AVL-order-statistic Bäume

4 Punkte

Erweitern Sie unsere Datenstruktur der AVL-Bäume – das schließt die Basisoperationen auf ihnen ein – so, dass in (Worst-Case-)Zeit $\mathcal{O}(\log n)$ das k-kleinste Element in einem AVL-Baum mit n Elementen gefunden werden kann, wobei $k \in [1..n]$. Die (asymptotischen) Laufzeiten der anderen Operationen sollen unverändert bleiben.

Begründen Sie die Korrektheit Ihrer Erweiterung.