Technische Universität
Kaiserslautern

Fachbereich Informatik
AG Algorithmen und Komplexität

Ausgabedatum: 06.12.2012
Version: 2012-12-11 17:11

9. Übungsblatt zur Vorlesung Entwurf und Analyse von Algorithmen, WS 12/13

Abgabe: *Bis Donnerstag, 13.12.2012, 12:00 Uhr, Abgabenkasten vor 48-694.*

53. Aufgabe

	a)	b)
Track ε	2	3
Track AI	2	[3]

Sei $f_{n,k}$ die Zahl der Teilmengen von $\{1, \ldots, n\}$, die

- Größe k haben und
- keine zwei aufeinanderfolgenden Zahlen enthalten.

a) Geben Sie eine Rekursionsgleichung für $f_{n,k}$ an.

b) Lösen Sie die Rekursionsgleichung aus a), finden Sie also eine geschlossene Form für $f_{n,k}$.

Hinweis: Sie können sich das Leben vereinfachen, indem Sie nicht beide Parameter gleichzeitig variabel lassen.

54. Aufgabe

	a)	b)
Track ε	2	[2]
Track AI	2	[2]

Wir erinnern uns: Ein Baum heißt α-balanciert, wenn für alle Knoten v mit linkem Kind l gilt, dass $\frac{1+|T_l|}{1+|T_v|} \in [\alpha, 1-\alpha]$, mit $\alpha \in [0, \frac{1}{2}]$ fest.

a) Zeigen oder widerlegen Sie: Wenn man AVL-Bäume mit dem Gewichtsbalancegrad annotiert und wie gewohnt rotiert, wenn diese verletzt sind, genügen nach Einfügen eines neuen Elements in einen α-balancierten Baum stets eine konstante Anzahl der bekannten AVL-Rotationen, um α-Balance wieder herzustellen.

b) Recherchieren Sie nach Antworten auf die folgenden Fragen:

   i) Was für Implementierungen von gewichts- bzw. α-balancierten Suchbäumen gibt es?

   ii) Wie verhalten sich die Laufzeiten der wesentlichen Operationen, insbesondere Einfügen und Suchen von Schlüsseln?

Listen Sie Ihre Funde mit kurzen Zusammenfassungen in einer strukturierte Übersicht, natürlich mit sachgemäßen Referenzen!

Hinweis: Als Ergebnis sind sowohl eine Liste von (nicht zwingend *allen* erdachten) Implementierungen, aber auch Verweise auf nicht konstruktive Positiv- wie Negativergebnisse denkbar! Eigene Ansätze sind natürlich ebenfalls gerne gesehen.

55. Aufgabe

	a)	b)	c)
Track ε	1	1	3
Track AI	1	1	3

Ein *Rot-Schwarz-Baum* ist ein erweiterter binärer Suchbaum mit den folgenden Eigenschaften:

(1) Jeder Knoten ist entweder rot oder schwarz.

(2) Die Wurzel ist schwarz.

(3) Ist ein Knoten rot, so sind alle seine Kinder schwarz.

(4) Sei v ein beliebiger Knoten und T_v der Teilbaum, dessen Wurzel v ist. Dann haben alle (einfachen) Pfade von v zu fiktiven Blättern von T_v die gleiche Anzahl an schwarzen Knoten.

In einem Rot-Schwarz-Baum müssen alle fiktiven Blätter die selbe Farbe bekommen; per Konvention legen wir fest, dass alle fiktiven Blätter schwarz sind.

a) Zeigen oder widerlegen Sie: Es gibt $n_0 \in \mathbb{N}$, sodass alle Rot-Schwarz-Bäume mit $n \geq n_0$ Knoten höhenbalanciert sind.

b) Zeigen oder widerlegen Sie: Es gibt $n_0 \in \mathbb{N}$ und $\alpha \in \left(0, \frac{1}{2}\right]$, sodass alle Rot-Schwarz-Bäume mit $n \geq n_0$ Knoten α-balanciert sind.

c) Zeigen oder widerlegen Sie: in Rot-Schwarz-Bäumen ist die Suchzeit – erfolgreich wie -los – in $\mathcal{O}(\log n)$, n die Knotenzahl.

56. Aufgabe

	a)	b)	c)
Track ε	[1]	[1]	[1]
Track AI	[1]	[1]	[1]

Wir betrachten Hashing mit dem Universum $\mathbb{N}$ und der Hash-Funktion $h(x) = x^2 \mod 17$. 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?

57. Aufgabe

Track ε	[2]
Track AI	[2]

Wir betrachten das folgende offene Verfahren zur Beseitigung von Kollisionen beim Hashing: Im Falle einer Kollision an der Speicherzelle mit Adresse a werden nacheinander die Adressen

$$
\begin{aligned}
&a + 1^2 = a + 1 \,, \\
&a + 2^2 = a + 4 \,, \\
&a + 3^2 = a + 9 \,, \\
&a + 4^2 = a + 16 \,, \\
&\quad \vdots
\end{aligned}
$$

modulo der Größe der Hashtabelle aufgesucht, bis eine freie Zelle gefunden wird.

Zeigen Sie, dass man bei diesem Verfahren immer eine freie Speicherzelle findet, wenn die Größe der Hashtabelle eine Primzahl größer als 3 und die Hashtabelle mindestens halb leer ist.

58. Aufgabe

	a)	**b)**
Track ε	[2]	[1]
Track AI	2	1

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

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

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 $\text{hash}(\texttt{FCK}) = 2$.