

## Übungsblatt 5

Abgabe bis zum 28. Juni 2011

### Aufgabe 28

*mündlich*

Geben Sie einen Algorithmus an, der  $m$  sortierte Listen der Gesamtlänge höchstens  $n$  in Zeit  $O(n \log m)$  zu einer sortierten Liste zusammenfügt.

*Hinweis:* Verwenden Sie die Prioritätswarteschlange aus Aufgabe 27.

### Aufgabe 29

**4 Punkte**

Implementieren Sie folgende Suchbaum-Prozeduren.

- (a) `Sort( $B$ )` gibt alle Schlüssel in  $B$  in sortierter Reihenfolge aus. (*mündlich*)
- (b) `Height( $B$ )` gibt die Höhe von  $B$  zurück. (*mündlich*)
- (c) `AverageLeafDepth( $B$ )` gibt die durchschnittliche Tiefe der Blätter in  $B$  zurück. (*mündlich*)
- (d) `AverageNodeHeight( $B$ )` gibt die durchschnittliche Höhe der Knoten in  $B$  zurück. (**4 Punkte**)

### Aufgabe 30 Zeigen Sie:

*mündlich*

- (a) Wird als Einfügesequenz eine zufällige Permutation von  $n$  verschiedenen Zahlen benutzt, so hat der resultierende binäre Suchbaum  $B$  eine mittlere Knotentiefe von  $O(\log n)$ .

*Hinweis:* Zeigen Sie, dass die *Summe der Knotentiefen* von  $B$  mit der durchschnittlichen Anzahl von Vergleichen von `QuickSort` beim Sortieren einer zufälligen Permutation von  $n$  verschiedenen Zahlen übereinstimmt.

- (b) Die mittlere Laufzeit der Prozedur `ST-Search( $B, k$ )` ist  $O(\log n)$ , falls  $B$  aus einer zufälligen Einfügesequenz von  $n$  verschiedenen Zahlen generiert wurde und  $k$  ein zufälliger Wert dieser Folge ist.

### Aufgabe 31

*mündlich, optional*

Implementieren Sie folgende AVL-Prozeduren.

- (a) `LeftRotate( $y$ )`, `LeftRightRotate( $y$ )` und `RightLeftRotate( $y$ )`.
- (b) `AVL-Remove( $B, z$ )`.

### Aufgabe 32

*mündlich*

Gegeben sei die Einfügesequenz 5, 1, 6, 2, 4, 3.

- (a) Geben Sie den Suchbaum  $S$  an, den die Prozedur `ST-Insert` bei dieser Einfügesequenz erzeugt.
- (b) Geben Sie den AVL-Baum  $T$  an, den die Prozedur `AVL-Insert` bei dieser Einfügesequenz erzeugt.
- (c) Wie groß ist jeweils die Wahrscheinlichkeit, dass `ST-Insert` (bzw. `AVL-Insert`) bei Eingabe einer zufälligen Permutation auf der Schlüsselmenge  $\{1, 2, 3, 4, 5, 6\}$  genau diesen Suchbaum  $S$  (bzw. AVL-Baum  $T$ ) erzeugt?

### Aufgabe 33

*mündlich*

Geben Sie Pseudocode für eine nicht-rekursive Variante der Prozedur `DFS-Explore` an, die anstelle eines Kellers das Feld `parent` für das Backtracking benutzt.

### Aufgabe 34

*mündlich*

Implementieren Sie folgende Prozeduren für einen (un)gerichteten Graphen  $G = (V, E)$  mit fester Knotenmenge  $V = \{1, \dots, n\}$ , falls  $G$  in einer Adjazenzmatrix gespeichert wird.

- (a) `Init( $G$ )` initialisiert  $G$  als den leeren Graphen  $E_n$ .
- (b) `InsertEdge( $u, v$ )` fügt der Kantenmenge  $E$  eine neue Kante  $\{u, v\}$  bzw.  $(u, v)$  hinzu.
- (c) `RemoveEdge( $u, v$ )` entfernt die Kante  $\{u, v\}$  bzw.  $(u, v)$ .
- (d) `Edge( $u, v$ )` testet, ob die Kante  $\{u, v\}$  bzw.  $(u, v)$  in  $E$  ist.

Geben Sie jeweils asymptotische Schranken für die Laufzeit ihrer Prozeduren an. Überlegen Sie, wie sich für diese Prozeduren eine konstante Laufzeit realisieren lässt.

### Aufgabe 35

**6 Punkte**

Sei  $G = (V, E)$  ein Digraph mit  $V = \{1, \dots, n\}$ . Eine Permutation  $t$  auf  $V$  heißt *topologische Sortierung* von  $G$ , falls für jede Kante  $(u, v) \in E$  gilt:  $t(u) < t(v)$ .

- (a) Zeigen Sie, dass für  $G$  genau dann eine topologische Sortierung existiert, wenn  $G$  azyklisch ist. (*mündlich*)
- (b) Geben Sie eine Prozedur `TopSort` an, die für einen als Feld von Adjazenzlisten gespeicherten Digraphen  $G$  in Linearzeit  $O(n + m)$  testet, ob  $G$  azyklisch ist und gegebenenfalls eine topologische Sortierung ausgibt. (**2 Punkte**)

*Hinweis:* Führen Sie auf  $G^T = (V, E^T)$  eine Tiefensuche aus und geben Sie die Knoten in der Reihenfolge aus, in der sie zum letzten Mal besucht werden.

- (c) Beweisen Sie sowohl die Laufzeitschranke als auch die Korrektheit Ihrer Prozedur. (**4 Punkte**)