Informatik II SS 2006

Übungsblatt 5

Abgabe: 30.5.06 - 2.6.06

Aufgabe 1 (DST)

Punkte: 2+1+3

a) Geben Sie eine Reihenfolge an, in der die Schlüssel A, B, C, D, E, F, G in einen anfangs leeren DST eingefügt werden können, so daß

- ein vollständiger Binärer Suchbaum entsteht
- ein Baum entsteht, in dem jeder Knoten einen kleineren Schlüssel enthält als all seine Kindknoten. (Einen Baum mit dieser Eigenschaft nennt man *Heap* und wird z.B. für ein Sortierverfahren verwendet.)

b) Gegeben sind die Schlüssel 0001, 0010,...,1110,1111. Geben Sie eine Einfügereihenfolge in einen anfangs leeren DST an, welcher die maximal mögliche Höhe hat.

c) Fügen Sie die folgenden Schlüssel in einen anfangs leeren Digitalen Suchbaum in dieser Reihenfolge ein: S,U,C,H,B,A,M
Wie sieht der Binäre Trie dazu aus, und wie der Patricia Trie?

Gehen Sie in **a)** und **c)** von einer ASCII-Kodierung der Zeichen aus (siehe ASCII-Tabelle aus Informatik I). Die höchsten Bits, die für alle Zeichen in der jeweiligen Teilaufgabe gleich sind, sollen weggelassen werden.

Aufgabe 2 (Tries in Python)

Punkte: 1+1+2+3+1+2

Entwerfen Sie eine Klasse *Trie*, die Binäre Tries implementiert. Die Routine *insert* können Sie von den Vorlesungsfolien übernehmen. Implementieren Sie

- **a)** eine Routine *search(key)* die den Schlüssel *key* im Trie sucht und bei Erfolg ausgibt.
- **b)** eine Routine *sort()*, welche die Schlüssel eines Tries in sortierter Form ausgibt.
- **c)** eine Routine *select(k)* welche den k-grössten Schlüssel eines Tries ausgibt.
- **d)** eine Routine *remove(key)* welche den Schlüssel *key* aus einem Trie löscht. Falls der Schlüssel nicht im Baum existiert soll der Baum unverändert bleiben.
- **e)** eine Routine *printPraefix(p,n)* welche alle Schlüssel des Tries ausgibt, die den Präfix "ersten n Bits von p" haben.

Überlegen Sie sich sinnvolle Testfälle für die Routinen und implementieren Sie diese.