Universität des Saarlandes
FR 6.2 Informatik
Prof. Dr. Hans-Peter Lenhof
Dr. Oliver Kohlbacher
Dipl. Inform. Andreas Hildebrandt

10. Übung zur Vorlesung Programmierung II, SS 2003

Hinweis: Bei der Definition der *Heapeigenschaft* kann man sich aussuchen, ob der Schlüsselwert in den Elternknoten größer oder gleich bzw. kleiner oder gleich dem in den Kindern gespeicherten ist. Auf diesem Übungsblatt gehen wir davon aus, dass immer die *kleineren oder gleichen* Werte in den Eltern gespeichert werden, so dass das Minimum in der Wurzel steht.

Aufgabe 1: Amortisierte Analyse (10 Punkte)
Definieren Sie analog zu den 2–4 - Bäumen aus Blatt 9 den 2–5 - Baum. Beschreiben Sie die Rebalancierungsoperationen *Spalten*, *Verschmelzen* und *Stehlen* für 2–5 - Bäume und zeigen Sie die folgende Behauptung: Eine Sequenz von n `insert` oder `delete` in beliebiger Reihenfolge auf einem anfangs leeren 2–5 - Baum führt zu höchstens 2n Rebalancierungsoperationen.

Aufgabe 2: d-näre Heaps (10 Punkte)
Analog zum binären Heap, der Ihnen aus der Vorlesung bekannt ist, kann man einen sogenannten d-nären Heap definieren. Dieser unterscheidet sich vom binären Heap nur dadurch, dass jeder Knoten d oder 0 Kinder hat.

(a) Beschreiben Sie die Repräsentation eines d-nären Heaps in einem Array.

(b) Finden Sie eine Funktion $f_d(n)$ so dass für die Höhe $h_d(n)$ eines d-nären Heaps mit n Elementen gilt: $h_d(n) \in \Theta(f_d(n))$.

(c) Beschreiben Sie eine effiziente Implementierung von `insert(const Item& i);` und analysieren Sie deren Laufzeit in Abhängigkeit von d und n.

Aufgabe 3: Binärheaps (10 Punkte)

(a) Gegeben Sei das Array $A := [5, 3, 17, 10, 84, 19, 6, 22, 9]$. Handelt es sich bei A um einen binären Heap? Begründen Sie Ihre Aussage! Sollten Sie feststellen, dass A *keinen* binären Heap bildet, stellen Sie graphisch dar, wie die in der Vorlesung definierte Funktion `heapify` A in einen Heap verwandelt.

(b) Gegeben Sei nun das Array $B := [15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1]$. Stellen Sie graphisch dar welche Operationen ausgeführt werden, wenn Sie `delete_min` für diesen Heap aufrufen.

Aufgabe 4: Binärheaps, Implementierung (10 Punkte)
Implementieren Sie die in der Vorlesung vorgestellte Template-Klasse `BinaryHeap <typename Key, typename Inf>`.

Zusatzaufgabe 5: Prioritätswarteschlangen, Implementierung (10 Bonuspunkte)
Implementieren Sie die in der Vorlesung vorgestellte Template-Klasse `PriorityQueue <typename Key, typename Inf>`. Verwenden Sie dazu Ihre `BinaryHeap<Key, Inf>` – Implementierung aus Aufgabe 4.

Abgabe: 05.07.2003