GOETHE UNIVERSITÄT FRANKFURT AM MAIN

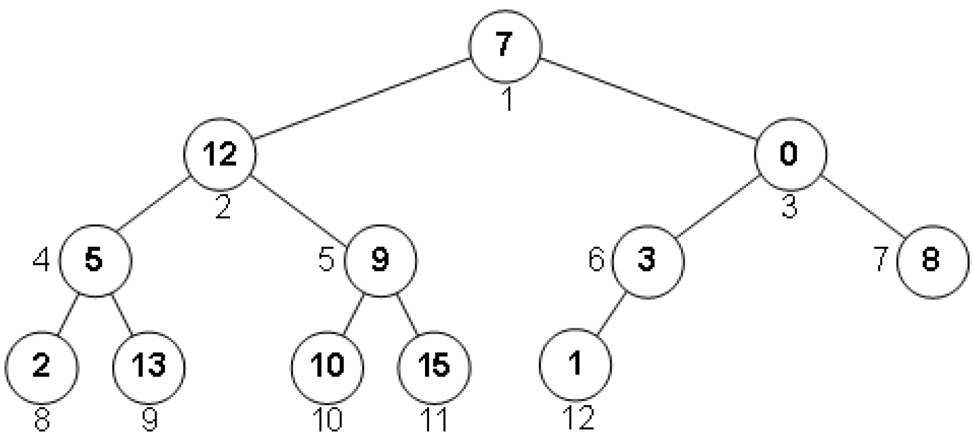
Computerorientierte Mathematik
WS 2017/18
Prof. Dr. Thomas Gerstner

Miniprojekt 2

Abgabe bis Freitag, 19.1.2018, 9.45 Uhr

Sortieren

Mit dem Heapsort-Algorithmus lässt sich eine Liste L von Elementen $L[i], i = 1, \dots, n$ sortieren, vorausgesetzt es existiert eine Ordnungsrelation $<$ zwischen den Listenelementen. Der Algorithmus ordnet die Listenelemente gedanklich in einer Binärbaumstruktur an (siehe Beispielgrafik für die Liste $[7, 12, 0, 5, 9, 3, 8, 2, 13, 10, 15, 1]$).



Algorithmus:

Gegeben sei eine Liste L mit n Elementen.

(1) Bringe die Liste in die Heap-Struktur:

Führe den folgenden Schritt (a) der Reihe nach für die Listenelemente $L[i], i = n, n-1, \dots, 2$ aus.

(a) Vertausche $p = L[i]$ so lange mit seinem jeweiligen Vaterelement im Baum bis der Vater größer als p ist, oder p an der Wurzel steht.

(2) Arbeite den Heap ab:

Führe die folgenden Schritte (a) und (b) jeweils der Reihe nach für Indizes $i = n, \dots, 2$ aus:

(a) Vertausche Listenelement $L[i]$ mit $L[1]$.

(b) Vertausche das erste Listenelement $q = L[1]$ so lange mit dem größeren der beiden Sohnelemente bis es größer als beide Söhne ist, oder kein Sohn mit Index $> i - 1$ mehr vorhanden ist.

(3) Gebe die sortierte Liste L aus.

Hinweise:

- Die Vertauschungsoperationen der Listenelemente können direkt in der Liste ausgeführt werden, da zwischen Liste und Baum folgender Zusammenhang besteht:

  Steht der Sohn an Stelle i in der Liste, so ist der zugehörige Vater an Stelle $\lfloor i/2 \rfloor$ zu finden.

  Steht der Vater an Stelle i in der Liste, so sind die Söhne an den Stellen $2i$ und $2i+1$ zu finden.

- Achten sie darauf, dass in Sage das erste Listenelement von L mit $L[0]$ angesprochen wird, der Algorithmus aber bei $L[1]$ beginnt.

Aufgaben:

(a) Schreiben sie eine Funktion *Heapaufbau(L)* mit einer Liste L als Input, welche diese Liste auf die Heapstruktur bringt (siehe Schritt 1 im Algorithmus) und geben sie die geänderte Liste wieder aus.

(b) Schreiben sie eine Funktion *Heapabbau(L)*, die für eine Liste L, welche die Heapstruktur besitzt, den Heap abarbeitet (siehe Schritt 2 im Algorithmus) und geben sie die geänderte Liste wieder aus.

(c) Schreiben sie eine Funktion *Heapsort(L)*, die eine Liste L mit Hilfe des Heapsort-Algorithmus der Größe nach sortiert und danach ausgibt.
Testen sie ihre Funktion für die Liste $L = [7, 12, 0, 5, 9, 3, 8, 2, 13, 10, 15, 1]$.

(d) Ergänzen sie ihre Funktionen *Heapaufbau(L)* und *Heapabbau(L)* so, dass sie mitzählen, wie viele Vertauschungen von Listenelementen stattfinden und geben sie in ihrer Funktion *Heapalgorithmus(L)* die Summe aller Vertauschungen aus.
Wie viele Vertauschungen benötigt das Sortieren der obigen Liste?

(e) Bestimmen sie durch eine Monte-Carlo Simulation den durchschnittlichen und maximalen Aufwand bei der Sortierung einer n-elementigen Liste.
Schreiben sie dafür eine Funktion $HeapAufwand(n)$, welche 10.000 Mal eine zufällige n-elementige Liste L mit Hilfe des Befehls *random()* erstellt, diese Liste sortiert und am Ende den durchschnittlichen und maximalen Sortieraufwand $Durchschnittsaufwand$ und $MaximalerAufwand$ der Simulationen ausgibt.
Plotten sie die Punkte $(n, Durchschnittsaufwand(n))$ und $(n, MaximalerAufwand(n))$ für $n = 4, \ldots, 9$ und fügen sie die Funktion $f(n) = n * \log(n)$ der Grafik hinzu. Was vermuten sie über die Größenordnung des durchschnittlichen und maximalen Sortieraufwandes in Abhängigkeit von n?

(f) Schreiben sie ein LaTeX-Dokument mit Titelseite und einer Dokumentation ihrer Ergebnisse, sowie dem vollständigen Lösungsweg der einzelnen Teilaufgaben.

Gesamtpunktzahl: 6/8/2/3/5/11 = 35 Punkte