Universität des Saarlandes
FR 6.2 Informatik
Prof. Dr. Hans-Peter Lenhof
Dipl. Inform. Jan Küntzer

Programmierung II, SS 2004
Musterlösung zur 11. Übung

Aufgabe 1: Building Heap (10 Punkte)

i) Nein! Bauen wir einen Max-Heap aus $A := [4, 1, 3, 2, 16, 9, 10, 14, 8, 7]$. Wie man in Abb.(a) sieht sind die beiden erzeugten Bäume unterschiedlich.

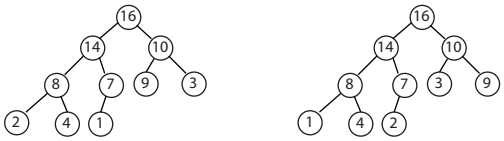

Abbildung 1: Der linke Max-Heap wurde mit BUILD-HEAP erzeugt. Der Rechte mit BUILD-HEAP'.

ii) Betrachten wir den worst-case: Beim Einfügen eines Elements mit HEAP-INSERT(A,A[i]) müssen wir im schlimmsten Fallvom Blatt bis zur Wurzel laufen. Damit erhalten wir eine Laufzeit von $\Theta(\log(k))$, wobei $\log(k)$ die Höhe des Heap ist ($k \leq n$). Dies wird $n-1$-mal aufgerufen um einen n-elementigen Heap zu bauen. Also gilt für die Laufzeit: $T(n) = T(n-1) + \log(n)$. Damit benötigt BUILD-HEAP' $\Theta(n \log(n))$ Zeit in worst case um einen n-elementigen Heap zu bauen.

Aufgabe 2: Binärheaps (10 Punkte)

i) Bei Interpretation als min-Heap lautet das Ergebnis [3,5,6,11,8,17,14,32,45,5].

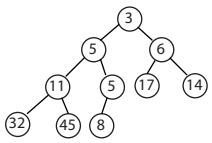

Abbildung 2: Heap von Aufgabe 2 (i)

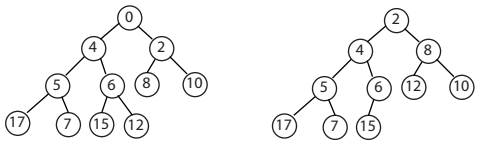

Abbildung 3: Der Heap vor und nach dem Löschen.

ii) Siehe Abb.(b)

Aufgabe 3: Gerichtete Graphen (10 Punkte)

i) Gegeben sei eine Adjazenzmatrix. Damit kann man in konstanter Zeit überprüfen, ob eine gegebene Kante existiert. Um zu überprüfen, ob eine Kante $(u, w) \in G^2$ existiert, müssen wir für alle möglichen Kanten v in $\mathcal{O}(1)$ überprüfen, ob (u, v) und (v, w) existiert. Da wir höchstens n Kanten zu überprüfen haben und n^2 Kantenpaare anschauen müssen, benötigen wir $\mathcal{O}(n^3)$ Zeit.

ii) Gegeben sei eine Adjazenzliste. Damit haben wir eine Liste mit allen Kanten im Graphen. Für eine Kante (u, v) laufen wir über alle Kanten von v in $\mathcal{O}(n)$ Zeit und füllen damit die Adjazenzmatrix von G^2, die am Anfang leer ist. Es braucht $\mathcal{O}(mn)$ um die Kanten zu konstruieren und $\mathcal{O}(n^2)$ zum Initialisieren und Einlesen der Adjazenz Matrix von G^2, also insgesamt $\mathcal{O}((n + m)n)$. Da $n \leq m$ solange der Graph nicht-zusammenhängend, kann man dies vereinfachen zu $\mathcal{O}(mn)$ und ist damit auf dünn besetzten Graphen.

Anmerkung: Weil das Quadrat der Adjazenzmatrix die Adjazenzmatrix des Quadrates ist, nennt man dies das Quadrat des Graphen. Damit erhält man einen noch schnellen Algorithmus!

Aufgabe 4: Euler-Circuit (10 Punkte)

i) Es kann nur einen Euler-Circuit geben, wenn es keinen Knoten mit ungeraden Anzahl an Kanten gibt. Also suchen wir alle Knoten mit ungerader Anzahl und duplizieren eine seiner Kanten. Vorzugsweise wählen wir dabei eine Kante die mit zwei Knoten mit ungeraden Kanten verbunden ist. Sollte es keine solche geben, haben wir damit einen neuen Knoten mit ungeraden Kantenzahl erzeugt. Daher muessen wir solange rekursiv vorgehen, bis wir mit dem Einfügen einer Kante bei zwei Knoten die Anzahl der Kanten gerade gemacht haben.

ii) Siehe Abb.(b).

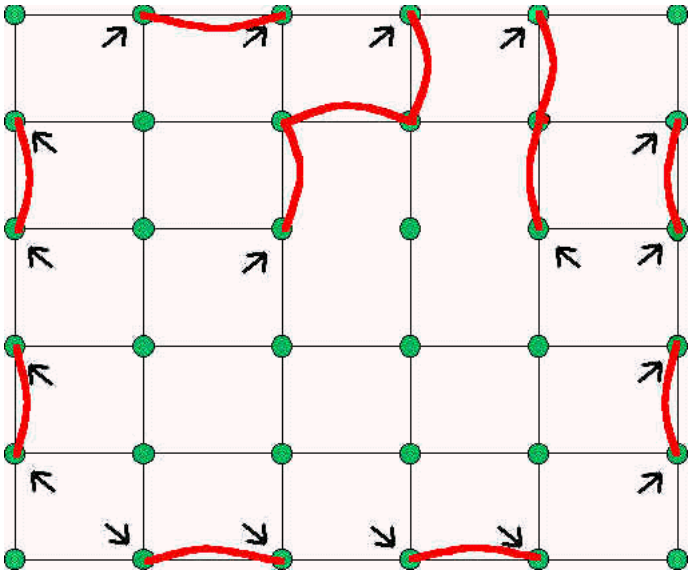

Abbildung 4: Eulerisierter Graph von Aufgabe 4 (iii)