Technische Universität München
Institut für Informatik
Prof. Dr. Angelika Steger

SS 2000
Übungsblatt 10
7. Juli 2000

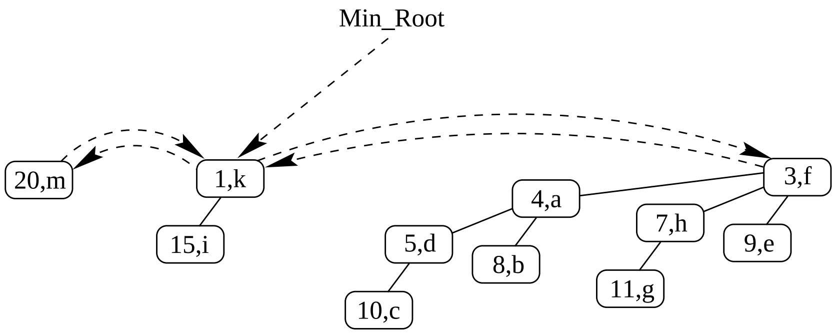
Einführung in die Informatik IV

Aufgabensteller: Klaus Holzapfel
Abgabetermin: Mittwoch, 19. Juli 2000, 9:25 Uhr

Tutoraufgabe 1

Gegeben sei der folgende Binomial Heap, der durch das Einfügen der Elemente in den leeren Heap entstanden ist. Hierbei steht an erster Stelle der *key* und an zweiter Stelle das Element.

```
 (4,a)  (8,b)  (10,c)  (5,d)  (9,e)  (3,f)  (11,g)  (7,h)
(15,i)  (1,k)  (20,m)
```

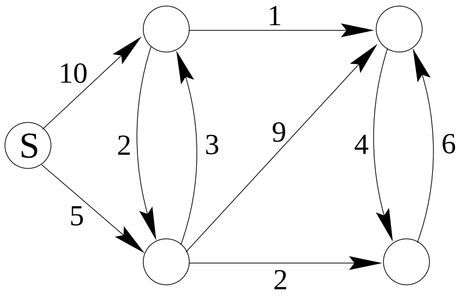


Wenden Sie darauf die nachfolgenden Operationen an und zeichnen sie den resultierenden Heap.

```
insert (12,n)
delete_min()
decrease_key(a,2)
```

Tutoraufgabe 2

Bestimmen sie mit dem Algorithmus von Dijkstra für den folgenden Graphen die Länge der kürzesten Verbindungen vom Knoten `s` zu allen anderen Knoten.



Konstruieren Sie einen Graphen mit ganzzahligen (nicht notwendigerweise positiven) Kantengewichten, für den der Algorithmus von Dijkstra ein falsches Ergebnis liefert. Diskutieren Sie ein Verfahren, mit dem man für beliebige ganzzahlige Kantengewichte das *Single-Source Shortest Path*-Problem lösen kann. Der Fall von Kreisen mit negativer Kantensumme soll dabei erkannt werden.

Hausaufgabe 1 (freiwillig)

Beschreiben Sie, wie in einem Binomial Heap die DELETE-Operation durchgeführt werden kann. Bestimmen Sie dabei auch die Komplexität dieser Operation.

Hausaufgabe 2 (freiwillig)

Gegeben sei ein gerichteter Graph $G = (V, E)$ eines Kommunikationsnetzes, wobei jeder Kante $(u, v) \in E$ eine unabhängige Ausfallwahrscheinlichkeit $p(u, v)$ (mit $0 \leq p(u, v) \leq 1$) zugeordnet ist. Es kann davon ausgegangen werden, dass diese Wahrscheinlichkeiten paarweise unabhängig sind.
Geben Sie einen effizienten Algorithmus an, um für einen gegebenen Knoten die zuverlässigsten Verbindungen zu allen anderen Konten zu ermitteln. Eine zuverlässigste Verbindung zwischen zwei Knoten ist ein Pfad, für den die Wahrscheinlichkeit am größten ist, dass keine Kante darauf ausfällt.

Hausaufgabe 3 (1+1+2+1 = 5 Punkte)

Zeigen Sie folgenden Eigenschaften für Binomialbäume:

a) Der Binomialbaum B_k hat Höhe k.

b) Der Binomialbaum B_k enthält 2^k Knoten.

c) Im Binomialbaum B_k gibt es genau $\binom{k}{i}$ Knoten auf Ebene i ($0 \leq i \leq k$, wobei Ebene 0 der Wurzel entspricht und $\binom{k}{0} = 1$).

d) Im Binomialbaum B_k hat die Wurzel Grad k und alle anderen Knoten haben kleineren Grad.

Hausaufgabe 4 (5 Punkte)

Implementieren Sie in Java die aus der Vorlesung bekannte *Union-Find*-Struktur. Es sind die in der Vorlesung festgelegten Konventionen einzuhalten:

a) Die Operation `union(a,b)` vereinigt die Mengen, die durch `a` und `b` repräsentiert werden, und weist der neuen Menge `a` als Repräsentanten zu.

b) Die Operation `find(x)` liefert den Repräsentanten der Menge, in der `x` enthalten ist.

c) Die Initialisierung erfolgt derart, dass sich zu Beginn alle Elemente in einer einelementigen Menge befinden und gleichzeitig Repräsentant dieser Menge sind.

Geben Sie das Programm unter dem Namen `UnionFind` an SOLEA ab.

Format der Ein/Ausgabe: Die Eingabe besteht aus einer Zahl n, die die Elemente der Struktur definiert (z.B. 8 steht für die Elemente $1, 2, 3, \ldots, 7, 8$). Danach folgt eine Reihe von Funktionsaufrufen, wobei `union(i,j)` bzw. `find(i)` durch `u i j` bzw. `f i` abgekürzt wird. Für jede `find`-Operation ist der jeweilige Repräsentant in einer eigenen Zeile auf die Standardausgabe zu schreiben.

Beispiel: Für die Eingabe

```
8
u 2 6
f 8
u 7 4
f 7
u 3 7
f 7
u 2 3
f 7
u 5 1
f 5
```

soll die Ausgabe

```
8
7
3
2
5
```

erzeugt werden.