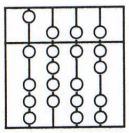
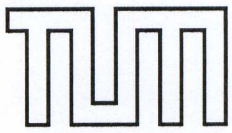


TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK



Lehrstuhl für Effiziente Algorithmen
Grundlagen: Algorithmen und Datenstrukturen
Prof. Dr. Christian Scheideler, Dr. Stefan Schmid

SS 2008
Endterm-Klausur
19.07.08

Endterm-Klausur zu Grundlagen: Algorithmen und Datenstrukturen

Name	Vorname	Studiengang	Matrikelnummer
Schmid	Stefan		
Hörsaal	Reihe	Sitzplatz	Unterschrift

Allgemeine Hinweise:

- **Bitte füllen Sie die oben angegeben Felder vollständig aus und unterschreiben Sie!**
- **Schreiben Sie nicht mit Bleistift oder in roter/grüner Farbe!**
- **Lösen Sie die Aufgaben auf dem freien Platz unter der Angabe (bzw. auf der Rückseite). Fordern Sie weiteres Papier von der Klausuraufsicht, falls Ihnen der Platz nicht ausreicht.**
- **Als Hilfsmittel ist ausschließlich ein handbeschriebenes DinA4-Blatt zugelassen. Bei Missachtung wird die gesamte Klausur mit null Punkten bewertet.**
- **Die Arbeitszeit beträgt 120 Minuten.**
- **Prüfen Sie, ob Sie alle 7 Seiten erhalten haben.**

! Lösungskizzen: Keine Garantie, andere Lösungen zum Teil auch korrekt (eg. bei Bäumen)!

Aufgabe 1 [5 Punkte] **Multiple Choice Aufgabe**

Kreuzen Sie pro Teilaufgabe höchstens ein Kästchen an. Für ein falsches Kreuz gibt es einen halben Minuspunkt, für ein richtiges einen halben Pluspunkt. Wenn Sie kein Kreuz setzen, bekommen Sie auch keine Punkte. Eine negative Gesamtpunktzahl dieser Aufgabe wird zu 0 aufgerundet. Maximieren Sie Ihre Punktzahl!

a) Selection Sort hat eine Laufzeit von $O(n^2)$. ☒ Richtig ☐ Falsch

b) Der Quicksort Algorithmus, in dem als Pivotelement immer das erste Element einer Folge gewählt wird, hat eine Laufzeit von $O(n \log n)$. ☐ Richtig ☒ Falsch

c) Jeder vergleichsbasierte Sortieralgorithmus braucht mindestens $\Omega(n \log n)$ Schritte, um eine Folge von n Elementen zu sortieren. ☒ Richtig ☐ Falsch

d) Ein binärer Heap kann aus einer unsortierten Liste von n Elementen in $O(n)$ Zeit aufgebaut werden. ☒ Richtig ☐ Falsch

e) Ein Binomialheap ist eine Liste unterschiedlicher Binomialbäume. ☒ Richtig ☐ Falsch

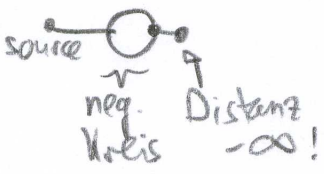
f) In einem (a, b)-Baum muss jeder Knoten Grad x haben wobei $a \leq x \leq b$. ☐ Richtig ☒ Falsch Wurzel!

g) In jedem Graphen, der als Adjazenzmatrix dargestellt ist, können die find(i, j), insert(e) und remove(i, j) Operationen in worst case konstanter Zeit durchgeführt werden. ☒ Richtig ☐ Falsch

h) Wenn wir die Knoten eines DAGs absteigend nach der finishTime eines DFS-Durchlaufs sortieren, dann erhalten wir eine topologische Sortierung. ☒ Richtig ☐ Falsch

i) Dijkstras Algorithmus kann kürzeste Wege für beliebige Graphen berechnen. ☐ Richtig ☒ Falsch

j) Ein Knoten kann nur dann eine Distanz von $-\infty$ zu einer Quelle s haben, wenn er auf einem negativen Kreis liegt. ☐ Richtig ☒ Falsch



Aufgabe 2 [8 Punkte] **Heaps**

a) [4 Punkte] Geben Sie für die folgende Folge von Zahlen eine Baumrepräsentation *plus* eine Arrayrepräsentation eines *binären* Heaps an (unter Verwendung von n Insert-Operationen!). Es reicht, das Endergebnis anzugeben (keine Zwischenschritte).

$$< 0, 18, 17, 21, 5, 1, 2, 15, 16, 12, 9, 14, 13, 23 >$$

b) [4 Punkte] Wenden Sie die merge Operation auf die folgenden *Binomialheaps* an (nur Endergebnis).

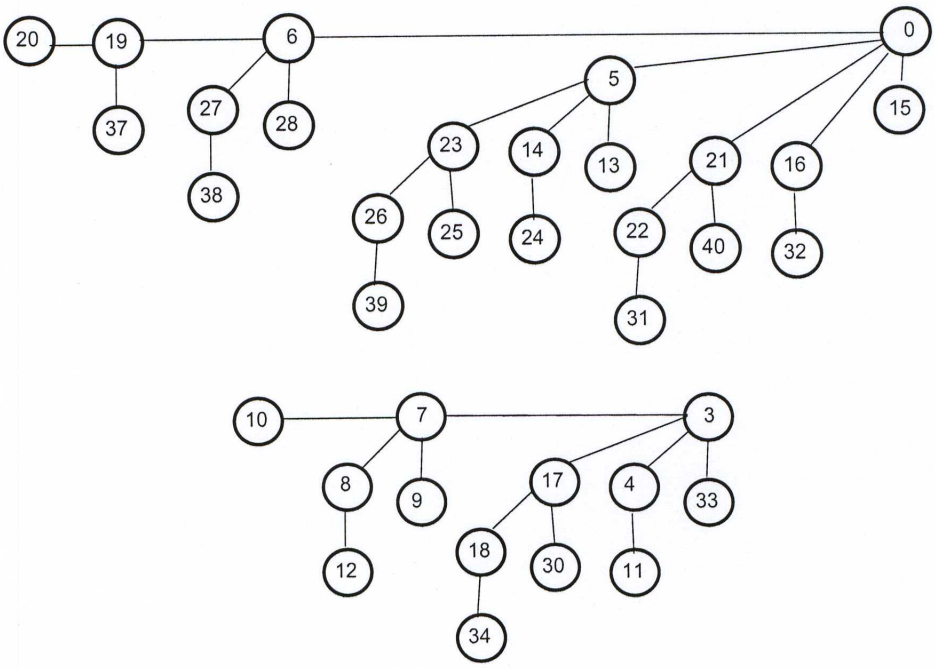
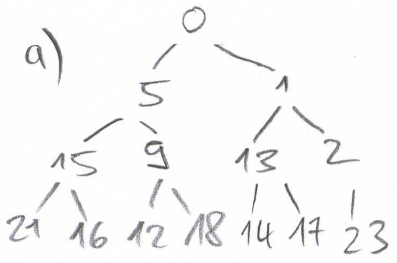
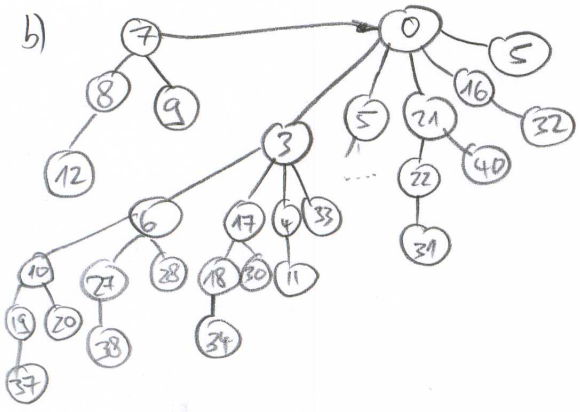


Abbildung 1: Zwei Binomialheaps.



$\Rightarrow$ 0, 5, 1, 15, 9, 13, 2, 21, 16, 12, 18, 14, 17, 23



Aufgabe 3 [10 Punkte] **Suchbäume**

a) [2 Punkte] Geben Sie *irgendeinen* binären Suchbaum für die folgende Elementmenge an (nur Endergebnis, aber mit ∞-Element und Liste unten):

$$\{16, 15, 14, 13, 30, 29, 18, 17, 23, 5, 19\}$$

b) [4 Punkte] Geben Sie für einen anfangs leeren Binärbaum die binären Suchbäume (wie in der Vorlesung behandelt mit ∞-Element) aus, die sich aus folgenden Operationen ergeben (Bild von Baum nach *jeder* einzelnen Operation zeichnen!):

$$insert(2), insert(1), insert(4), insert(7), insert(5),$$

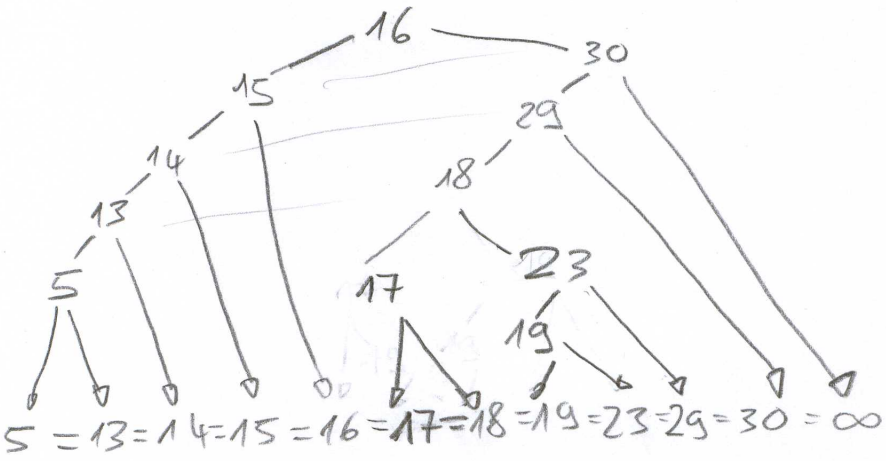
$$insert(8), insert(3), remove(4), insert(6), remove(2).$$

c) [4 Punkte] Geben Sie für einen anfangs leeren $(2,4)$-Baum die $(2,4)$-Bäume für die folgenden Operationen (wie in der Vorlesung behandelt mit ∞-Element) aus (nach *jeder* Operation Baum angeben!).

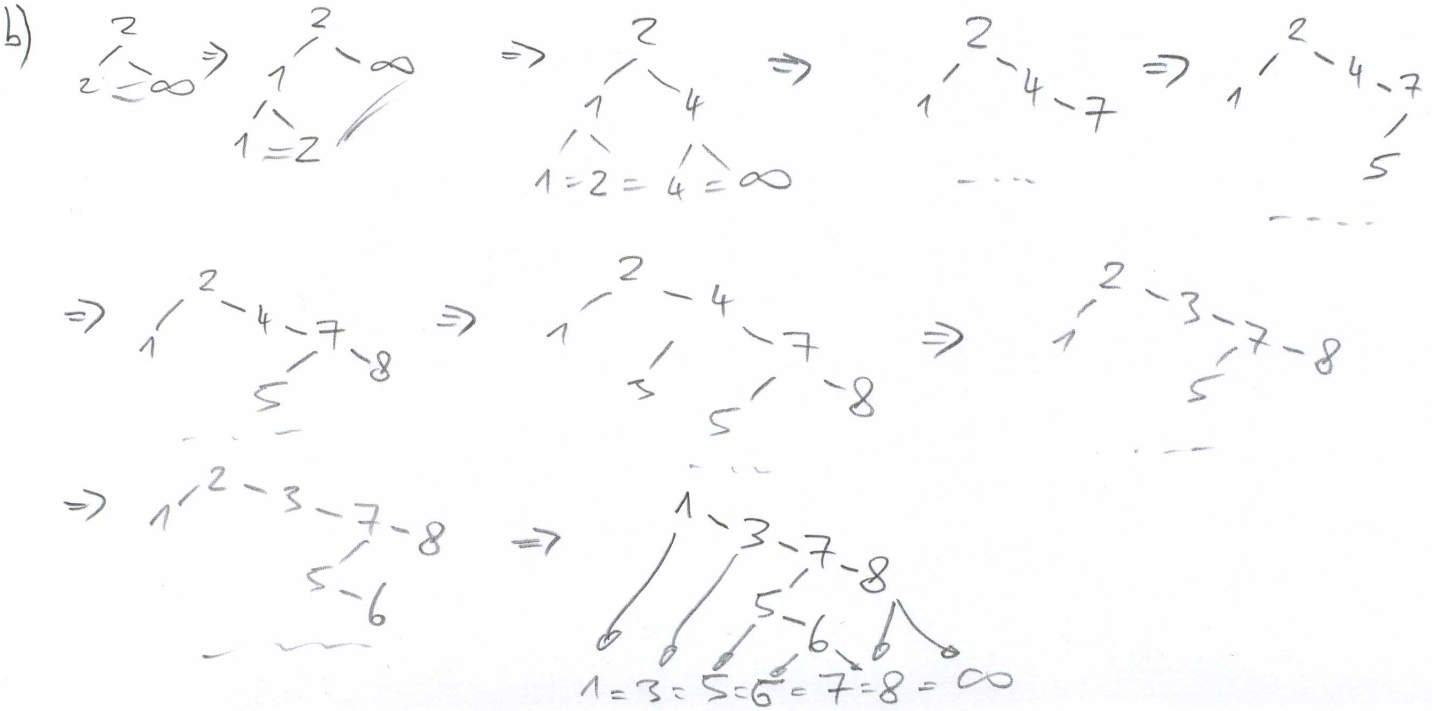
$$insert(3), insert(9), insert(23), insert(5), insert(4), insert(1),$$

$$insert(7), insert(6), insert(8), remove(4), remove(3), remove(1).$$

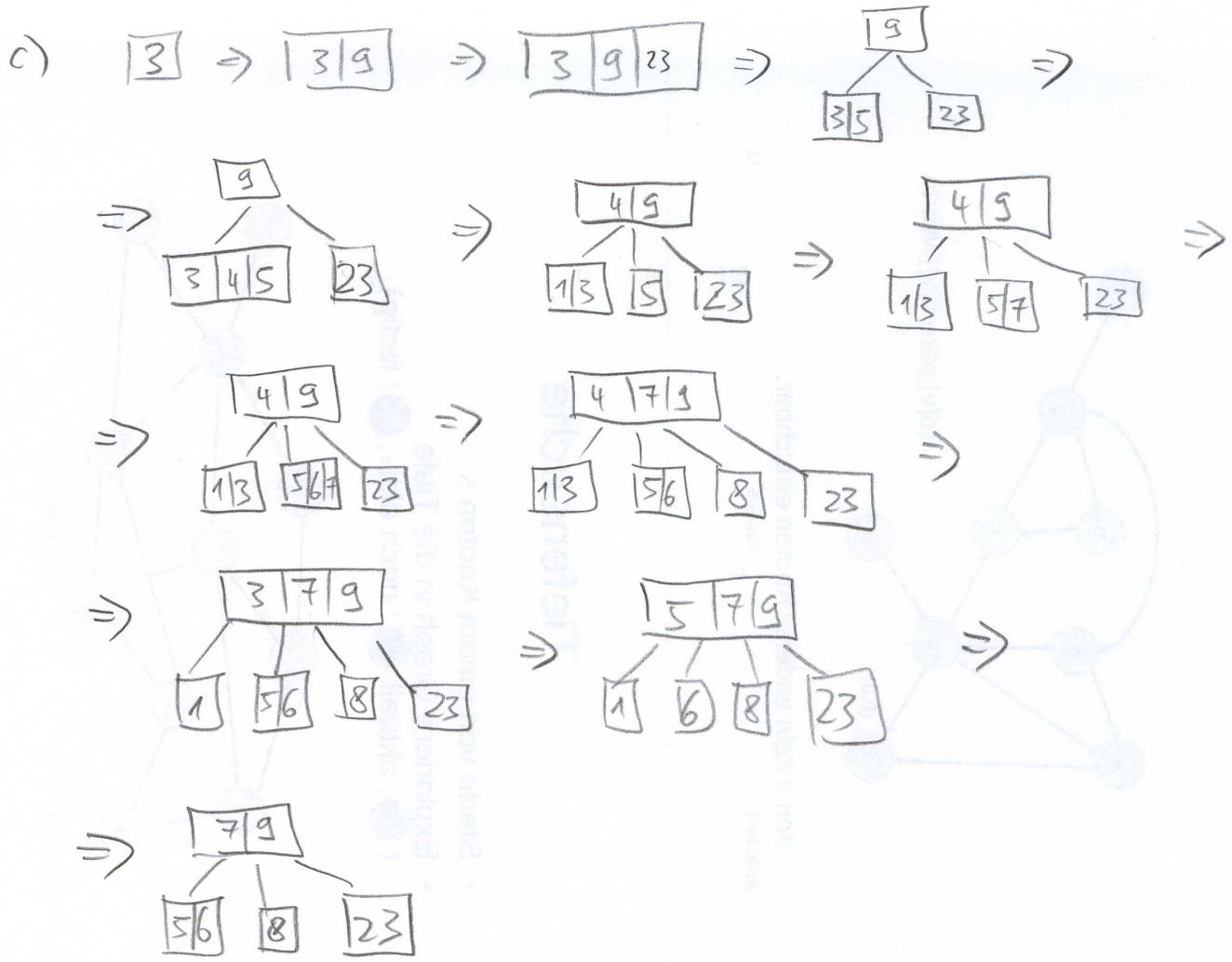
a)



b)



c)



Aufgabe 4 [6 Punkte] **Graphen**

a) [2 Punkte] Geben Sie die Adjazenzmatrix von folgendem Graph G an.

	1	2	3	4	5
1	1	0	1	0	0
2	0	1	0	0	1
3	0	1	0	0	1
4	0	0	1	0	0
5	0	0	1	1	1

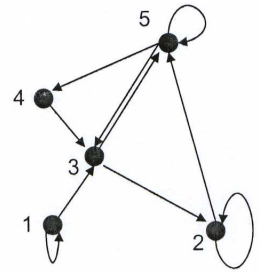
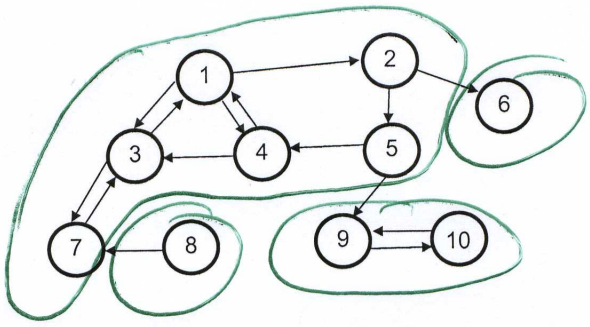


Abbildung 2: Graph G.

b) [2 Punkte] Zeichnen Sie die starken Zusammenhangskomponenten in H ein (nur Endergebnis).



4 Komponenten

Abbildung 3: Graph H.

b) [2 Punkte] Geben Sie eine topologische Sortierung für den DAG I an (direkt in Knoten schreiben).

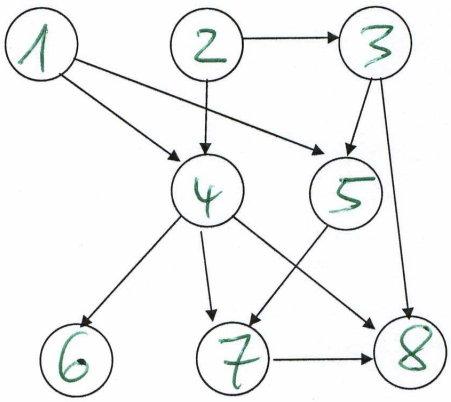


Abbildung 4: DAG I.

Aufgabe 5 [8 Punkte] **Kürzeste Wege und minimale Spannbäume**

Betrachten Sie den folgenden ungerichteten Graphen G.

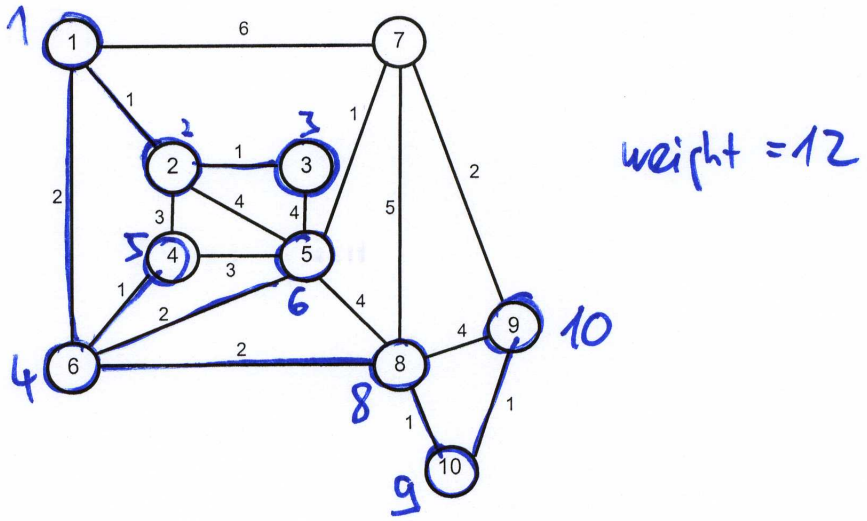
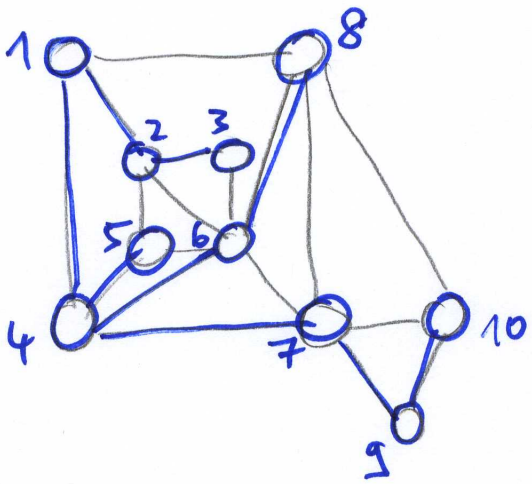


Abbildung 5: Graph G.

a) [4 Punkte] Berechnen Sie mithilfe des Jarnik-Prim Algorithmus den minimalen Spannbaum des Graphen G ausgehend vom Knoten 1. Markieren Sie dazu durch Zahlen an den Knoten (direkt oben im Bild!), in welcher Reihenfolge die Knoten vom Jarnik-Prim Algorithmus aus der Priority Queue entnommen werden, und zeichnen Sie den resultierenden minimalen Spannbaum.

b) [4 Punkte] Berechnen Sie mittels Dijktras Algorithmus die kürzesten Wege vom Knoten 1 zu jedem anderen Knoten im Graphen G. Protokollieren Sie den Ablauf ausführlich (z.B. mittels Tabelle, oder markieren Sie durch Zahlen an den Knoten, in welcher Reihenfolge die Knoten vom Dijkstra Algorithmus aus der Priority Queue entnommen werden). Zeichnen Sie auch den resultierenden Baum der kürzesten Wege.



Aufgabe 6 [4 Punkte] **Bonusaufgabe**

Seien die Kosten eines Weges p in einem Graphen G definiert als das Maximum aller Kantenkosten in p. Schlagen Sie einen Algorithmus vor, mit dem man in $O((n+m)\log(n))$ Zeit für einen Knoten s in einem gerichteten Graphen $G = (V, E)$ mit $|V| = n$ und $|E| = m$ und beliebigen Kantenkosten $c : E \to \mathbb{R}$ die minimalen Wegekosten von s zu allen anderen Knoten in G berechnen kann.

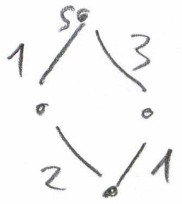
Analog zu Dijkstra, aber statt

$$d(w) = d(v) + c(v,w)$$

nehme

$$d(w) = \max(d(v), c(v,w)) \; !$$

Beispiel dass das was anderes als MST ist:



MST:

weight = 5

KWB:

weight = 6