Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Tobias Lieber

Sommersemester 2011
Wiederholungsklausur
28. April 2011

Effiziente Algorithmen und Datenstrukturen I

Name	Vorname	Studiengang	Matrikelnummer
.....	.....	☐ Diplom ☐ Inform. ☐ Bachelor ☐ Mathe. ☐ Master ☐ Bio-Inf.	.....

Hörsaal	Reihe	Sitzplatz	Unterschrift
.....	.....	.....	.....

Allgemeine Hinweise zur Klausur

- Bitte füllen Sie obige Felder in Druckbuchstaben aus und unterschreiben Sie!
- Schreiben Sie nicht in *roter* oder *grüner* Farbe bzw. mit Bleistift!
- Außer Ihrem Schreibgerät und einem handbeschriebenen DIN-A4-Blatt sind keine weiteren Hilfsmittel erlaubt.
- Die Bearbeitungszeit beträgt 180 Minuten.
- Alle Antworten sind in die geheftete Angabe auf den jeweiligen Seiten (bzw. Rückseiten) der betreffenden Aufgaben einzutragen. Begründen Sie alle Ihre Schritte. Auf dem Schmierblatt können Sie Nebenrechnungen machen. Der Schmierblattbogen muss ebenfalls abgegeben werden, wird aber in der Regel nicht bewertet.

Hörsaal verlassen von ..... bis ..... / von ..... bis .....

Vorzeitig abgegeben um .....

Besondere Bemerkungen:

	A1	A2	A3	A4	A5	A6	A7	A8	Σ	Korrektor
Erstkorrektur										
Zweitkorrektur										

Aufgabe 1 (8 Punkte)

Beantworten Sie die folgenden Fragen jeweils mit einem Satz:

a) Sei $A \in \{0,1\}^{m \times n}$ eine Matrix, so dass jeder Eintrag der Matrix nur dann ein Null-Eintrag sein kann, wenn er an der ersten Position der Zeile steht oder der vorhergehende Eintrag in der Zeile auch Null ist. Beschreiben Sie einen möglichst effizienten Algorithmus und dessen Laufzeit, der die Anzahl der Nichtnull-Einträge ausgibt.

Binäre Suche auf allen Zeilen. Also $\mathcal{O}(m \log n)$ Zeit

b) Geben Sie mittels eines treffenden Begriffs eine Möglichkeit an, um untere Laufzeitschranken für Probleme zu finden.

Gegenspieler-Argument.

c) Wozu benützt man die Stirlingsche Formel bei der Bestimmung einer unteren Schranke für die Laufzeit von vergleichsbasierten Sortieralgorithmen?

Zur Bestimmung der Baumhöhe eines Vergleichsbaums

d) Mit welcher geringen strukturellen Modifikation kann Quicksort in ein Sortierverfahren umgewandelt werden, das deterministisch im worst-case in $\mathcal{O}(n \log n)$ Schritten sortiert?

Durch Benutzung eines $\mathcal{O}(n)$ Median-Algorithmus zur Bestimmung des Pivot-Elements.

e) Welche Laufzeitgarantien liefert die Amortisierte Analyse der Splay-Trees im Vergleich zu beliebigen anderen Suchbäumen?

Sie sind, wenn genügend Operationen ausgeführt werden, nur bis auf einen konstanten Faktor schlechter.

f) Geben Sie eine möglichst optimale Laufzeitschranke für DFS und BFS auf einem Graphen $G = (V, E)$ an. Erklären Sie auch kurz, warum Ihre angegebene Schranke für die

Algorithmen optimal ist.

Weil jede Kante nur konstant oft besucht wird.

g) Warum ist der Heiratssatz von Hall im Algorithmen-Entwurf für Matching-Probleme nur von eingeschränktem Interesse?

Es gibt $\Omega(2^n)$ Teilmengen.

h) Nennen (oder beschreiben) Sie einen Algorithmus, der überprüft, ob ein Matching M maximale Kardinalität hat. Geben Sie auch dessen Laufzeit an.

Wir können alle Matchings aufzählen und schauen, ob es ein größeres gibt. Die Laufzeit ist dabei in $\mathcal{O}(2^m)$.

Eleganter geht es mit mit Vaziranis alternierenden Pfaden und Blossoms in $\mathcal{O}(\sqrt{n}m)$.

Aufgabe 2 (9 Punkte)

a) Lösen Sie die Rekursion durch Transformation des Wertebereichs und der Methode des charakteristischen Polynoms:

$$a_n = \sqrt{a_{n-1}a_{n-2}} \quad \text{mit } a_0 = 1 \text{ und } a_1 = 2$$

b) Lösen Sie die Rekursion durch eine geeignete Substitution:

$$b_n = \sqrt{1 + b_{n-1}^2} \quad \text{mit } b_0 = 0$$

c) Lösen Sie die folgende Rekursion mit einer Erzeugendenfunktion:

$$c_n = 3c_{n-1} + c_{n-2} - 3c_{n-3} \quad \text{für } n \geq 3$$

mit den Anfangsbedingungen: $c_0 = 9, c_1 = 13$ und $c_2 = 33$.

Lösungsvorschlag:

a) Wir setzen $d_n := \log_2 a_n$. Damit folgt:

$$d_n = \frac{1}{2}d_{n-1} + \frac{1}{2}d_{n-2}$$

Daher ergibt sich das charakteristische Polynom zu $x^2 - \frac{1}{2}x - \frac{1}{2} = 0$ mit den Nullstellen: $x_1 = -\frac{1}{2}$ und $x_2 = 1$. Nach dem Einsetzen der Startbedingungen $d_0 = 0$ und $d_1 = 1$ erhalten wir für d_n die geschlossene Form: $-\frac{2}{3}\left(-\frac{1}{2}\right)^n + \frac{2}{3}$.

Damit ergibt sich für $a_n = 2^{-\frac{2}{3}\left(-\frac{1}{2}\right)^n + \frac{2}{3}}$.

b) Wir quadrieren und erhalten $b_n^2 = 1 + b_{n-1}^2$. Durch Ersetzen von b_n^2 mit f_n erhalten wir: $f_n = 1 + f_{n-1}$. Mit der Startbedingung $f_0 = 0$ ergibt sich: $f_n = n$. Damit folgt direkt $b_n = \sqrt{n}$.

c) Durch Aufsummieren erhalten wir

$$C(z) = \frac{9 - 14z - 15z^2}{1 - 3z - z^2 + 3z^2} = \frac{9 - 14z - 15z^2}{(1 - 3z)(1 + z)(1 - z)}$$

mit $C(z) = \sum_{n=0}^{\infty} c_n z^n$.

Somit ergibt sich nach der Partialbruchzerlegung:

$$C(z) = \frac{3}{1 - 3z} + \frac{1}{1 + z} + \frac{5}{1 - z}$$

Damit erhalten wir für $c_n = 3^{n+1} + (-1)^n + 5$.

Aufgabe 3 (5 Punkte)

Sei $c \in \mathbb{N}$ eine beliebige aber feste Konstante mit $c \geq 2$. In einem Array $A[0 \dots \ell - 1]$ der Größe ℓ seien $k \leq \ell$ Elemente gespeichert. Fügen wir nun ein neues Element ein, so gehen wir wie folgt vor:

- Falls $k < \ell$ gilt, so speichern wir das neue Element in $A[k]$, was eine Zeiteinheit kostet.
- Falls $k = \ell$ gilt, so legen wir ein neues Array A' der Größe $c \cdot \ell$ an, kopieren alle ℓ Elemente aus A an die ersten ℓ Positionen von A' und speichern das neue Element in $A'[\ell]$. Das Anlegen von A' und Kopieren der ℓ Elemente kostet dabei $\leq 2c \cdot \ell$ Zeiteinheiten. Das Speichern des neuen Elements kostet eine Zeiteinheit.

Zeigen Sie mit Hilfe der Amortisationsanalyse, dass die Laufzeit für das Einfügen von n Elementen in ein anfangs leeres Array der Größe $\ell = 1$ insgesamt $\mathcal{O}(n)$ ist.

Lösungsvorschlag:

Wir berechnen für jedes Einfügen $3 + 2/(c-1)$ Kosten. Somit können wir bei jedem Einfügevorgang, bei dem $k < \ell$ gilt, $2 + 2/(c-1)$ Einheiten auf das Konto legen. Daher sind nach $(c-1)\ell$ Einfügeoperationen (bei der frühestens $k = \ell$ gilt) gerade $2c \cdot \ell$ Einheiten auf dem Konto.

Somit kann jedes erweitern des Arrays bezahlt werden.

Aufgabe 4 (10 Punkte)

a) Gegeben sei die Hashtabelle in Abbildung 1, in der Kollisionen durch Linear Probing behandelt werden. Für die Hashtabelle wird die Hashfunktion

$$h(x) = 2x \bmod 13$$

verwendet. Führen Sie auf der Hashtabelle die angegebenen Operationen nacheinander aus und zeichnen Sie nach jeder Operation das Zwischenresultat:

(a) `insert(39)`

(b) `insert(33)`

(c) `insert(31)`

(d) `delete(20)`

(e) `delete(44)`

(f) `insert(16)`

(g) `insert(26)`

Bemerkung: Die `delete`-Operation markiert ein zu entfernendes Element *nicht*, sondern entfernt das Element tatsächlich aus der Hashtabelle. Dabei muss darauf geachtet werden, dass eventuell noch Elemente verschoben werden müssen.

	20	14		28			10		18	44	25	
0	1	2	3	4	5	6	7	8	9	10	11	12

Abbildung 1: Hash Tabelle

b) Eine Familie $\mathcal{H}$ von Hashfunktionen von einem Universum U in eine Menge B heißt ϵ-universell, falls für alle Paare $k, \ell \in U$ mit $k \neq \ell$ gilt:

$$\Pr[h(k) = h(\ell)] \leq \epsilon,$$

wobei h eine gleichverteilt aus $\mathcal{H}$ gezogene Hashfunktion sei.

Gegeben sei der Satz: Für jede Familie H von Hashfunktionen von einem Universum U in eine Menge B gibt es Elemente $x, y \in U$, $x \neq y$ mit

$$\delta(x, y, H) \geq \frac{|H|}{|B|} - \frac{|H|}{|U|},$$

wobei $\delta(x, y, H) := |\{h \in H | h(x) = h(y)\}|$.

Leiten Sie mit Hilfe des Satzes für ϵ eine untere Schranke in Abhängigkeit von $|U|$ und $|B|$ her.

Lösungsvorschlag:

a) (a) `insert(39)` $2 \cdot 39 \bmod 13 = 0$

39	20	14		28			10		18	44	25	
0	1	2	3	4	5	6	7	8	9	10	11	12

(b) `insert(33)` $2 \cdot 33 \bmod 13 = 66 \bmod 13 = 1$

39	20	14		28			10		18	44	25	33
0	1	2	3	4	5	6	7	8	9	10	11	12

(c) `insert(31)` $2 \cdot 31 \bmod 13 = 62 \bmod 13 = 10$

39	20	14		28			10	31	18	44	25	33
0	1	2	3	4	5	6	7	8	9	10	11	12

(d) `delete(20)` Es muss nur das Element 33 verschoben werden.

39	33	14		28			10	31	18	44	25	
0	1	2	3	4	5	6	7	8	9	10	11	12

(e) `delete(44)` $2 \cdot 18 \bmod 13 = 2 \cdot 44 \bmod 13 = 88 \bmod 13 = 10$, daher wird 18 auf die 10 geschoben, auf die freie 9 kommt dann die 31.

39	33	14		28			10		31	18	25	
0	1	2	3	4	5	6	7	8	9	10	11	12

(f) `insert(16)` $2 \cdot 16 \bmod 13 = 32 \bmod 13 = 6$

39	33	14		28		16	10		31	18	25	
0	1	2	3	4	5	6	7	8	9	10	11	12

(g) `insert(26)` $2 \cdot 26 \bmod 13 = 0$

39	33	14		28		16	10		31	18	25	26
0	1	2	3	4	5	6	7	8	9	10	11	12

b) Da ϵ-universell eine stärkere Aussage ist, als die Vorausetzung im Satz, gilt direkt: $\frac{1}{|B|} - \frac{1}{|U|} \leq \epsilon$.

Aufgabe 5 (6 Punkte)

In dem unten gezeichneten (Min-)Fibonacci-Heap wurden noch keine Knoten auf Grund von Löschungen von Kindern markiert. Führen Sie auf dem dargestellten Fibonacci-Heap, wobei der Min-Pointer zu Beginn auf das Element mit dem Schlüssel 2 zeigt, die Operationen, in folgender Reihenfolge durch:

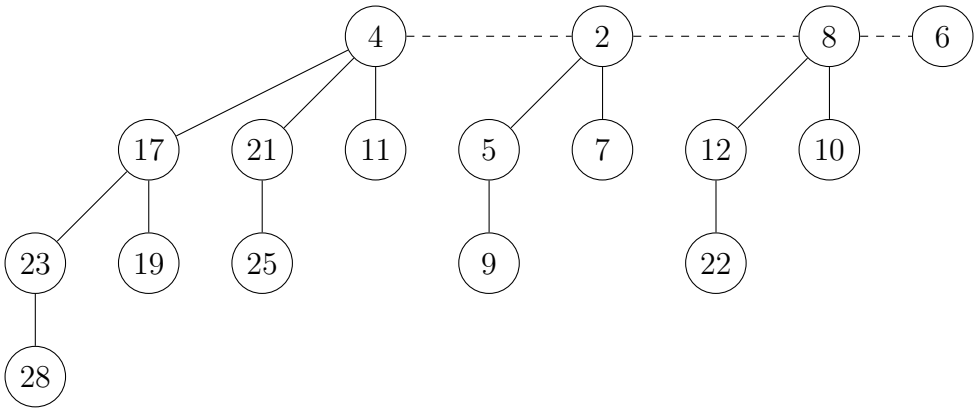
a) insert(1),

b) decreaseKey(23,3),

c) extractMin(),

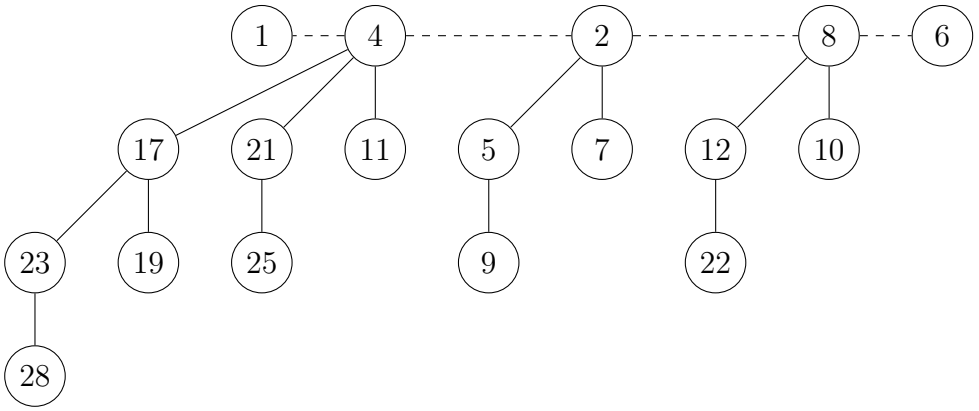
d) delete(19)

Zeichnen Sie nach jeder Operation den resultierenden Fibonacci-Heap.

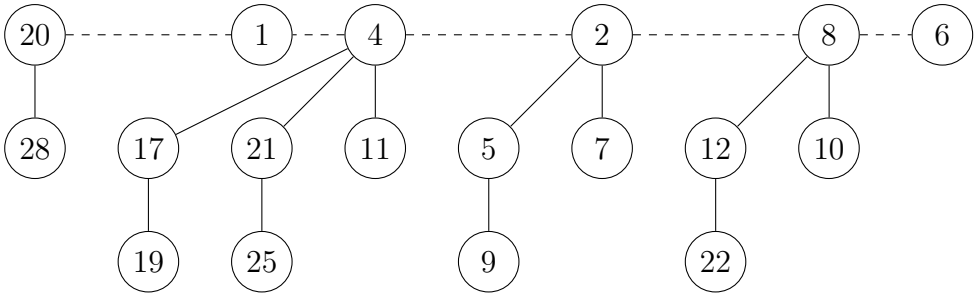


Lösungsvorschlag:

a)

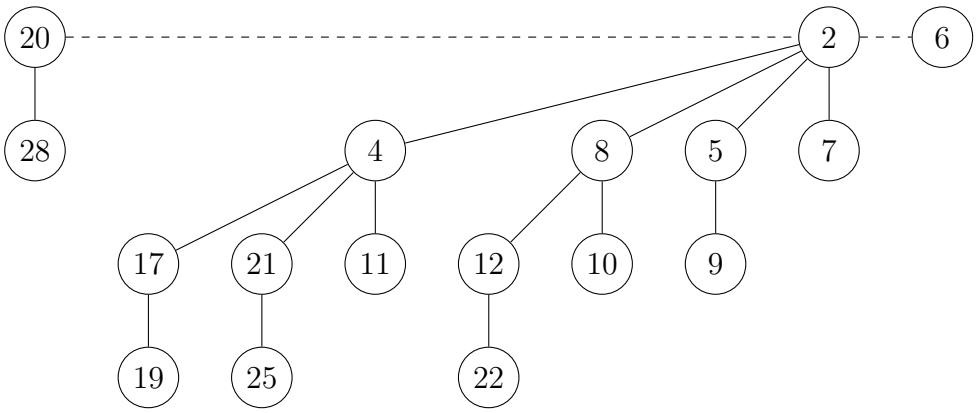


b)

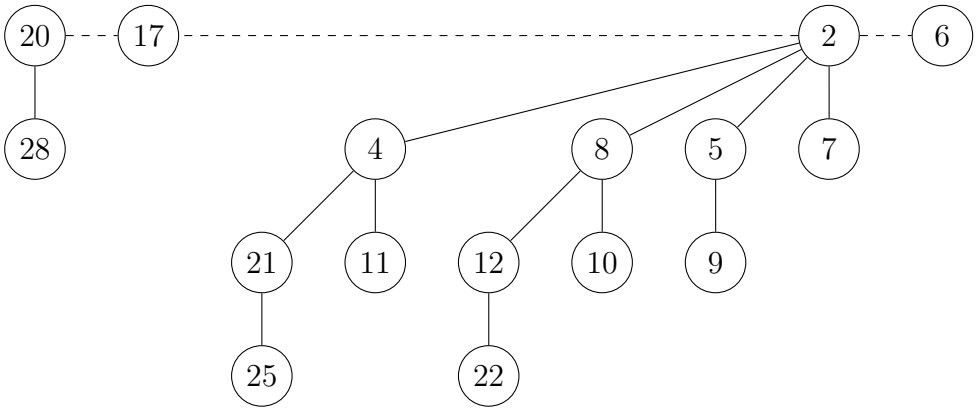


und 17 ist markiert

c)



d) Da 17 schon markiert ist, wird sie in die Wurzel geschoben



Aufgabe 6 (9 Punkte)

Sei $G = (V, E)$ ein zusammenhängender Graph und $C \subseteq V$. Wir definieren $d(v, w)$ als die kleinste Anzahl von Kanten auf einem Pfad zwischen zwei Knoten v und w aus V. Wir erweitern d zu $d : V \times 2^V \to \mathbb{N}$ und setzen $d(v, X) := \min_{w \in X} d(v, w)$.
Wir definieren den Radius von C als $r_C := \max_{v \in V} d(v, C)$. Die Menge C wird k-Center von G genannt, wenn $|C| = k$ gilt und

$$r_C \leq \min_{Y \subseteq V, |Y| = k} r_Y$$

erfüllt ist.
Es ist im Allgemeinen NP-vollständig, ein k-Center zu berechnen.

a) Nehmen Sie an, Sie hätten ein Orakel, welches Ihnen das k-Center C eines Graphen $G = (V, E)$ verrät. Geben Sie einen möglichst effizienten Algorithmus an, der einen Wald aus k disjunkten Bäumen T_i $1 \leq i \leq k$ berechnet, so dass $V = \biguplus_{i=1}^{k} V(T_i)$ gilt und der Durchmesser von jedem Baum T_i kleiner oder gleich $2 \cdot r_C$ ist.

   Geben Sie auch die Laufzeit ihres Algorithmus an, und beweisen Sie die Korrektheit Ihres Algorithmus.

b) Geben Sie einen Algorithmus (in Pseudo-Code) an, der ein 1-Center in Polynomialzeit findet. Beweisen Sie die worst-case-Laufzeit Ihres Algorithmus.

Lösungsvorschlag:

a) Wir initialisieren die Queue des BFS-Algorithmus mit den Nachbarn der Absoluten Zentren.

   Dann führen wir den BFS-Algorithmus mit der Queue aus und merken uns die k Spannbäume (die sich nicht interferieren). Die letzte Kante hat eine maximale Entfernung zu einem Zentrum von r_C, sonst wäre C kein k-Center.

   Die Laufzeit beträgt dann gerade $\mathcal{O}(m + n)$.

b) Wir berechnen von jedem Knoten eine BFS. Der Knoten, dessen BFS den geringsten Radius liefert, bildet ein 1-Center. Damit ergibt sich ein Laufzeit von $\mathcal{O}(n \cdot (n + m))$.

Aufgabe 7 (6 Punkte)

Sei T ein gewurzelter Binärbaum mit n Knoten. Der *kleinste gemeinsame Vorfahr* zweier Knoten x und y in T ist der Knoten z mit der größten Tiefe im Baum, der sowohl Vorfahr von x als auch Vorfahr von y ist. Das **kgV-Problem** besteht darin, zu zwei gegebenen Knoten (nicht den Schlüsseln) x und y den Schlüssel des kleinsten gemeinsamen Vorfahren beider Knoten zu bestimmen.

a) Geben Sie einen effizienten Algorithmus an, der das **kgV-Problem** (ohne vorherige Berechnungen auf dem gesamten Baum) für allgemeine gewurzelte Binärbäume löst und analysieren Sie die Laufzeit Ihres Algorithmus in Abhängigkeit von den Knotentiefen d_x, d_y und d_z.

b) Nun nehmen wir an, dass die Schlüssel eines gewurzelten, geordneten Binärbaums T in allen Knoten die Suchbaumeigenschaft erfüllen. Geben Sie einen möglichst effizienten Algorithmus an, der das **kgV-Problem** unter dieser Einschränkung löst, und analysieren Sie die Laufzeit Ihres Algorithmus in Abhängigkeit von den Knotentiefen d_x, d_y und d_z.

Hinweis: Ein geordneter Binärbaum ist ein Binärbaum, bei dem die Kinder von internen Knoten geordnet sind.

Lösungsvorschlag:

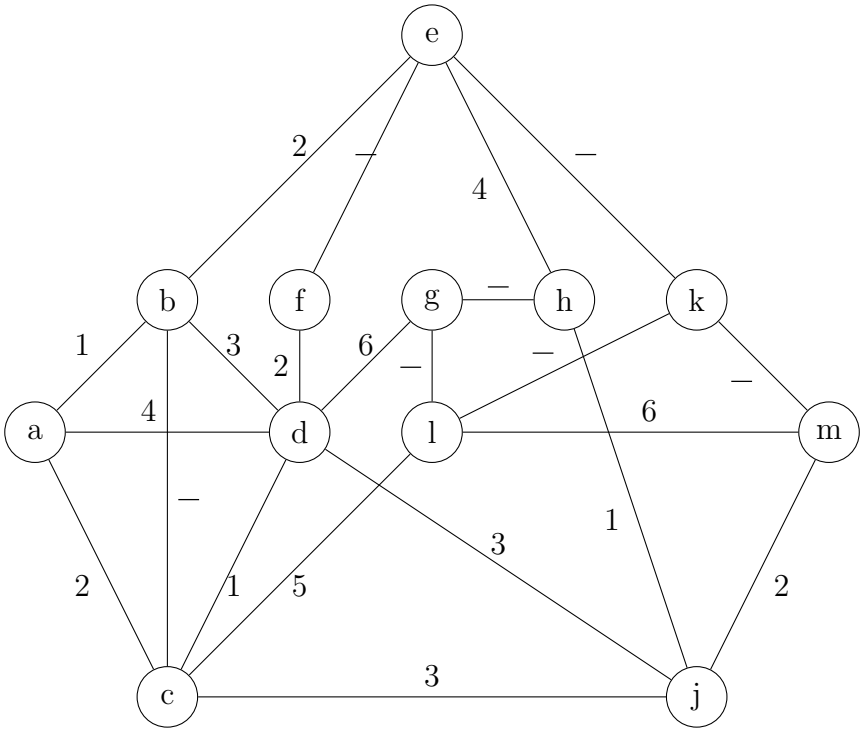
1. Zuerst geht mach vom Knoten x zur Wurzel des Baumes und markiert auf dem Weg dahin alle Knoten als besucht. Danach geht man vom Knoten y ebenfalls solange in Richtung Wurzel, bis man auf den ersten markierten Knoten stößt.

   Zur Analyse: Da von jeden Knoten maximal bis zur Wurzel gelaufen wird, ist die Laufzeit: $\mathcal{O}(\max\{d_x, d_y\})$.

2. Falls die Schlüssel die allgemeine Suchbaumeigenschaft erfüllen, dann kann man einen Algorithmus konstruieren, der nur die Laufzeit $\mathcal{O}(d_z)$ hat, wobei z der kleinste gemeinsame Vorfahr von x und y ist. Die Idee ist, solange von der Wurzel des Baumes nach unten zu gehen, bis man ein Elemente p gefunden hat, so dass $key[x] \leq key[p] \leq key[y]$ gilt. Das ist das gesuchte Elemente z.

Aufgabe 8 (7 Punkte)

Gegeben sei der untenstehende Graph. Führen Sie den Algorithmus von Kruskal zur Berechnung eines Minimalen Spannbaums aus und vervollständigen Sie die leeren Felder im Graphen und im Berechnungsprotokoll. Kann im Berechnungsprotokoll mehr als eine Kante eingetragen werden, so ist die Kante auszuwählen, die einen Knoten mit möglichst kleinem Knotengrad besitzt.
Markieren Sie den berechneten Spannbaum.
Hinweis: Alle Kantengewichten sind natürliche Zahlen.



Reihenfolge, in der die Kanten vom Algorithmus angeschaut werden.

1. ab,
2. _,
3. gl,
4. jh,
5. be,
6. _,
7. _,
8. bc,
9. fe,
10. ac,

11. bd,
12. _,
13. dj,
14. ad,
15. ek,
16. _,
17. cl,
18. dg,
19. gh,
20. km,
21. lk,
22. lm.

Lösungsvorschlag:

1. ab,
2. cd,
3. gl,
4. jh, (letze Kante in Gewicht 1)
5. be,
6. fd, (hat kleineren Grad als mj in f)
7. mj,
8. bc,
9. fe,
10. ac, (letze Kante mit Gewicht 2)
11. bd,
12. cj,
13. dj, (letzte Kante mit Gewicht 3)
14. ad,
15. ek,

16. eh, (letzte Kante mit Gewicht 4)
17. cl, (letzte Kante mit Gewicht 5)
18. dg,
19. gh,
20. km,
21. lk,
22. lm. (letzte Kante mit Gewicht 6)