

Informatik II: Algorithmen & Datenstrukturen

Donnerstag, 28. August, 2014, 14:00 – 17:00

Name:

Matrikelnummer:

Unterschrift:

Blättern Sie nicht um bevor Sie dazu aufgefordert werden!

- Schreiben Sie **auf alle Blätter** (inklusive Deckblatt und etwaiger zusätzlicher Blätter) Ihren **Vornamen, Nachnamen** und Ihre **Matrikelnummer**.
- **Unterschreiben Sie das Deckblatt!** Ihre Unterschrift bestätigt, dass Sie alle Fragen ohne nicht erlaubte Hilfsmittel beantwortet haben.
- Schreiben Sie **lesbar** und nur mit **dokumentenechten** Stiften. Schreiben Sie nicht in **rot** oder **grün** und benutzen Sie keinen Bleistift!
- Alle schriftlichen Hilfsmittel (Bücher, Vorlesungsunterlagen, handschriftliche Notizen, etc.) sind erlaubt. **Elektronische Hilfsmittel sind nicht erlaubt.**
- Die Klausur besteht aus 7 Aufgaben und 120 Punkten. Zum Bestehen reichen 50 Punkte.
- Benutzen Sie für jede Aufgabe eine eigene Seite.
- Es wird nur eine Lösung pro Aufgabe gewertet. Vergewissern Sie sich, dass Sie zusätzliche Lösungen selbst entwerten. Falls mehrere Lösungen zu einer Aufgabe existieren, so wird die schlechtere Lösung gewertet.
- Die folgenden Regeln gelten überall, außer sie werden explizit außer Kraft gesetzt. Bei Laufzeitfragen ist wie üblich nur die asymptotische Laufzeit notwendig. Wenn Sie einen Algorithmus angeben sollen, so können Sie Pseudocode angeben, eine Beschreibung der Funktionsweise Ihres Algorithmus ist allerdings ausreichend. Algorithmen sind immer effizient zu konstruieren, d.h., mindestens polynomiell und i.d.r. schneller als eine naive Lösungsmethode. Algorithmen aus der Vorlesung können grundsätzlich als Blackbox verwendet werden.
- **Erklären Sie Ihre Lösungen, außer es wird explizit darauf hingewiesen, dass dies nicht nötig ist! Nur das Endresultat aufzuschreiben ist nicht ausreichend.**

Frage	1	2	3	4	5	6	7	Total
Punkte								
Maximum	26	8	13	22	17	16	18	120

Aufgabe 1: Kurze Fragen (26 Punkte)

Beantworten Sie die folgenden Fragen kurz.

- (a) **(2 Punkte)** Was ist die Best-Case Laufzeit von Selection Sort (in \mathcal{O} -Notation, ohne Begründung).
- (b) **(2 Punkte)** Welches der in der Vorlesung besprochenen Sortierverfahren verhält sich am Besten, wenn die Eingabe ein bereits sortiertes Array ist? (ohne Begründung)
- (c) **(3 Punkte)** Eine Anwendung verwendet eine Hashtabelle so, dass zur einmaligen Initialisierung alle (Schlüssel, Wert)-Paare eingefügt werden und danach nur noch (sehr viele) *find*-Operationen ausgeführt werden. Welche Hashtabellen-Implementierung aus der Vorlesung eignet sich besonders für diese Anwendung und wieso?
- (d) **(3 Punkte)** Bestimmen Sie die Editierdistanz von `Tier` und `Tor` und geben Sie die notwendigen Operationen an. Begründen Sie auch, wieso es nicht besser geht?
- (e) **(4 Punkte)** Geben Sie einen effizienten Algorithmus an, um einen Spannbaum mit maximalem Gesamtgewicht zu berechnen (kurze Beschreibung genügt).
- (f) **(4 Punkte)** Ihnen ist von einem großen Netzwerk eine Liste der Knoten gegeben, zusammen mit den Knotengraden. Ihre Aufgabe ist es, die Knoten des Netzwerks nach ihren Knotengraden zu sortieren. Welches Sortierverfahren aus der Vorlesung eignet sich am besten dazu (mit Begründung)?
- (g) **(4 Punkte)** Beschreiben Sie, wie eine FIFO-Warteschlange *Queue* mit Hilfe einer Prioritätswarteschlange (Heap) implementiert werden kann.
- (h) **(4 Punkte)** Gegeben sei ein Baum T mit n Elementen, welcher für die gleichen Schlüssel sowohl ein binärer Suchbaum, als auch ein Min-Heap ist (alle Schlüssel seien verschieden). Welche Höhe kann T haben?

Hinweis: Überlegen Sie sich zuerst an einem kleinen Beispiel, was passiert.

Aufgabe 2: Landau-Notation (8 Punkte)

Beweisen oder widerlegen Sie die folgenden Behauptungen durch Benutzen der exakten Definitionen von $\mathcal{O}(\cdot)$ und $\Omega(\cdot)$.

- (a) (4 Punkte) $\sqrt{n+7} \in \mathcal{O}(\sqrt{n})$.
- (b) (4 Punkte) $\log(n^2) \in \Omega((\log n)^2)$.

Aufgabe 3: Mystische Multiplikationen (13 Punkte)

Gegeben sei ein Integer Array A der Länge n und folgender Code:

```
boolean myst(int[] A) {
    int n = A.length;
    for (int i = 0; i < n-1; i++) {
        for (int j = i+1; j < n; j++) {
            for (int k = 0; k < n; k++) {
                if (A[i] * A[j] == A[k]) {
                    return true;
                }
            }
        }
    }
    return false;
}
```

- (a) (4 Punkte) Was berechnet die Funktion *myst* (bzw., in welchen Fällen gibt *myst* true zurück)?
- (b) (3 Punkte) Welche (asymptotische) Laufzeit hat *myst(int[] A)* im Worst Case und im Best Case (in Abhängigkeit von n , ohne Begründung)?
- (c) (6 Punkte) Geben Sie, z.B. unter Zuhilfenahme einer geeigneten Datenstruktur, einen Algorithmus mit gleicher Ausgabe wie *myst* an, dessen *asymptotische* Laufzeit im Worst Case jedoch strikt besser als diejenige von *myst* ist. Was ist die Laufzeit Ihres Algorithmus?

Aufgabe 4: Binäre Suchbäume (22 Punkte)

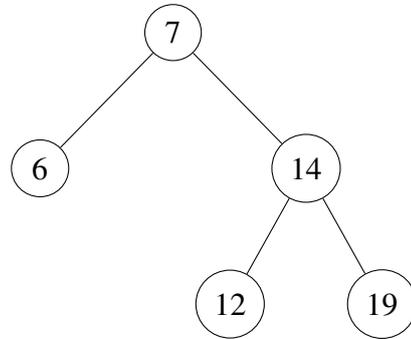
(a) (6 Punkte) Führen Sie nacheinander die nachfolgenden 8 Einfüge- und Lösch-Operationen auf den unten angegebenen binären Suchbaum aus und zeichnen Sie den resultierenden Baum nach **jeder Gruppe** von Operationen.

1) *insert(23), delete(19)*

2) *insert(13), insert(10), insert(4), insert(18)*

3) *delete(6)*

4) *delete(14)*



(b) (6 Punkte) Können die folgenden Folgen in einer *Suchabfrage* in einem binären Suchbaum auftreten? Zeichnen Sie einen Beispielbaum auf, wenn die Antwort “ja” lautet, andernfalls begründen Sie Ihre Antwort.

- Folge 1: 44, 12, 15, 37, 19, 29, 28
- Folge 2: 53, 64, 83, 75, 69, 60, 66

(c) (10 Punkte) Man nennt einen binären Suchbaum T einen *AVL-Baum*, wenn für *jeden* Knoten u gilt, dass die Höhen der beiden in $u.left$ und $u.right$ gewurzelten Teilbäume sich um maximal 1 unterscheiden. Beschreiben Sie eine Methode *isAVL* als Pseudocode, welche einen binären Suchbaum T auf diese Eigenschaft testet. Was ist die Laufzeit Ihres Algorithmus, und warum?

Hinweis: Schreiben Sie zuerst eine rekursive Methode $avlHeight(v)$, welche, angewandt auf einen Knoten v , die Höhe des in v gewurzelten Teilbaums wiedergibt falls der Teilbaum selbst ein AVL-Baum ist, und -1 sonst.

Aufgabe 5: Divide & Conquer, Sortieren (17 Punkte)

- (a) **(5 Punkte)** Gegeben sei ein Integer-Array A der Länge n . Finden Sie einen Divide & Conquer Algorithmus, der das Maximum der Elemente in A findet, ohne das Array zu sortieren. Geben Sie die Rekursionsgleichung für die Laufzeit an – diese muss *nicht* gelöst werden.
- (b) **(12 Punkte)** Gegeben sei ein aufsteigend sortiertes Integer-Array B mit n Elementen. Nun werden k beliebige Elemente ($k \ll n$) individuell verringert (man weiß nicht, welche k Elemente verändert wurden), so dass das resultierende Array im Allgemeinen nicht mehr sortiert ist.

Geben Sie einen Algorithmus an, der das Array in Zeit $\mathcal{O}(n + k \log k)$ erneut in einen sortierten Zustand bringt.

Hinweis: Finden Sie zuerst alle Elemente, welche dadurch, dass sie heruntersetzt wurden, die Sortierreihenfolge zerstören (wenn man diese Elemente entfernt, sind die restlichen Elemente richtig sortiert). Beachten Sie, dass das höchstens k Elemente sind, aber nicht exakt k Elemente sein müssen.

Aufgabe 6: Graphen & Graphtraversierung (16 Punkte)

Gegeben sei ein *gerichteter* Graph $G = (V, E)$. Herausgefunden werden soll, ob G einen Kreis enthält. Ein Kreis ist eine Knotenfolge u_1, u_2, \dots, u_k , so dass $(u_i, u_{i+1}) \in E$ für alle $1 \leq i < k$, sowie $(u_k, u_1) \in E$. Es gelte $k \geq 2$, d.h., es kann Kreise der Länge 2 geben.

- (a) **(4 Punkte)** Erläutern Sie einen Algorithmus, welcher möglichst effizient herausfindet, ob ein gegebener gerichteter Graph G einen Kreis enthält.
- (b) **(2 Punkte)** Welche Datenstruktur würden Sie verwenden, um den Graph zu repräsentieren und warum: Adjazenzlisten oder Adjazenzmatrix?
- (c) **(10 Punkte)** Geben Sie Pseudocode für Ihren Algorithmus an.

Aufgabe 7: Kürzeste Wege zur Arbeit (über den Berg) (18 Punkte)

Geben Sie jeweils einen Algorithmus an, um die folgenden Probleme zu lösen (dabei dürfen in der Vorlesung behandelte Algorithmen als Blackbox verwendet werden):

Gegeben sei ein Straßennetz als ein ungerichteter, gewichteter Graph $G = (V, E, w)$, $n = |V|$, $m = |E|$.

- (a) **(2 Punkte)** Sie wohnen in Punkt $a \in V$ und arbeiten in Punkt $b \in V$. Wie berechnen Sie den kürzesten Weg zur Arbeit?

Geben Sie außerdem die asymptotische Laufzeit Ihres Algorithmus (ohne Begründung) an.

- (b) **(4 Punkte)** Sie möchten umziehen in eine neue Wohnung $a' \in W$, wobei $W \subseteq V$ eine Menge von potentiellen Wohnungen ist. Wie finden Sie a' , wenn diese von allen Wohnungen in W den kürzesten Weg zur Arbeit $b \in V$ haben soll?

Geben Sie außerdem die asymptotische Laufzeit Ihres Algorithmus an (ohne Begründung).

- (c) **(12 Punkte)** Sie haben sich wegen der schönen Lage für eine andere Wohnung $c \in V$ mit Fahrradnähe zu $b \in V$ entschieden. Jeder Knoten v im Graphen hat eine Höhe $h_v \in \mathbb{R}$ (der Einfachheit halber seien alle Knotenhöhen voneinander verschieden) und je nachdem, in welcher Richtung man über eine Kante $e = \{u, v\}$ geht, ist diese eine Bergaufkante oder eine Bergabkante. Aus unbekanntem Grund wollen Sie von Ihrer Wohnung c aus zunächst ausschließlich bergauf fahren – bis zu einem Punkt $x \in V$ – und von dort aus ausschließlich bergab bis zum Punkt b .

Beschreiben Sie einen (effizienten) Algorithmus, welcher herausfindet, ob so ein Weg existiert und welcher bei Existenz den kürzesten aller solchen Wege, inklusive des höchsten Punkts x , zurückgibt. Welche Laufzeit hat Ihr Algorithmus (ohne Begründung)?

Hinweis: Verfahren aus der Vorlesung können als Blackbox verwendet werden. Beschreiben Sie Ihren Algorithmus nur, zu hoher Genauigkeit (z.B. via Pseudocode) kostet Sie zu viel Zeit.

Informatik II: Algorithmen & Datenstrukturen

Mittwoch, 4. März, 2015, 9:00 – 12:00

Name:

Matrikelnummer:

Unterschrift:

Blättern Sie nicht um bevor Sie dazu aufgefordert werden!

- Schreiben Sie **auf alle Blätter** (inklusive Deckblatt und etwaiger zusätzlicher Blätter) Ihren **Vornamen, Nachnamen** und Ihre **Matrikelnummer**.
- **Unterschreiben Sie das Deckblatt!** Ihre Unterschrift bestätigt, dass Sie alle Fragen ohne nicht erlaubte Hilfsmittel beantwortet haben.
- Schreiben Sie **lesbar** und nur mit **dokumentenechten** Stiften. Schreiben Sie nicht in **rot** oder **grün** und benutzen Sie keinen Bleistift!
- Alle schriftlichen Hilfsmittel (Bücher, Vorlesungsunterlagen, handschriftliche Notizen, etc.) sind erlaubt. **Elektronische Hilfsmittel sind nicht erlaubt. Das inkludiert Mobiltelefone.**
- Die Klausur besteht aus 7 Aufgaben und 120 Punkten. Zum Bestehen reichen 50 Punkte.
- Bleiben Sie nicht unnötig lange an einer Aufgabe hängen.
- Benutzen Sie für jede Aufgabe eine eigene Seite.
- Markieren Sie Schmierpapier als solches. Dieses können Sie dann auch abgeben, Notizen darauf können im Zweifelsfall zu Ihren Gunsten verwendet werden, nicht jedoch zu Ihrem Ungunsten.
- Es wird nur eine Lösung pro Aufgabe gewertet. Vergewissern Sie sich, dass Sie zusätzliche Lösungen selbst entwerten. Falls mehrere Lösungen zu einer Aufgabe existieren, so wird die schlechtere Lösung gewertet.
- Die folgenden Regeln gelten überall, außer sie werden explizit außer Kraft gesetzt. Bei Laufzeitfragen ist wie üblich nur die asymptotische Laufzeit notwendig. Wenn Sie einen Algorithmus angeben sollen, so *können* Sie Pseudocode angeben — eine Beschreibung der Funktionsweise Ihres Algorithmus ist allerdings ausreichend. Algorithmen sind immer effizient zu konstruieren, d.h., mindestens polynomiell und i.d.r. schneller als eine naive Lösungsmethode. Algorithmen aus der Vorlesung können grundsätzlich als Blackbox verwendet werden.
- **Erklären Sie Ihre Lösungen, außer es wird explizit darauf hingewiesen, dass dies nicht nötig ist! Nur das Endresultat aufzuschreiben ist nicht ausreichend.**

Frage	1	2	3	4	5	6	7	Total
Punkte								
Maximum	20	12	10	12	14	18	34	120

Aufgabe 1: Kurze Fragen (20 Punkte)

Beantworten Sie die folgenden Fragen kurz.

- (a) **(4 Punkte)** Sie haben $m = 2^k$ ($k \in \mathbb{N}$) aufsteigend sortierte Arrays A_1, \dots, A_m mit jeweils n Elementen gegeben. Geben Sie einen effizienten Algorithmus an, um all diese Arrays zu einem einzelnen **sortierten** Array A der Größe mn zusammenzufassen. Was ist die Laufzeit Ihres Algorithmus in Abhängigkeit von m und n ?
- (b) **(6 Punkte)** Gegeben ist ein Array mit n **positiven** Integers. Geben Sie einen Divide & Conquer Algorithmus in **Pseudocode** an, um das kleinste gerade Element im Arrays zu finden, ohne das Array zu sortieren. Wenn es kein solches Element gibt, soll der Algorithmus -1 zurückgeben. Geben Sie auch die Rekursionsgleichung an; diese müssen Sie nicht lösen.
- (c) **(3 Punkte)** Beweisen Sie oder geben Sie ein Gegenbeispiel an zu folgendem Satz: “Editierdistanz erfüllt die Dreiecksungleichung.” D.h., für alle Strings x, y und z gilt $d(x, y) + d(y, z) \geq d(x, z)$, wenn d die Editierdistanz ist.
- (d) **(3 Punkte)** Ein Graph $G = (V, E)$ mit n Knoten und $m = n^{4/3}$ Kanten ist aus Speicherplatzgründen als Adjazenzliste abgespeichert. Für ein beliebiges Knotenpaar (u, v) soll mit dem Befehl `EXISTS($\{u, v\}$)` in möglichst kurzer Zeit getestet werden können, ob $\{u, v\}$ in E enthalten ist. Wie erweitern Sie die Datenstruktur, um das möglichst effizient zu realisieren und den Speicherplatz dabei gering zu halten? Geben Sie den asymptotischen Platzbedarf der Datenstruktur und die asymptotische Laufzeit von `EXISTS` jeweils in Abhängigkeit von n an.
- (e) **(4 Punkte)** Für ein großes Programmierprojekt müssen Sie wiederholt eine Vielzahl von Dateien kompilieren. Dabei treten Abhängigkeiten bezüglich der Kompilierreihenfolge auf, wie z.B. dass Datei A vor Datei B kompiliert werden muss (abgekürzt als $A \rightarrow B$). Die Abhängigkeitsrelationen sind als **gerichteter** und **kreisfreier** Graph $G = (V, E)$ gegeben (mit $n = |V|$ und $m = |E|$). Geben Sie einen Algorithmus an, welcher effizient das Problem löst, in welcher Reihenfolge die Dateien zu kompilieren sind. Was ist die asymptotische Laufzeit Ihres Algorithmus als Funktion von n und m ?

Aufgabe 2: Landau Notation (12 Punkte)

Entscheiden Sie für jede der folgenden Aussagen, ob diese wahr oder falsch ist. Geben Sie im letzteren Fall eine Funktion f und/oder eine Funktion g an, welche die Ungültigkeit der Aussage zeigen. Eine weitere Begründung ist nicht notwendig.

1) $\forall f(n) \in \Omega(\log n), g(n) \in \mathcal{O}(n) : g(n) \in \mathcal{O}(f(n))$

2) $\forall f(n) \in \Omega(\log n), g(n) \in \mathcal{O}(n) : f(n) \in \mathcal{O}(g(n))$

3) $\forall f(n) \in \Omega(\log n), g(n) \in \mathcal{O}(n) : f(n) \in \Omega(\log(g(n)))$

4) $\forall f(n) \in \Omega(\log n), g(n) \in \mathcal{O}(n) : f(n) \in \Theta(\log(g(n)))$

5) Falls $f(n) \in \mathcal{O}(g^2(n))$, dann gilt $f(n) \in \Omega(g(n))$.

6) Falls $f(n) \in \Theta(2^n)$, dann gilt $f(n) \in \Theta(3^n)$.

7) Falls $f(n) = \mathcal{O}(n^3)$, dann gilt $\log f(n) \in \mathcal{O}(\log n)$.

Aufgabe 3: Mengen vereinigen (10 Punkte)

Gegeben sei ein Array A von Tupeln (a, b) , wobei das Tupel (a, b) das Intervall $[a, b] \subseteq \mathbb{R}$ repräsentiert. Ihre Aufgabe ist es, eine Prozedur **SIMPLIFY** zu schreiben, welches so ein Array nimmt und ein neues Array dieser Form produziert, welches die Vereinigung **aller** Intervalle in A darstellt und dabei eine **minimale** Anzahl an Tupeln verwendet. Angewandt auf sich zwei überschneidende Intervalle $[a_1, b_1]$ und $[a_2, b_2]$ wird das Tupel $(\min\{a_1, a_2\}, \max\{b_1, b_2\})$ zurückgegeben, gibt es keine Überschneidung, so ist keine Simplifizierung möglich und beide Tupel werden unverändert zurückgegeben.

Beispiel: **SIMPLIFY** angewandt auf $A = \langle(3, 7), (1, 4), (10, 12), (6, 8)\rangle$ gibt entweder das Array $\langle(10, 12), (1, 8)\rangle$ oder das Array $\langle(1, 8), (10, 12)\rangle$ zurück, da $[1, 8] = [1, 4] \cup [3, 7] \cup [6, 8]$; eine weitere Simplifizierung ist allerdings nicht möglich.

Geben Sie einen Algorithmus an, der dieses Problem möglichst effizient löst. Welche Laufzeit hat Ihr Algorithmus bei einem Array aus n Elementen?

Aufgabe 4: Algorithmus raten (12 Punkte)

Gegeben seien zwei Integer Arrays a und b der Länge n und m , und folgender Code:

```
// add(w) fuegt w am Ende der verketteten Liste an.
```

```
List<Integer> myst(int[] a, int[] b) {  
    List<Integer> c = new LinkedList<Integer>();  
    riddle(a, b, c);  
    riddle(b, a, c);  
    return c;  
}  
  
void riddle(int[] x, int[] y, List<Integer> z) {  
    for (int i = 0; i < x.length; i++) {  
        boolean take = true;  
        for (int j = 0; j < y.length; j++)  
            if (x[i] == y[j]) take = false;  
        if (take == true) z.add(x[i]);  
    }  
}
```

- (a) (6 Punkte) Erklären Sie was die Funktion MYST berechnet. Welche Komplexität hat diese Funktion in Abhängigkeit von n und m ?
- (b) (6 Punkte) Können Sie den gegebenen Algorithmus so modifizieren, dass die Funktion das gleiche Ergebnis berechnet, allerdings asymptotisch möglichst effizient ist (in n und m)? Beschreiben Sie Ihre Lösung – die Verwendung von Pseudocode oder Java ist nicht notwendig. Welche Zeitkomplexität hat Ihr Algorithmus?

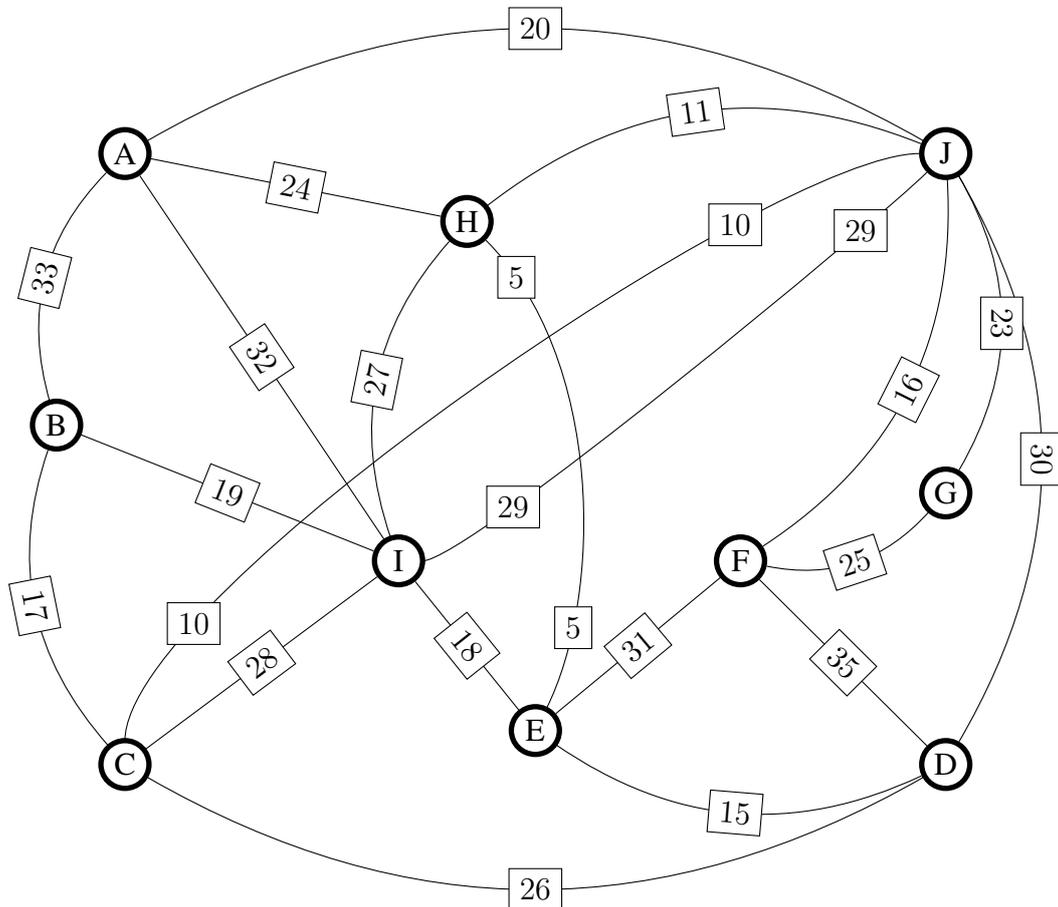
Aufgabe 5: Minimaler Spannbaum (14 Punkte)

- (a) (10 Punkte) Gegeben ist ein **ungerichteter** Graph $G = (V, E)$ mit $|E| = |V| = n$ und einer Gewichtsfunktion $w : E \mapsto \mathbb{N}$. Geben Sie einen Algorithmus an, welcher in Zeit $\mathcal{O}(n)$ einen minimalen Spannbaum T auf G berechnet.

Hinweis: Beachten Sie, dass Bäume auf n Knoten immer $n - 1$ Kanten haben, d.h., bei Graph G handelt es sich um einen "Fastbaum".

- (b) (4 Punkte) Führen Sie auf dem abgebildeten Graph **Prim's** Algorithmus aus. Starten Sie mit dem Knoten C . Markieren Sie (falls farblich: nicht mit rot!!) die Kanten, die am Ende im Baum sind und schreiben Sie neben die Kanten, in welcher **Reihenfolge** diese eingefügt werden.

Hinweis: Wenn Kantengewichte zweifach auf einer Kante sind, dann ist das aus Übersichtsgründen – das Gewicht ist nicht als doppelt so groß zu betrachten.



Aufgabe 6: Ab ins Kino (18 Punkte)

Gegeben sei ein Straßennetz der großen Stadt Kinopolis, dargestellt als **gerichteter** Graph $G = (V, E)$ (mit $n = |V|$ und $m = |E|$) sowie einer Gewichtsfunktion $w : E \mapsto \mathbb{N}$, welche Reisezeiten mit dem **Auto** widerspiegelt. $K \subset V$ sind die Positionen der Kinos von Kinopolis.

- (a) **(6 Punkte)** Zwei Freunde (wohnhaft an den Knoten v_1 und v_2) möchten zusammen ein Kino besuchen. Es gibt $k = |K|$ Kinos, und sie wollen dasjenige Kino in K wählen, welches für beide günstig zu erreichen ist, d.h., die **Summe** der Reisezeiten von beiden soll möglichst kurz sein. Beschreiben Sie, wie Sie dieses Problem lösen würden. Geben Sie die Laufzeit Ihres Algorithmus in Abhängigkeit von n , m und k an.
- (b) **(6 Punkte)** Nehmen Sie nun an, dass $k = |K| = \log n$. Mit der Ankunft von “Star Wars Zero” in den Kinos beschließen sogar $l = \sqrt{n}$ Freunde (wohnhaft bei v_1, v_2, \dots, v_l), gemeinsam ein Kino zu besuchen und wieder soll das Kino gefunden werden, bei dem die Summe der Reisezeiten minimiert wird. Sie merken, dass der direkte Transfer Ihres vorherigen Algorithmus eine recht lange Laufzeit ergibt. Wie müssen Sie Ihren Algorithmus ändern, so dass Ihr Algorithmus schneller läuft? Was ist die Laufzeit?

Einige Kinos sind schwer mit dem Auto zu erreichen und es empfiehlt sich, das Auto ein Stück weit entfernt zu parken und den Rest zu Fuß zurückzulegen (es kann an jedem Knoten in V geparkt werden). Die Gewichtsfunktion $w' : E \mapsto \mathbb{N}$ spiegelt die Laufzeiten **für Fußgänger** wider.

- (c) **(6 Punkte)** Eine einzelne Person will ins Kino. Wie berechnen Sie für diese Person den Parkplatz $p \in V$ und das Kino $x \in K$, so dass die Reisezeit minimiert wird? Was ist die Laufzeit Ihres Algorithmus?

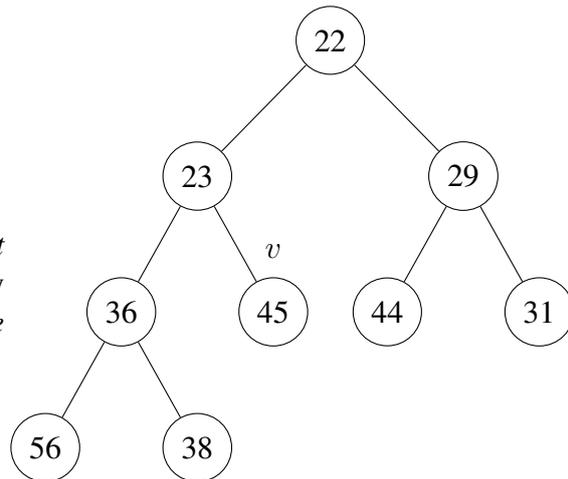
Aufgabe 7: Binäre Suchbäume und Prioritätswarteschlangen (34 Punkte)

- (a) (6 Punkte) Gegeben sei ein einfacher binärer Suchbaum T (die simple Variante aus der Vorlesung, d.h., ohne Rotationen) und zwei Elemente x und y , so dass $x \notin T$ und $y \in T$. Wenn auf diesem Baum $\text{insert}(x)$ und direkt danach $\text{remove}(x)$ ausgeführt werden - ist der resultierende Baum immer identisch zu T ? Was, wenn $\text{remove}(y)$ ausgeführt wird, und direkt danach $\text{insert}(y)$? Argumentieren Sie kurz Ihre Antwort bzw. konstruieren Sie ein Beispiel, falls Ihre Antwort "nein" ist.
- (b) (6 Punkte) $\text{pre}_1 = \{15, 9, 8, 5, 14, 11, 22, 25, 23, 28\}$ und $\text{pre}_2 = \{5, 3, 2, 4, 7, 8, 6\}$ seien zwei Schlüsselreihenfolgen, die angeblich als Ergebnis einer **Preorder** Traversierung eines binären Suchbaums erzeugt wurden. Zeichnen Sie die zugehörigen Suchbäume T_1 und T_2 sofern dies möglich ist; falls nicht, erklären Sie, warum nicht.
- (c) (6 Punkte) Führen Sie **nacheinander** die nachfolgenden Operationen auf der abgebildeten (Min-Heap) Prioritätswarteschlange aus und zeichnen Sie den resultierenden Baum nach **jeder Gruppe** von Operationen.

1) $\text{insert}(42)$, $\text{insert}(19)$, $\text{insert}(12)$

2) $\text{decreaseKey}(v, 15)$, $\text{deleteMin}()$

Hinweis: $\text{decreaseKey}(x, \text{newkey})$ setzt den Wert des Knotens x auf newkey falls newkey kleiner ist als der derzeitige Schlüssel von x .



- (d) In einer Auflistung A von n Zahlenwerten ist der **Median** von A der $\lceil \frac{n}{2} \rceil$ -kleinste Zahlenwert, d.h., wenn man A sortiert, dann steht der Median an der Position $\lceil \frac{n}{2} \rceil$. In der Liste $(4, 1, 37, 2, 0)$ ist der Median 2, in $(1, 0, 9, 4, 12, 8)$ ist er 4.

Nehmen Sie an, ein binärer Suchbaum T ist gegeben, in welchem in jedem Knoten v neben dem Schlüssel **key** auch ein Wert **size** abgespeichert ist, welcher die Anzahl der Knoten im in v gewurzelten Unterbaum T_v wiedergibt (inklusive v).

- (12 Punkte) Schreiben Sie einen Algorithmus **BST-MEDIAN** in **Pseudocode**, welcher als Eingabe die Wurzel von T bekommt und den Median der in T gespeicherten Werte zurückgibt.

Hinweis: Wenn Sie Ihren Algorithmus auch kurz beschreiben, dann können wir eventuelle Fehler in Ihrem Code leichter verstehen und Ihnen in diesem Fall auch mehr Punkte geben; notwendig ist dies jedoch nicht.

- (4 Punkte) Welche Laufzeit hat Ihr Algorithmus in Abhängigkeit von der Anzahl n der Knoten und der Höhe h von T ?