

Matrikelnummer:			
 ÜBUNGSKLAUSUR	Fakultät	Technik	
	Studiengang:	Informatik	
	Jahrgang / Kurs :	2018 A/B/C/D	
	Studienhalbjahr:	2. Semester	
Datum:	KW29/2019	Bearbeitungszeit:	90 Minuten
Modul:	T3INF1003.1	Dozent:	Schulz, Dietzsch,
Unit:	Algorithmen		Kötter, Kochanowski
Hilfsmittel:	Zwei gebundene Dokumente, z.B. Vorlesungsskript, eigene Notizen		

Aufgabe	erreichbar	erreicht
1	9	
2	9	
3	10	
4	10	
5	10	
6	12	
7	16	
Summe	76	

1. Sind Sie gesund und prüfungsfähig?
2. Sind Ihre Taschen und sämtliche Unterlagen, insbesondere alle nicht erlaubten Hilfsmittel, seitlich an der Wand zum Gang hin abgestellt und nicht in Reichweite des Arbeitsplatzes?
3. Haben Sie auch außerhalb des Klausorraumes im Gebäude keine unerlaubten Hilfsmittel oder ähnliche Unterlagen liegen lassen?
4. Haben Sie Ihr Handy ausgeschaltet und abgegeben?

(Falls Ziff. 2 oder 3 nicht erfüllt sind, liegt ein Täuschungsversuch vor, der die Note „nicht ausreichend“ zur Folge hat.)

Aufgabe 1 (5+2+2 Punkte)

Betrachten Sie die Folge $S = (3, 15, 22, 2, 37, 7, 111, 9, 12)$

- a) Sortieren Sie die Folge S aufsteigend gemäß der normalen Ordnung $<$ auf den natürlichen Zahlen. Verwenden Sie das in der Vorlesung gezeigte *Selection-Sort*-Verfahren. Geben Sie den Zustand von S nach jedem Durchlauf der äußersten Schleife an.
- b) Wie viele *Vergleiche* von Elementen der Folge S benötigen Sie?
- c) Betrachten Sie nun die *lexikographische Ordnung* $<_{lex}$, bei der Zahlen wie Namen im Telefonbuch verglichen werden - zunächst die erste Ziffer, und bei Gleichheit genau so die folgenden. Dabei ist z.B. $11 <_{lex} 3$ und $3 <_{lex} 9$.

Wie viele Vergleiche würden Sie benötigen, wenn sie die Folge S in Bezug auf $<_{lex}$ aufsteigend sortieren würden?

Fortsetzung

Aufgabe 2 (2+3+4 Punkte)

Für die folgenden Funktionen sei $x \in \mathbb{R}$.

- a) Seien $f : \mathbb{R} \rightarrow \mathbb{R}$, $g : \mathbb{R} \rightarrow \mathbb{R}$ zwei Funktionen definiert durch $f(x) = x^3$, $g(x) = 2^x$.
- a1) Bestimmen Sie ein beliebiges $k > 3$ für das $g(k) \geq f(k)$ gilt. Begründen Sie ihre Aussage.
- a2) Gilt $f \in \mathcal{O}(g)$? Beweisen Sie ihre Aussage.
- b) Sei $f : \mathbb{R} \rightarrow \mathbb{R}$ mit $f(x) = \sqrt[3]{x}$ und $g : \mathbb{R} \rightarrow \mathbb{R}$ mit $g(x) = \ln(x) + \frac{2}{x}$.
- Entscheiden Sie, ob $f \in \mathcal{O}(g)$ oder $g \in \mathcal{O}(f)$. Beweisen Sie Ihre Behauptung.

Fortsetzung

Aufgabe 3 (1+1+2+5+1 Punkte)

Betrachten Sie die folgende C-Funktion.

```
int tuwas(int n)
{
    int i, res = 0;

    if(n != 0)
    {
        for(i=0; i<n; i++)
        {
            res = res+(2*i);
        }
        res = res+tuwas(n-1);
    }
    return res;
}
```

- a) Bestimmen Sie den Rückgabewert für die Eingaben $n = 1, n = 2, n = 3$.
- b) Bestimmen Sie das kleinste $k \in \mathbb{N}$, so dass die *Laufzeitkomplexität* von `tuwas()` in $\mathcal{O}(n^k)$ ist. Sie können davon ausgehen, dass n nicht negativ ist. Begründen Sie Ihre Antwort.
- c) Was passiert, wenn n negativ ist?

Aufgabe 4 (3+4+3 Punkte)

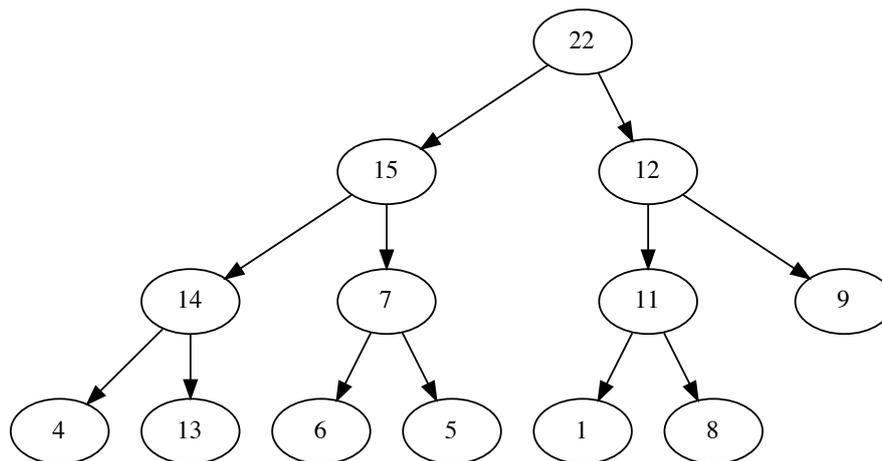
a) Betrachten Sie einen binären Max-Heap mit 123 Elementen.

a1) Wie tief ist der Heap als Baum? D.h. wie viele Ebenen hat der Heap? Begründen Sie ihre Aussage!

a2) Wie viele Knoten hat der Heap auf der letzten (tiefsten) Ebene? Begründen Sie ihre Aussage!

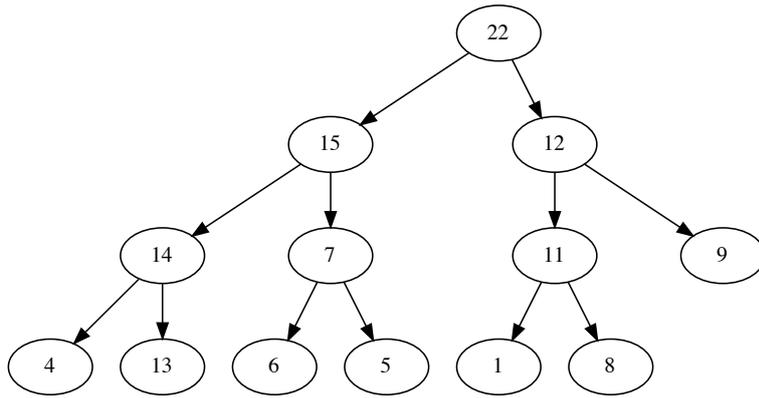
b) Betrachten Sie den Max-Heap H unten auf der Seite. Fügen Sie nacheinander die Elemente 3, 20 und 42 in den Heap ein und stellen Sie jeweils die Heap-Eigenschaft mit dem in der Vorlesung gezeigten Verfahren wieder her. Zeichnen Sie den Heap nach jeder Einfügeoperation!

c) Betrachten Sie wieder den Original-Heap H (nicht das Ergebnis von Teil b!). Extrahieren Sie das größte Element des Heaps und stellen Sie die Heap-Eigenschaft wieder her. Zeichnen Sie das Ergebnis.

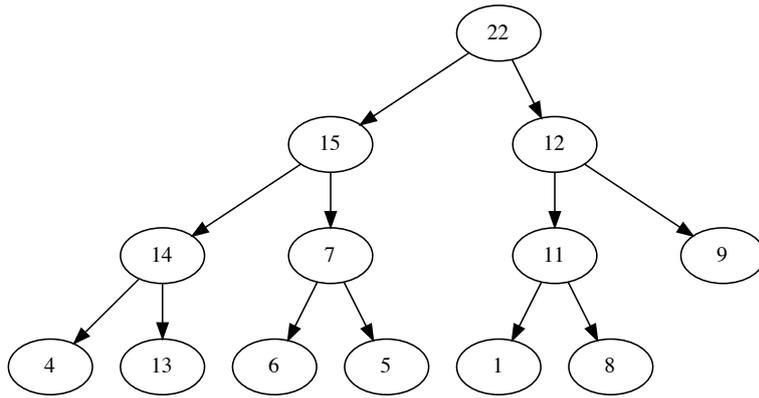


Heap H (auf den nächsten Seiten wiederholt)

Fortsetzung



Fortsetzung

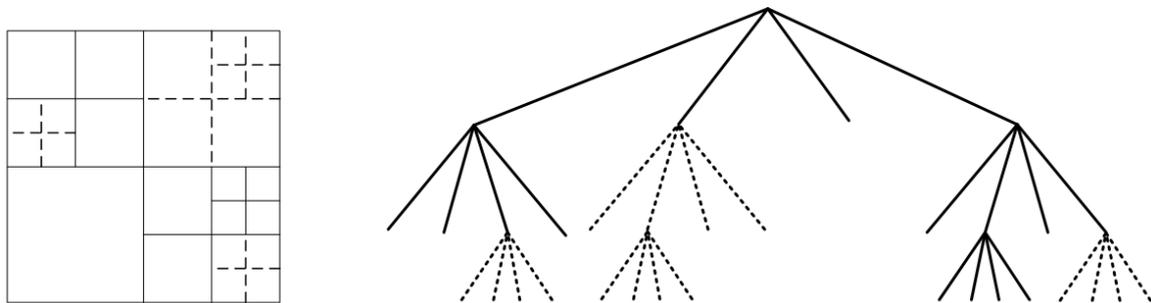


Aufgabe 5 (2+4+4 Punkte)

a) Geben Sie für die Rekurrenzrelation G eine \mathcal{O} -Lösung an.

$$G(n) = 8 \cdot G\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 3n^4 \quad \text{für } n \in \mathbb{N}$$

b) Ein *QuadTree* ist eine Datenstruktur zum schnellen Finden von Objekten anhand von kartesischen Koordinaten in der Ebene (also anhand der Position in X,Y). Jeder Knoten des QuadTrees repräsentiert ein Quadrat in der Ebene. Der Wurzelknoten repräsentiert den gesamte relevanten Ausschnitt der Ebene. In einem Knoten werden - direkt oder indirekt - alle Objekte gespeichert, die sich in diesem Quadrat befinden. Wenn die Anzahl der direkt in einem Quadrat gespeicherten Objekte zu groß wird, so wird das Quadrat in 4 Teilquadrate mit jeweils halber Seitenlänge gespalten, und nur diese direkt im Knoten gespeichert, während die Objekte selbst in den Knoten zum dem passenden Unterquadrat verlagert werden. Ein Quadrat enthält also immer entweder nur genau 4 Unterquadrate oder bis zu k Objekte in einer linearen Liste. Sie können im folgenden davon ausgehen, dass der gesamte QuadTree n Objekte enthält, und dass die gespeicherten Objekte gleichmäßig in der Ebene verteilt sind.



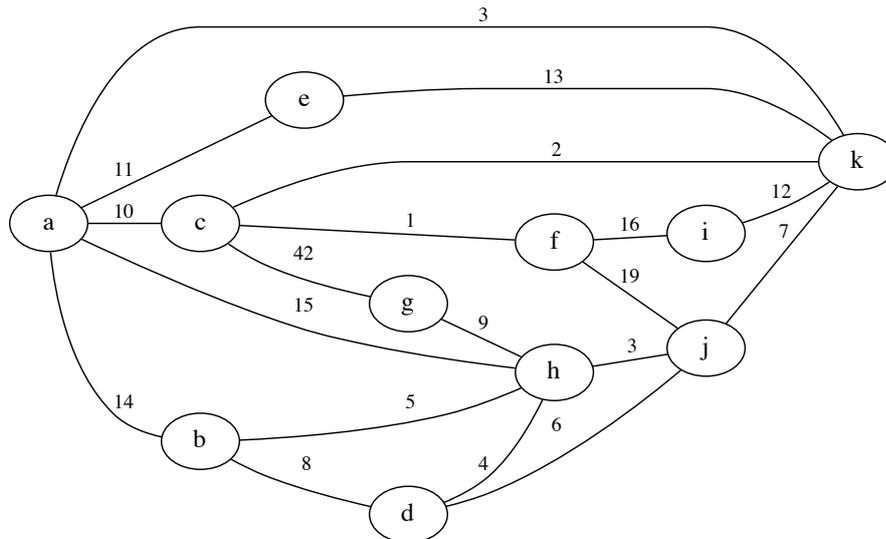
Beispiel für einen QuadTree (©Sheng-Hsiang Chang wikimedia commons)

- b1) Bestimmen Sie (im Sinne der \mathcal{O} -Notation) die Komplexität der Operation $\text{find}(\text{tree}, x, y)$, die zu einem Satz Koordinaten herausfindet, ob ein Objekt an diesen Koordinaten existiert. Sie können davon ausgehen, dass die Suche nach einem Objekt in einem Blatt-Quadrat Komplexität $\mathcal{O}(1)$ hat (es sind ja höchstens k Objekte zu prüfen).
- b2) Bestimmen Sie (im Sinne der \mathcal{O} -Notation) die Komplexität der Operation $\text{collect}(\text{tree})$, die eine Liste aller gespeicherten Objekte erzeugt. Sie können davon ausgehen, dass das Zusammenfassen von Teilergebnissen mit insgesamt n Objekten Kosten $\mathcal{O}(n)$ hat.

Fortsetzung

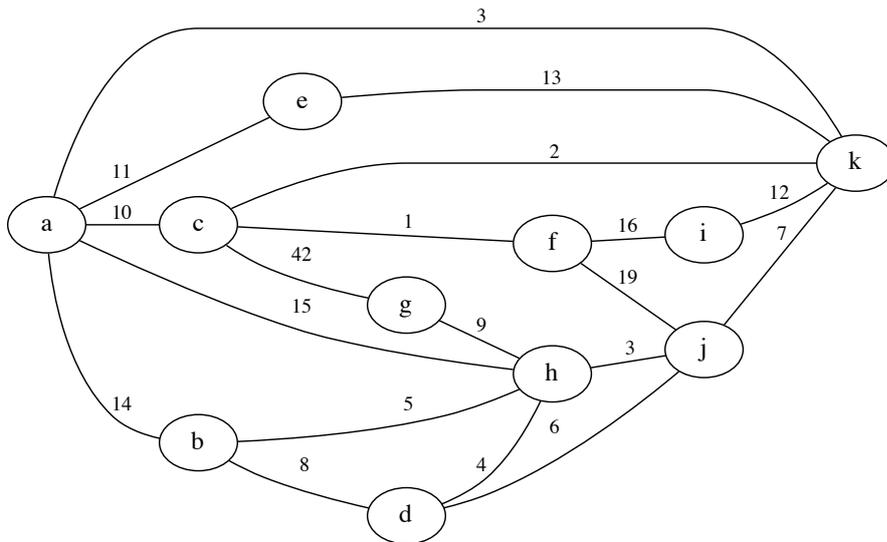
Aufgabe 6 (5+7 Punkte)

Gegeben sei der folgende Graph G .

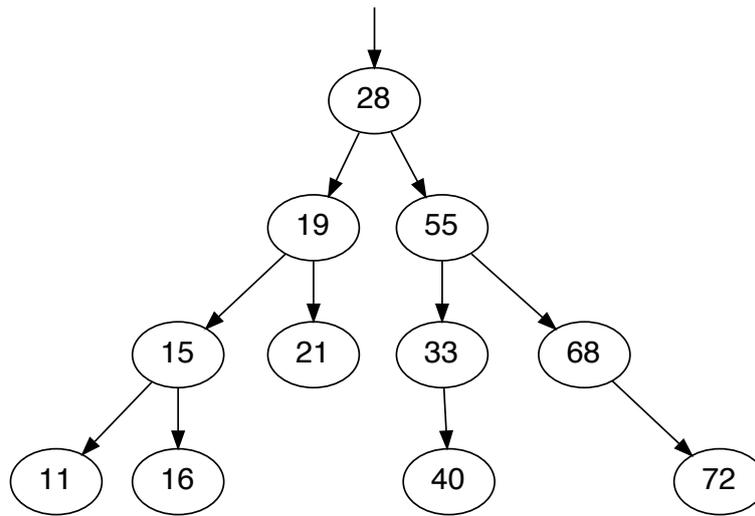


- Bestimmen Sie für G ausgehend vom Knoten g einen minimalen Spannbaum mit Hilfe des Prim-Algorithmus. Sie können hierzu die benutzten Kanten im Bild *sauber* markieren oder eine Liste der verwendeten Kanten angeben. Geben Sie die Reihenfolge an, in der Sie die Knoten dem Spannbaum hinzufügen. Wie hoch ist das Gesamtgewicht der Kanten des minimalen Spannbaums?
- Verwenden Sie den Algorithmus von Dijkstra, um die minimale Entfernung aller Knoten in G vom Knoten a zu bestimmen. Geben Sie die Knoten in der Reihenfolge an, in der sie im Algorithmus durchlaufen werden. Auf der nächsten Seite finden Sie eine Kopie des Graphen und eine Tabelle für das Ergebnis.

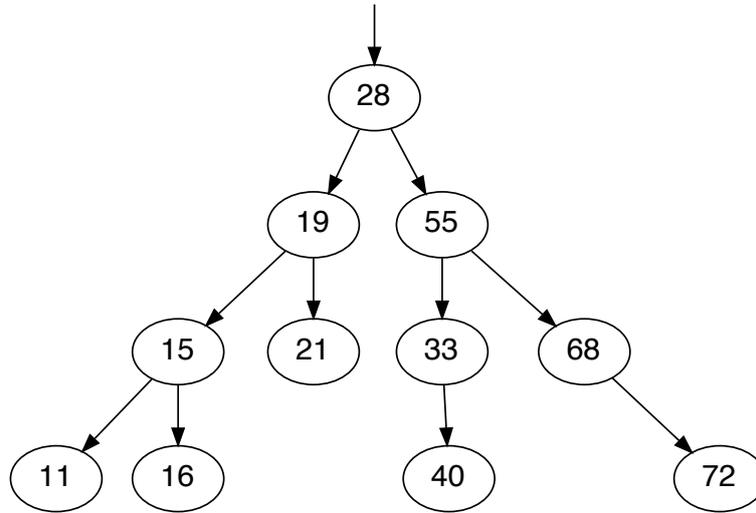
Fortsetzung



Fortsetzung

AVL Baum *A* (Einzufügen sind 75 (b2) und 17 (b3))

Fortsetzung

AVL Baum *A* (Einzufügen sind 75 (b2) und 17 (b3))