Grundzüge Algorithmen und Datenstrukturen (WS 2021/2022)

Lösung Probeklausur

Lösung 1. Aufgabe: (15 Punkte)

Die Reihenfolge lautet:

1000	$5\log n$	$4n$	$12n\log n$	n^4	4^n

- $1000 \in O(5\log n)$ direkter Beweis:
  Für $c = 200$ gilt $1000 \leq_{\text{fü}} c \times 5\log n = 1000\log n$, da $\log 3 > 1$ und log monoton wachsend ist ($n_0 = 3$).

- $5\log n \in o(4n)$ Beweis durch Grenzwertbetrachtung:

$$\lim_{n\to\infty} \frac{5\log n}{4n} = \frac{5}{4} \lim_{n\to\infty} \frac{\log n}{n}$$

  Nach L'Hospital gilt:

$$\frac{5}{4} \lim_{n\to\infty} \frac{\log n}{n} = \frac{5}{4} \lim_{n\to\infty} \frac{1/n}{1} = \frac{5}{4} \lim_{n\to\infty} \frac{1}{n} = 0$$

- $4n \in o(12n\log n)$ Beweis durch Grenzwertbetrachtung:

$$\lim_{n\to\infty} \frac{4n}{12n\log n} = \frac{1}{3} \lim_{n\to\infty} \frac{1}{\log n} = 0$$

- $12n\log n \in o(n^4)$ Beweis durch Grenzwertbetrachtung:

$$\lim_{n\to\infty} \frac{12n\log n}{n^4} = 12 \lim_{n\to\infty} \frac{\log n}{n^3}$$

  Nach L'Hospital gilt:

$$12 \lim_{n\to\infty} \frac{\log n}{n^3} = 12 \lim_{n\to\infty} \frac{1/n}{3n^2} = 4 \lim_{n\to\infty} \frac{1}{n^3} = 0$$

- $n^4 \in O(4^n)$ direkter Beweis:
  Für $c = 1$ gilt $n^4 \leq_{\text{fü}} c \times n^4 = n^4$, da für alle $n > n_0 = 5$ gilt dass $n^4 < 4^n$.
  Beweis durch Induktion über n:
  Induktionsanfang: Die Induktionsaussage gilt für $n = 5$ da $5^4 = 625 < 1024 = 4^5$.
  Induktionsschritt: $(n+1)^4 = n^4 + 4n^3 + 6n^2 + 4n + 1$. Da $n > 5$, ist $4n^3 < n^4$, $6n^2 < n^4$ und $4n + 1 < n^4$. In Summe ergibt sich $(n+1)^4 < 4n^4$. Nach Induktionsannahme gilt $4 \times n^4 < 4 \times 4^n = 4^{n+1}$.

Grundzüge Algorithmen und Datenstrukturen (WS 2021/2022)

Lösung Probeklausur

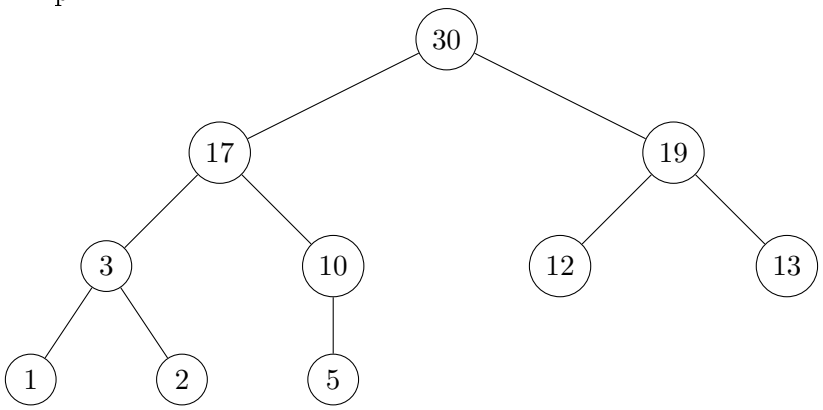
2. Aufgabe: (18 Punkte)

Zeichnen Sie für jede der folgenden potentiellen Inhalte des Feldes $A[1..10]$ den entsprechenden durch $A[]$ implizit dargestellten binären Baum. Klassifizieren Sie jeden der drei Fälle als Heap, Beinahe-Heap, oder nicht Heap. Verwenden Sie hierbei Max-Heaps. Illustrieren Sie für den Fall des Beinahe-Heaps eine Heapify Operation.

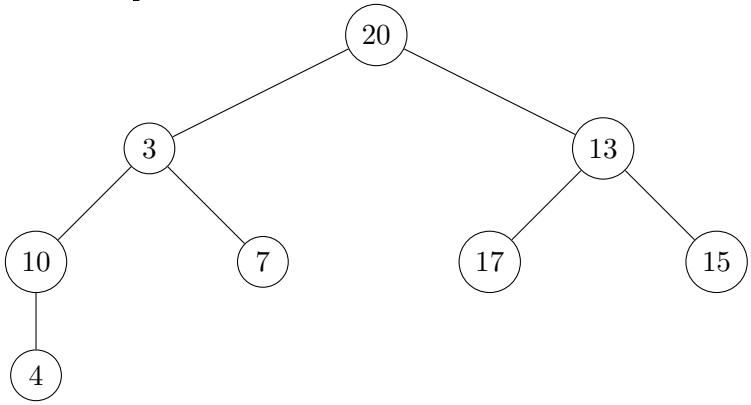
1. $\langle 30, 17, 19, 3, 10, 12, 13, 1, 2, 5 \rangle$
2. $\langle 20, 3, 13, 10, 7, 17, 15, 4 \rangle$
3. $\langle 5, 19, 14, 13, 7, 12, 8, 3, 4, 6 \rangle$

Lösung

1. Heap



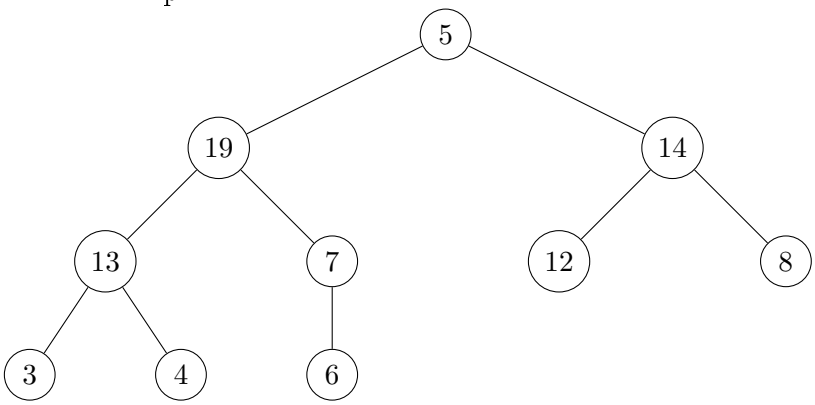
2. Nicht Heap



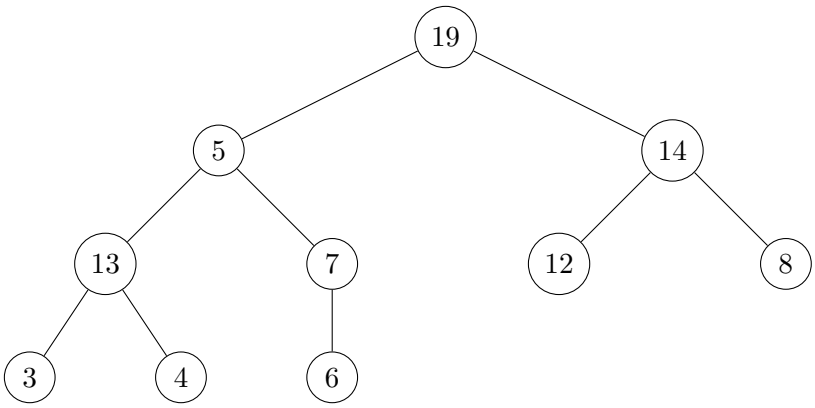
Grundzüge Algorithmen und Datenstrukturen (WS 2021/2022)

Lösung Probeklausur

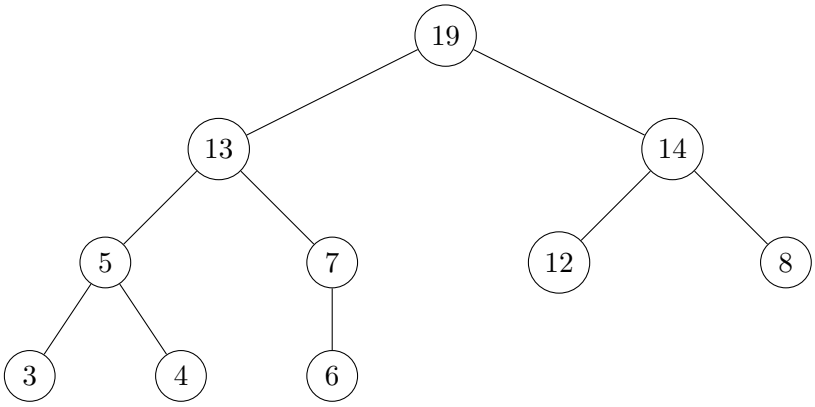
3. Beinahe Heap



Nach einem Schritt von `heapify`:



Nach dem zweiten und finalen Schritt von `heapify`:



Grundzüge Algorithmen und Datenstrukturen (WS 2021/2022)

Lösung Probeklausur

3. Aufgabe: (12 Punkte) Es seien $A[]$ und $B[]$ zwei Felder, die jeweils n Schlüssel aus einem geordneten Schlüsseluniversum enthalten. Die Einträge in den beiden Feldern sind nicht notwendigerweise alle verschieden.

Entwerfen Sie ein Programm, das ein Feld $C[]$ erzeugt, dass genau jene Schlüssel enthält, die sowohl in $A[]$ wie auch in $B[]$ vorkommen. In $C[]$ soll es keine Duplikate geben.

Was ist die Laufzeit Ihres Programms (in Abhängigkeit von n) und warum?

Lösung

Algorithmus: Man sortiere A und B durch jene Variante von Mergesort die beim Mergen Duplikate verwirft und damit die Liste von allen Duplikaten befreit. Anschließend überprüft man für jedes Element in B der Reihe nach durch binäre Suche, ob es in A enthalten ist. Ist dies der Fall, so fügt man es C hinzu.

Korrektheit: Offensichtlich sind alle Elemente in C sowohl in A als auch in B, jedes Element aus B stammt und in A gefunden wurde. Gleichzeitig werden alle (einzigartigen) Elemente in B betrachtet (und in A zum Vergleich herangezogen). Da B von Duplikaten befreit wurde ist auch C frei von Duplikaten.

Laufzeit: Das Sortieren der beiden Listen hat jeweils eine Laufzeit in $\mathcal{O}(n \log n)$ (Mergesort). Jede binäre Suche hat eine Laufzeit in $\mathcal{O}(\log n)$ und somit haben n vielen Suchen (für jedes Element in B) eine Laufzeit in $\mathcal{O}(n \log n)$. Insgesamt ist die Laufzeit also in $\mathcal{O}(n \log n)$.

Grundzüge Algorithmen und Datenstrukturen (WS 2021/2022)

Lösung Probeklausur

4. Aufgabe: (15 Punkte) Es sei X eine Menge mit n (verschiedenen) Schlüsseln aus einem geordneten Schlüsseluniversum U. Wir nennen einen Schlüssel $z \in U$ einen 2/3-Splitter für X, wenn

$$|\{x \in X : x \leq z\}| \leq 2n/3 \qquad \text{und} \qquad |\{x \in X : x \geq z\}| \leq 2n/3\,.$$

Anders ausgedrückt, mindestens ein Drittel der Schlüssel in X müssen kleiner als z sein und mindestens ein Drittel muss größer sein.

Entwickeln Sie einen Algorithmus, der für ein gegebenes Paar A, B von Mengen mit jeweils n verschiedenen Schlüsseln entscheidet, ob es einen Schllüssel z gibt, der sowohl für A wie auch für B ein 2/3-Splliter ist.

Was ist die Laufzeit Ihres Algorithmus und warum? Idealerweise sollte sie $O(n)$ sein.

Lösung

Die 2/3-Splitter für eine Menge sind genau jene Schlüssel die größer oder gleich dem $\lfloor 1/3 \rfloor$-ten Schlüssel nach Rang aber gleichzeitig kleiner oder gleich dem $\lceil 2/3 \rceil$-ten Schlüssel nach Rang sind. Es reicht also aus, für beide Mengen diese beiden Intervallgrenzen zu finden und zu überprüfen ob die Intervalle der 2/3-Splitter für beide Mengen überlappen.

Auswählen nach Rang ist in $\mathcal{O}(n)$ möglich. Ob die Intervalle überlappen lässt sich dann mit konstant vielen Vergleichen überprüfen. Also ist die Gesamtlaufzeit in $\mathcal{O}(n)$.

Grundzüge Algorithmen und Datenstrukturen (WS 2021/2022)

Lösung Probeklausur

Lösung 5. Aufgabe

Die Kosten des ersten Schnitts sind unabhängig von der Schnittstelle für jedes Stück L^2. Kennt man die optimalen Kosten für das Zerschneiden aller möglichen Teilstücke, so lassen sich die optimalen Gesamtkosten für das Zerschneiden eines Stückes einfach dadurch ermitteln alle $n-1$ möglichen Schnitte durchzugehen und jenen auszuwählen bei dem die Summe der optimalen Gesamtkosten für das Zerschneiden der beiden Teilstücke minimal ist. Die optimalen Kosten für die Teilstücke lassen sich rekursiv ermitteln.

Dieser rekursive Algorithmus lässt sich durch dynamische Programmierung, durch das Speichern der Zwischenergebnisse, verbessern.

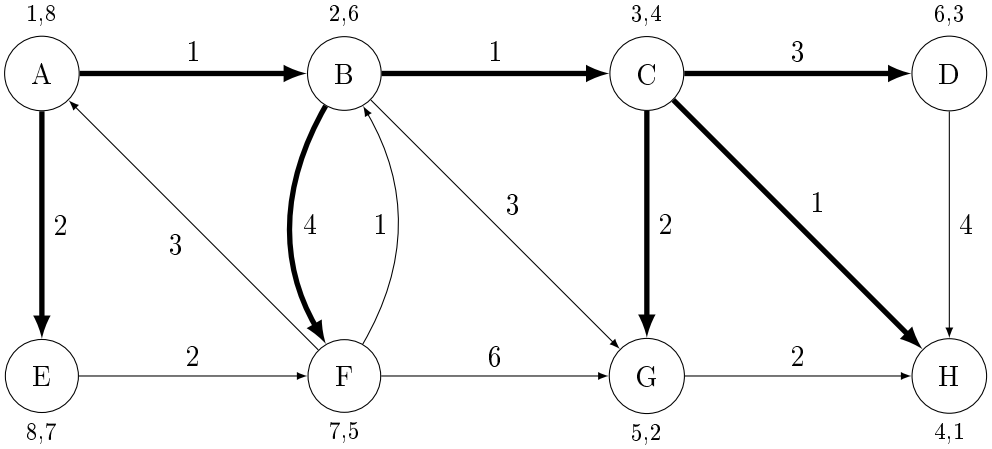
```
cutting(x) // Annahme x = [0, x_1, x_2, ..., x_{n-1}, L]
      global S ← n × n array
      return cut(x, 0, n)

cut(x, i, j) // Kosten für das Zerschneiden des Stückes zwischen x_i und x_j
      if j - i = 1 then
            return 0
      if S[i, j] = ⊥ then
            min_cost ← ∞
            for k = i + 1 to j - 1 do
                  min_cost ← min(min_cost, cut(x, i, k) + cut(x, k, j))
            S[i, j] ← min_cost + (x[j] − x[i])^2
      return S[i, j]
```

Es gibt $\mathcal{O}(n^2)$ viele mögliche Teilstücke und damit höchstens $\mathcal{O}(n^2)$ definierende Aufrufe von `cut`. Die definierende Aufrufe erzeugen jeweils Kosten in $\mathcal{O}(n)$. Jeder definierende Aufruf erzeugt höchstens $\mathcal{O}(n)$ viele Aufrufe, wodurch die Anzahl aller Aufrufe und damit die der nicht-definierenden Aufrufe auf $\mathcal{O}(n^3)$ beschränkt ist. Jeder nicht-definierende Aufruf erzeugt nur konstante Kosten. Die Gesamtlaufzeit ist also in $\mathcal{O}(n^3)$.

Grundzüge Algorithmen und Datenstrukturen (WS 2021/2022)

Lösung Probeklausur

Lösung 6. Aufgabe

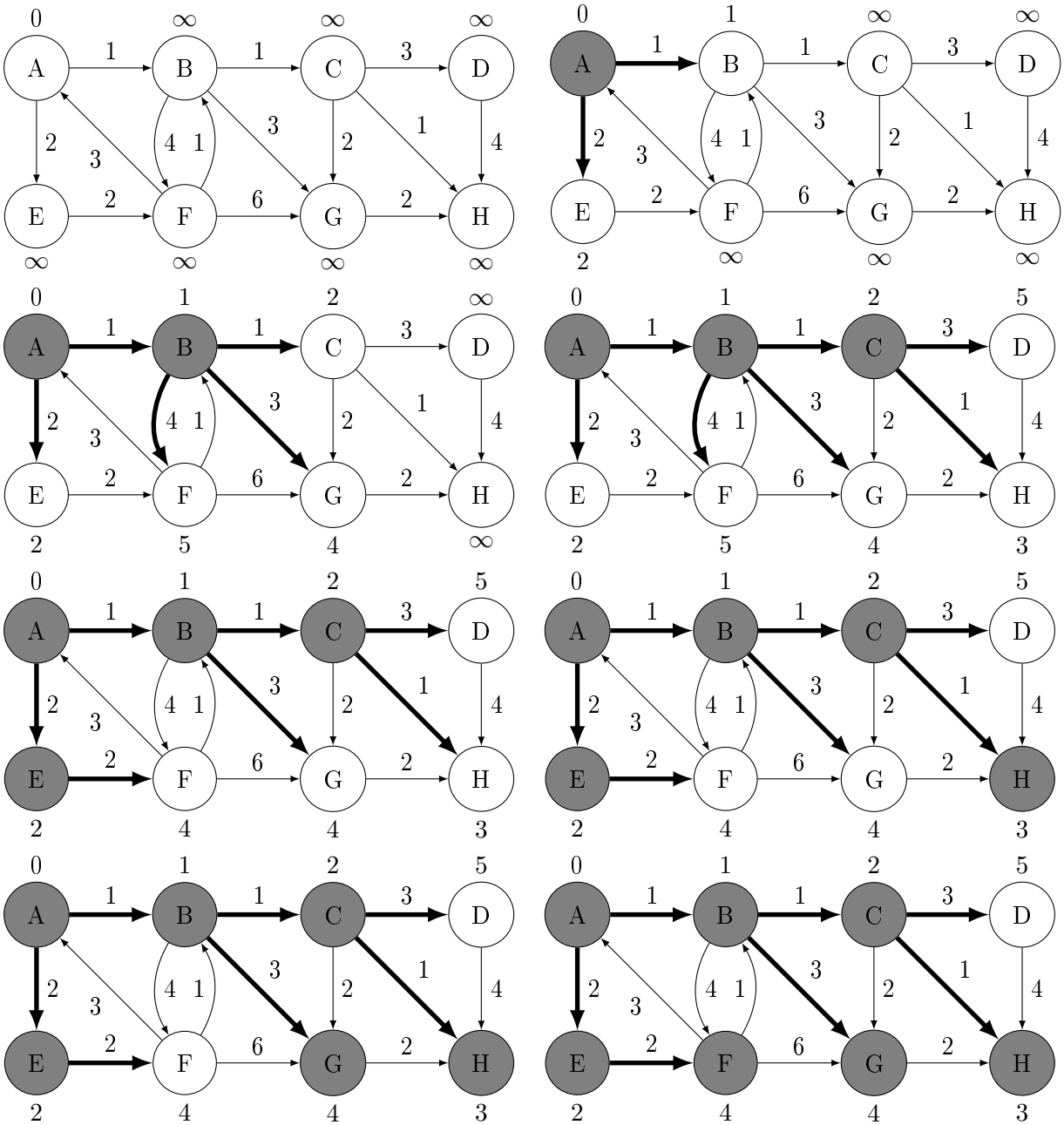


Grundzüge Algorithmen und Datenstrukturen (WS 2021/2022)

Lösung Probeklausur

Lösung 7. Aufgabe

Genutzt wird der Algorithmus von Dijkstra. Markierte Knoten sind fertig bearbeitet. Markierte Kante verbinden Knoten mit ihrem gespeicherten Elter und bilden am Ende den kürzeste-Wege-Baum. Für jeden Knoten ist der gespeicherte δ-Wert angegeben.



Grundzüge Algorithmen und Datenstrukturen (WS 2021/2022)

Lösung Probeklausur

8. Aufgabe: (15 Punkte)

Hier wird der Algorithmus von Kruskal angewandt, d.h. in jedem Schritt wird eine minimale Kante ausgewählt die mit den bereits ausgewählten Kanten keinen Kreis bildet.