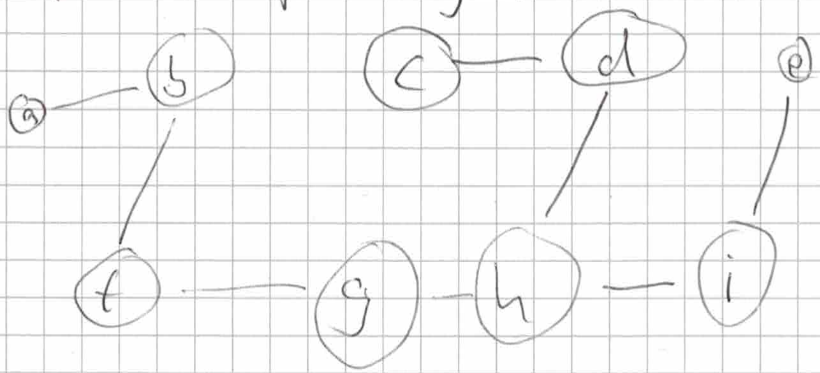
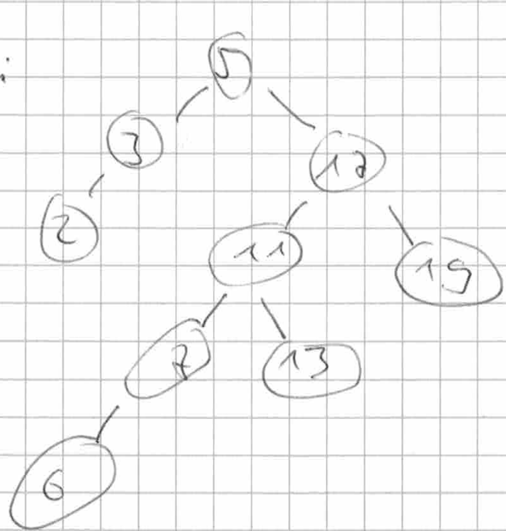
Mats Radloff 6946325

1)

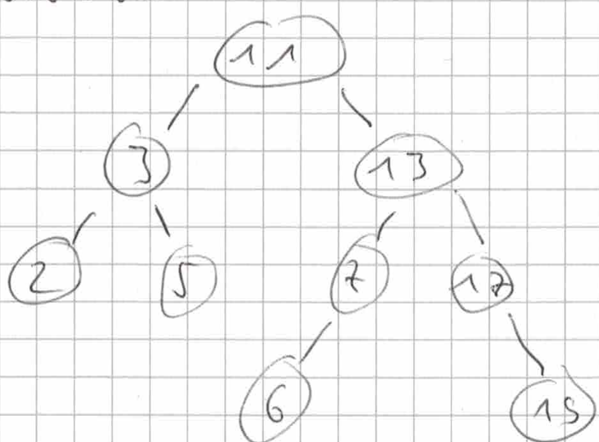
a -> b -> f -> g -> h -> d -> c -> i -> e



2) Einfügen:



Linksrotation:



Matz Radloff 6946325

3)

a) $f(n) = o(g(n))$

$\Leftrightarrow \limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| \overset{!}{=} 0$

$= \limsup_{n\to\infty} \left|\frac{n(\log n)^{42}}{n^{1,23}}\right| = \left|\frac{(\log n)^{42}}{n^{0,23}}\right| \to \infty$

$\to f(n) \neq o(g(n))$

$f(n) = O(g(n))$

$\Leftrightarrow \limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| < \infty \neq \infty \to f(n) \neq O(g(n))$

$\cancel{\Rightarrow \limsup_{n\to\infty}} |$

$f(n) = \Omega(g(n))$

$\Leftrightarrow \limsup_{n\to\infty} \left|\frac{g(n)}{f(n)}\right| < \infty$

$= \quad 0 < \infty \to f(n) = \Omega(g(n))$

$f(n) = \omega(g(n))$

$\Leftrightarrow \limsup_{n\to\infty} \left|\frac{g(n)}{f(n)}\right| \overset{!}{=} 0$

$= \quad 0 = 0 \to f(n) = \omega(g(n))$

2

Matz Radloff 6346325

3)

b) $\limsup_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| = \limsup_{n\to\infty} \left|\frac{n \cdot |\sin n|}{\sqrt{n}}\right|$

$= \limsup_{n\to\infty} \left|\frac{n^2 \cdot \sin^2 n}{n}\right|$

$= \limsup_{n\to\infty} \left| n \cdot \sin^2 n \right| \to \infty$

$\to f(n) \notin o(g(n))$

$f(n) \notin O(g(n))$

$f(n) = \Omega(g(n))$

$f(n) = \omega(g(n))$

c) $\limsup_{n\to\infty} \left|\frac{n^2 + n^{\frac{3}{2}} + n\log n}{\binom{n}{2}}\right|$

$= \limsup_{n\to\infty} \left|\frac{(n^2 + n^{\frac{3}{2}} + n\log n) \cdot 2 \cdot (n-2)!}{n!}\right|$

Da $(n-2)! \cdot n^2 \gtrsim n! < (n^2 + n^{\frac{3}{2}} + n\log n) \cdot 2 \cdot (n-2)!$

$\to \infty$

$\to f(n) \notin o(g(n))$

$f(n) \notin O(g(n))$

$f(n) = \Omega(g(n))$

$f(n) = \omega(g(n))$

Max Radloff 6846325

3)
d)

$$\limsup_{n\to\infty} \left| \frac{8^{\log n}}{n^3 \cdot \log n} \right|$$

$$= \limsup_{n\to\infty} \left| \frac{n \cdot \cancel{n}}{n^{\cancel{3}2} \cdot \log n} \right| \to 0$$

$\to f(n) = o(g(n))$

$f(n) = O(g(n))$

$f(n) \neq \Omega(g(n))$

$f(n) \neq \omega(g(n))$

4)
a) $\text{MakeSet}(a) = \{a\}$

$\text{MakeSet}(b) = \{b\}$

$\text{MakeSet}(c) = \{c\}$

$\text{Union}(\{b\},\{c\}) = \{b,c\}$

$\text{Makeset}(d) = \{d\}$

$\text{Union}(\{b,c\},\{d\}) = \{b,c,d)$

$\text{Makeset}(e) = \{e\}$

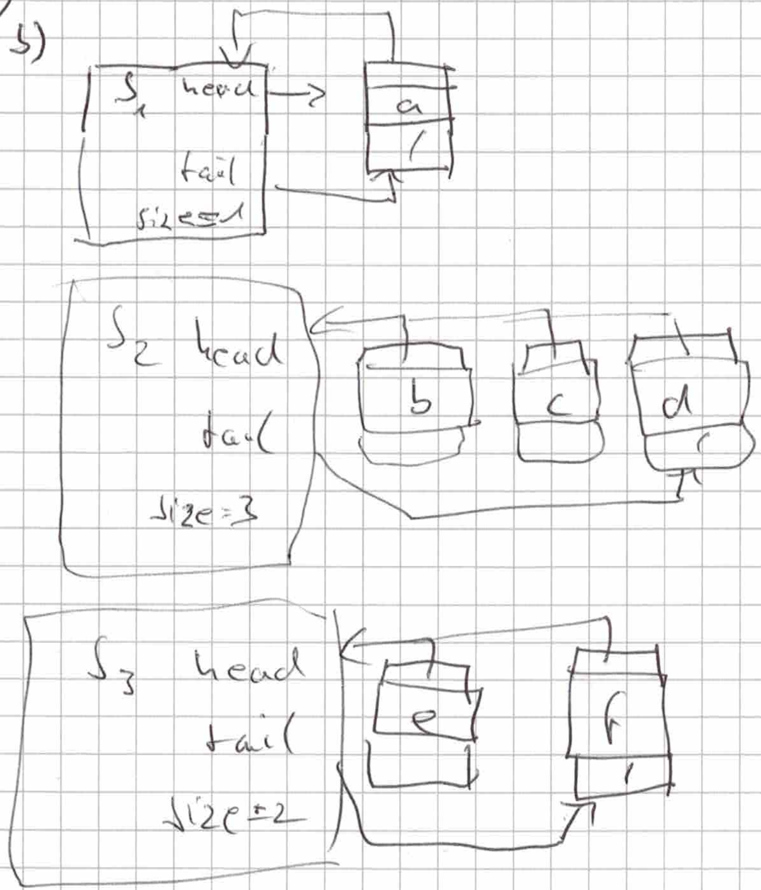
$\text{Makeset}(f) = \{f\}$

$\text{Union}(\{e\},\{f\}) = \{e,f\}$

Mats Radloff 6946325

a)

b)

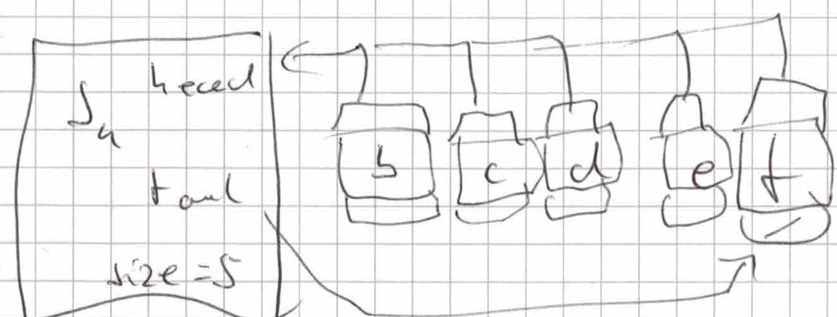


c) FindSet(a) = a

FindSet(d) = b

d)

S_4 = Union(c, f)



Mats Radloff 6546325

5

a) $$T(n) = \begin{cases} \Theta(1) & , n \le 2 \\ 2T(\lceil \frac{n}{2} \rceil) + n & , n > 2 \end{cases}$$

b) $$T(n) = 2T\left(\lceil \tfrac{n}{2} \rceil\right) + n$$

$$= 2 \cdot \left(2T\left(\lceil \tfrac{n}{2 \cdot 2} \rceil\right) + n-1\right) + n$$

$$= 2^k \left(T\left(\lceil \tfrac{n}{2^k} \rceil\right) + (n-k)\right)$$

Für $k = \log_2 n$

$$= n \cdot (n - \log_2 n) = n^2 - n \log n = O(n^2)$$

6) Als Datenstruktur bietet sich ein normaler Binärbaum an. Für das Einfügen und Löschen sind im worst-case ~~$O(n)$~~ h-Operationen notwendig. Da ein Binärbaum $h = \log n$ besitzt, ~~wird~~ ist die Laufzeit $O(\log n)$ gegeben.

Der Binärbaum speichert zusätzlich die Anzahl der jeweils linken und rechten Kinder, damit Rang(x) auch in $O(\log n)$ läuft. Andernfalls müsste im worst-case x = größtes Element der gesamte Baum abgegangen werden. So müssen ~~nur~~ nur die gespeicherten Werte entlang des Suchpfads zusammengerechnet werden.

Malte Radloff 694 6325

7)

a) $$OPT(k) = \begin{cases} w_0 & , k=0 \\ OPT(k-1) + \left(w_k \cdot \left\lfloor \frac{\cos(\pi k)+1}{2} \right\rfloor\right) & , k>0 \end{cases}$$

Durch $OPT(k-1)$ werden für alle $k \in \{1,2,\dots,k\}$ die Zwischenergebnisse zusammengezählt bis $k=0$ erreicht wird. Da nur alle 2 Kilometer eine Werbetafel aufgestellt werden darf stellt $\left\lfloor \frac{\cos(\pi k)+1}{2} \right\rfloor$ sicher, dass nur bei geraden k, w_k addiert wird.

$$\frac{\cos(\pi \cdot 1)+1}{2} = 0$$

$$\frac{\cos(2\pi)+1}{2} = 1$$

$$\frac{\cos(3\pi)+1}{2} = 0$$

$$\vdots$$

b)

```
1: Gewinn(W, n):
2:     W_Bis_I ← Array(n) = {0} // Gewinn bis W_i zwischenspeichern
                                 // mit 0 initialisieren
3:     for i ← 1 to n
4:         if W[i-1] == 0
5:             W_Bis_I[i] = max(W[i], W[i+1])
6:             if W[i+1] > W[i]: i++
7: return W_Bis_I[n]
```

Zeile 4 stellt sicher, dass nicht 2 Kilometer hintereinander verwendet werden. Zeile 5 zählt den aktuellen oder nächsten Kilometer hinzu und inkrementiert i falls der nächste größer ist.

Matz Radloff 684 6325

8) Angenommen die Greedy Lösung sei nicht optimal, dann würde es einen Abschnitt e_i geben, der größer als d_i für gleiche i ist. Alle d_i, außer dem letzten Abstand sind schon $= D$, haben also die maximale Länge. Vergrößert man den letzten Abstand, würde man entweder über das Ziel hinausschießen oder müsste einen vorherigen Abstand verringern, was die Anzahl der Stopps nicht verringert. Da $\sum_{i=0} d_i = const.$ und nur das letzte d vergrößert werden kann, ist die Greedy-Lösung optimal

Max Rodloff 6846725

5)

a) Turing-Maschine:

Nehme eine Teilmenge $W \subseteq V$

Wenn $|W| \neq k$, return false

Für alle $u, v \in W$, wenn $(u,v) \in E$, return false

ansonsten, return true

→ Entscheidet in polynomieller Zeit

Da es auch NP-hart ist, ist UnabhängigeMenge NP-vollständig