Algorithmentheorie
Präsenzübung 3 Tutorium

23.11.23

# Nochmal zur Abgabe Aufgabe 2.2

b) $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \sqrt{n}$

$T(n) = \Theta(n)$

Beh.: $T(n) \leq cn - d$

$$T(n) \overset{IV}{\leq} 2 \cdot \left(c\frac{n}{2} - d\right) + \sqrt{n} \leq \ldots \leq \ldots$$

wähle $d = \sqrt{n}$ → darf nicht gemacht werden, da $d$ eine Konstante sein muss, was es nicht ist, wenn es von $n$ abhängt

Wenn $d$ von $n$ abhängt muss aufgestellt werden:

Beh. $T(n) \leq c \cdot n - d(n)$

$$T(n) \overset{IV}{\leq} 2 \cdot \left(c\frac{n}{2} - d\left(\frac{n}{2}\right)\right) + \sqrt{n} \leq \ldots \leq \ldots$$

wähle $d(n) = \sqrt{n}$

↳ ist erlaubt.

## (3.1) a) Handshake Lemma

Zeige: $\sum_{v \in V} \deg(v) = 2|E|$

formaler Beweis:

Sei $\delta : V \times E \longrightarrow \{0,1\}$

definiert durch:

$$\delta(v,e) = \begin{cases} 1 & v \text{ ist inzident zu } e \\ 0 & \text{sonst} \end{cases}$$

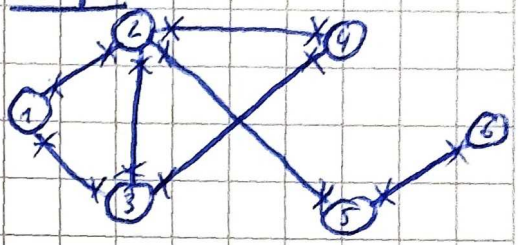
Dann ist $\sum_{v \in V} \deg(v) = \sum_{v \in V} \sum_{e \in E} \delta(v,e)$

addition ist kommutativ $= \sum_{e \in E} \sum_{v \in V} \delta(v,e)$

da jede Kante zwei Knoten hat $= \sum_{e \in E} 2 = 2 \cdot |E|$

↳ für jede Kante sind für zwei Knoten $\delta(v,e) = 1$, für den Rest 0, da eine Kante nur zwei Knoten haben kann. Daher ist $\sum_{v \in V} \delta(v,e)$ hier zwei.

Beispiel




$\Sigma = 2+4+3+2+2+1$

$= 14$

visuell: Auf jeder Kante sind zwei Kreuze, somit $\Sigma \deg(v) = 2|E|$

b) Zeige: Anzahl Knoten mit ungeradem Grad ist gerade.

Wir haben:

$$2 \cdot |E| \overset{a)}{=} \sum_{v \in V} \deg(v) = \sum_{v \in V} \deg(v) + \sum_{v \in V} \deg(v)$$

$2 \cdot |E|$: gerade

$\sum_{v \in V} \deg(v)$: $\deg(v)$ gerade → gerade

$\sum_{v \in V} \deg(v)$: $\deg(v)$ ungerade → muss auch gerade sein, da gerade + ungerade nicht gerade sein kann.

Also ist $\sum_{v \in V, \deg(v) \text{ ungerade}} \deg(v)$ ist gerade.

Da jeder Summand ungerade ist, muss die Anzahl der Summanden gerade sein. Also ist die Anzahl Knoten mit ungeradem Grad gerade.

(3.2) Auf Basis der Breitensuche

Eingabe: $G = (V, E)$

Frage: Ist $G$ kreisfrei (also ein Wald/Baum)?

Zeilen

1 $U := \emptyset$

2 $pred(v) := \perp$ für alle $v \in V$

3 while $U \neq V$ do:

4 wähle $s \in V \setminus U$

5 $pred(s) := s$; $Q := (s)$ Schlange; $U := U \cup \{s\}$

6 while $Q \neq \emptyset$ do:

7 $v := Dequeue(Q)$

8 for $u \in N(v)$ do:

9 if $u \notin U$ then

10 $Enqueue(Q, u)$

11 $U := U \cup \{u\}$

12 $pred(u) = v$

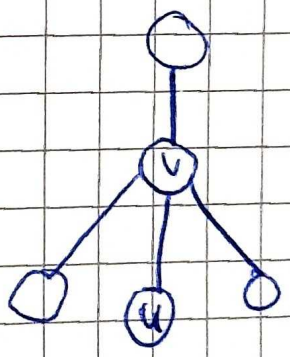
13 else

14 if $pred(v) \neq u$ return false

15 return true

pred: Vorgänger

→ wichtig falls Graph nicht komplett zusammenhängend ist



Laufzeit: Alle Zeilen in Zeit $O(1)$ bis auf Zeile 2; Zeile braucht Zeit $O(n)$. Jeder Knoten $v$ wird genau einmal zu $Q$ hinzugefügt. Für jeden Knoten brauchen wir Zeit $O(\deg(v))$ (beim Entfernen aus $Q$). Also insgesamt

$$O\left(n + \sum_{v \in V} \deg(v)\right) \overset{1.2)}{=} O(n + 2|E|)$$

$$= O(n+m)$$

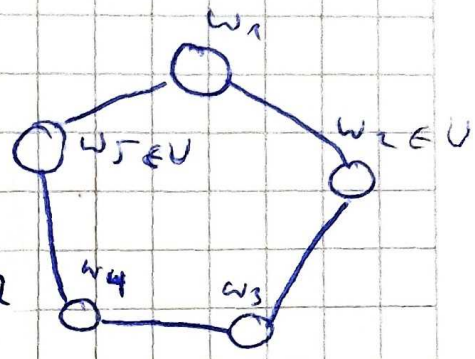
$\Rightarrow$ lineare Laufzeit.

Terminierung:
- innere while Schleife: jeder Knoten kann nur einmal zu $Q$ hinzugefügt werden. in jeder Iteration wird ein Knoten entfernt (Zeile 7).
- äußere Schleife: In jeder Iteration werden Knoten hinzugefügt zu $U$.

$\Rightarrow$ Terminierung folgt.

Zeige: Algorithmus gibt true aus, wenn G kreisfrei ist. (Alg gibt true aus $\Leftrightarrow$ G ist kreisfrei)

„$\Rightarrow$" Angenommen G hat einen Kreis $(w_1, w_2, w_3, \dots, w_k, w_1)$.

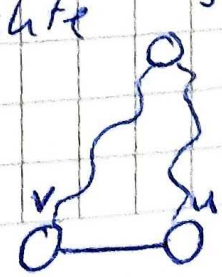



Wir betrachten den letzten Knoten in $w_1, \dots, w_k$, der aus Q entfernt wird. O.B.d.A. sei dies $w_1$.

Da $w_2, w_k$ bereits erkundet gilt, dass $w_2, w_k \in U$ bei Erkundung von $w_1$.

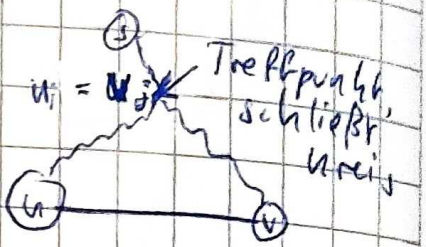
Nur ein Knoten von $w_2, w_k$ kann gleich $pred(w_1)$, also geben wir false zurück.

„$\Leftarrow$" Angenommen der Algorithmus gibt false aus mit Knoten $u, v$. Betrachte




$u = u_0, u_1, \dots, u_k = s$, wobei $u_{i+1} = pred(u_i)$, und $v = v_0, v_1, \dots, v_{l-1}$,

Kann auch




wobei $v_{i+1} = pred(v_i)$.

Sei $i, j$ minimal sodass

$u_i = v_j$. Dann ist

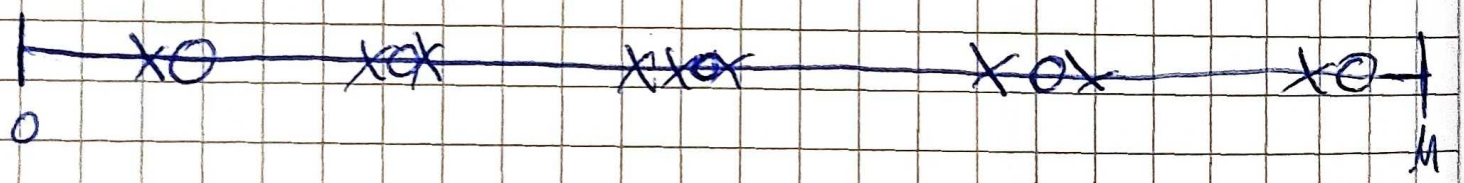
$(v = v_0, v_1 \ldots, v_i = u_j, u_{j-1}, \ldots, u_0 = u, v)$ ein Kreis.

(3.3)

a)

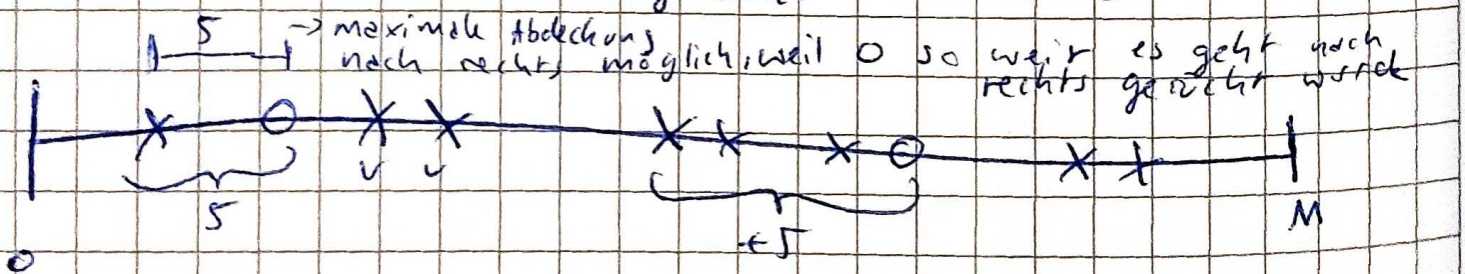
! für (3.3) b) und (3.4) wird es eine Musterlösung geben

x - steht für ~~alle~~ alle $x \in [0, M]$ $0 \in S$




Wie viele Kreise werden benötigt?

v: abgedeckt durch 0



current: das am weitesten rechts gesetzte $s$

$S = \emptyset$

current := $-\infty$

sortiere $x_1, \ldots, x_n$ aufsteigend

for $i = 1, \ldots, n$ do

  if current $\notin [x_i - 5, x_i + 5]$ then

    current := $\min(x_i + 5, M)$

    $S := S \cup \{\text{current}\}$

return $S$

Laufzeit: sortieren $O(n \log n)$ + Schleife $O(n)$

$\rightsquigarrow O(n \log(n))$

Terminierung: ✓

Korrektheit: Sei $S$ die vom Algorithmus berechnete Menge. $S$ ist eine Lösung, da für jedes $x_i$:

ein entsprechender (current) Punkt hinzugefügt wird.

Sei $S = \{s_1, \dots, s_k\}$, $s_1 < s_2 < \dots < s_k$.

Angenommen wir haben die Lösung $S' = \{s_1', s_2', \dots, s_l'\}$ mit $s_1' < s_2' < \dots < s_l'$

Behauptung: $s_i' \leq s_i$ für alle $i = 1, \dots, l$

I.A.: $i = 1$ Da $x_1$ abgedeckt sein muss durch $S$ gilt $s_1' \leq x_1 + 5 = s_1$

I.S.: Betrachte $s_i'$, $s_i$. Sei $x_j$ das Element für das $s_i$ eingefügt wurde. $x_j$ nicht durch $s_1, \dots, s_{i-1}$ abgedeckt, also auch nicht nach I.V. durch $s_1', \dots, s_{i-1}'$.
Also $s_i' \leq x_j + 5 = s_i$

Danach analog $l \geq k$.

Daher ist $S$ optimal.