

Algorithmen & Datenstrukturen

Woche 12

Marius Tomek, Nicolas Wehrli, Tim Rieder

12. Dezember 2022

ETH Zürich

Kurze Kommentare zur letzten Serie

Theory: Minimum Spanning Tree

Homework 11

Peergrading 11.2

Kurze Kommentare zur letzten Serie

Generell gut gelöst!

10.1)

- Beide Arten von DFS sind in Ordnung. (Knoten H im Graph oder nicht)

10.2)

- DP mit allen verlangten Punkten definieren (3. enthält computation **und** base case!)

10.3)

- Pseudo code so erklären, dass eure Idee verständlich ist
- Laufzeit **und** Korrektheits Analyse nicht vergessen!

Theory: Minimum Spanning Tree

Minimum Spanning Tree

zusammenhängender Teilgraph mit minimalem Gewicht

Sei $G = (V, E)$ zusammenhängend

aufspannende Kanten: $A \subseteq E$ wobei (V, A) zusammenhängend

Gewicht: $w(A) = \sum_{e \in A} w(e)$

aufspannende Kanten können gefunden werden indem Kante für Kante, die gewichtsminimale Kante zum Spannbaum hinzugefügt wird, sodass kein Kreis entsteht

Beobachtung: die kleinste Kante inzident zu einer ZHK im geg. Teilgraphen ist im Spannbaum enthalten: sog. sichere Kante

Füge sukzessive sichere Kanten von ZHKs hinzu

Algorithm 1 Boruvka ($G = (V, E)$)

- 1: $F \leftarrow \emptyset$ ▷ Set von sicheren kanten
 - 2: **while** F nicht spanning für V **do**
 - 3: $S \leftarrow \{ZHK_1, \dots, ZHK_k\}$
 - 4: $A \leftarrow \{e_1, \dots, e_k\}$ ▷ e_i : min Kante inzident zu ZHK_i
 - 5: $F \leftarrow F \cup A$
-

Läuft in $O((|V| + |E|) \cdot \log |V|)$

ZHKs finden mit DFS läuft in $O(|V| + |E|)$

Inzidente Kanten finden läuft ebenfalls in $O(|V| + |E|)$

Obere Schranke für ausgeführte Iterationen: $O(\log |V|)$

Ähnlich Boruvka, konzentriere auf eine ZHK

Algorithm 2 Prim ($G = (V, E)$)

- 1: $F \leftarrow \emptyset$
 - 2: $S \leftarrow \{s\}$ ▷ ZHK von s in F
 - 3: **while** F nicht spanning für V **do**
 - 4: $e = \{u, v\} \leftarrow \operatorname{argmin}_{e \in E, u \in S, v \notin S} w(e)$ ▷ min Kante an S ($u \in S, v \notin S$)
 - 5: $F \leftarrow F \cup \{e\}$
 - 6: $S \leftarrow S \cup \{v\}$
-

Wie schnell minimale Kante finden? (Idee: Dijkstra min-heap/priority queue)

Prim - priority queue (min-heap)

Algorithm 3 Prim ($G = (V, E)$)

```
1:  $\text{dis} \leftarrow [\infty, \infty, \dots, \infty]$ ;
2:  $H \leftarrow \emptyset$ ;  $\text{dis}[s] \leftarrow 0$ ;                                ▷ Initialize Priority Queue
3:  $H.\text{enqueue}(s, 0)$ 
4:  $S \leftarrow \{s\}$                                                 ▷ ZHK von  $s$  in  $F$ 
5: while  $H \neq \emptyset$  do
6:    $v \leftarrow H.\text{dequeue}()$ 
7:    $S \leftarrow S \cup \{v\}$ 
8:   for  $\{v, u\} \in E, u \notin S$  do
9:      $\text{dis}[u] \leftarrow \min\{\text{dis}[u], w(\{v, u\})\}$ 
10:     $H.\text{enqueue}(u, \text{dis}[u])$ 
```

Läuft nun in $O((|V| + |E|) \cdot \log |V|)$ wie Dijkstra und Boruvka

Algorithm 4 Kruskal ($G = (V, E)$)

```
1:  $F \leftarrow \emptyset$ 
2: for  $\{u, v\} \in E$ , aufsteigend sortiert do
3:   if  $u, v$  in unterschiedlichen ZHKs von  $F$  then
4:      $F \leftarrow F \cup \{\{u, v\}\}$ 
```

Homework 11

Peergrading 11.2
