negative Zyklen

wie testen ob existent?

Idee: wie Bellman-Ford aber eine zusätzliche Iteration "Schranken verbessern" (also n Iterationen)

Korrektheit:

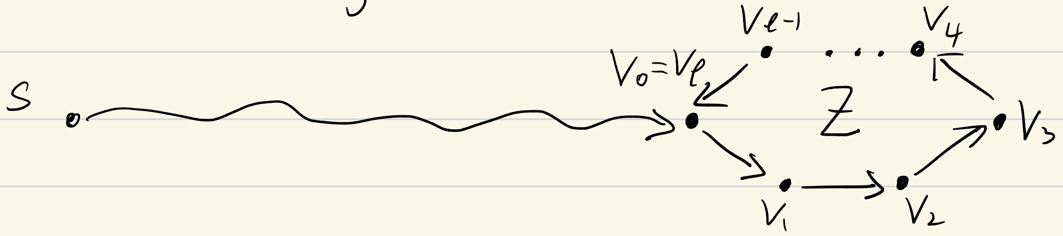
$d[\,]$: Schranken nach $n-1$ Iterationen

$d'[\,]$: Schranken nach n Iterationen

Behauptung s erreicht neg. Zyklus $\Leftrightarrow \exists v.\ d'[v] < d[v]$

Beweis: "$\Leftarrow$" folgt direkt ($\nexists$ neg. Zyklus $\Rightarrow d[v] = d(s,v)$)

"$\Rightarrow$": sei Z neg. Zyklus erreichbar von s



$\ell \geq 1$

$$d'[v_i] \leq d[v_{i-1}] + c(v_i, v_{i-1})$$ (nach "Schranken verbessern")

$$\Rightarrow \sum_{i=1}^{\ell} d'[v_i] \leq \sum_{i=1}^{\ell} d[v_{i-1}] + \underbrace{c(Z)}_{<0}$$

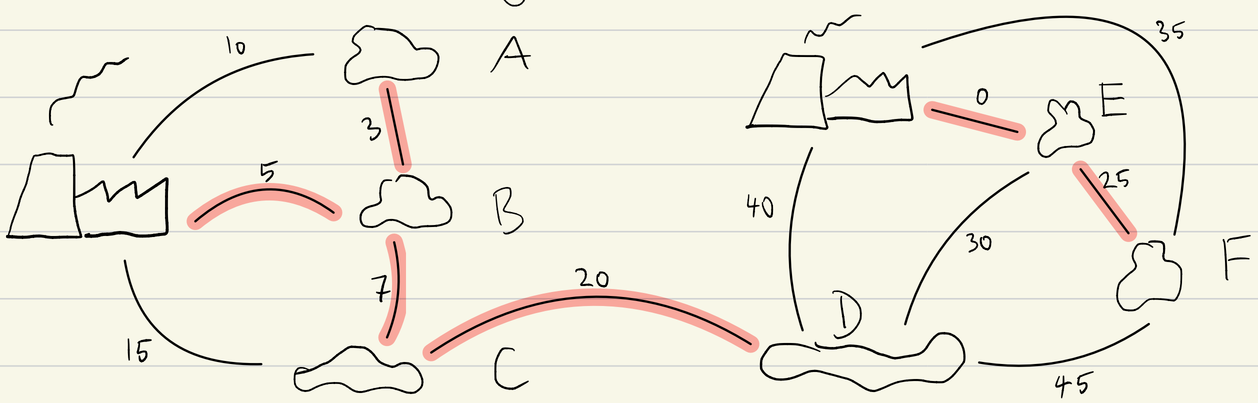
$$< \sum_{i=1}^{\ell} d[v_{i-1}] = \sum_{i=1}^{\ell} d[v_i]$$ (Z neg; $v_\ell = v_0$)

$$\Rightarrow \sum_{i=1}^{\ell} d'[v_i] < \sum_{i=1}^{\ell} d[v_i]$$

$$\Rightarrow \exists i \in \{1, \dots, \ell\}.\ d'[v_i] < d[v_i] \quad \boxtimes$$ (mindestens eine Schranke wurde strikt kleiner)

1

1926: Elektrifizierung Mährens (Region in Tschechien)

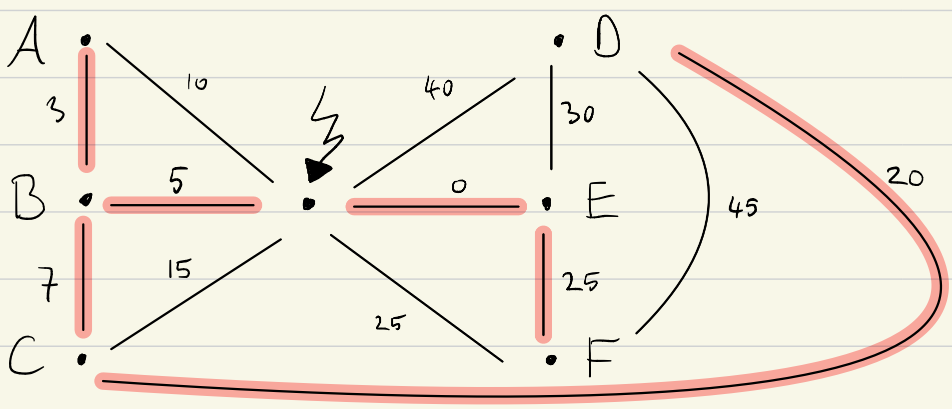


Frage an Otakar Boruvka (Mathematiker)

welche Stromleitungen bauen, so dass

- alle Städte mit Strom versorgt
- Gesamtkosten minimal

gewichteter Graph



Ziel: zusammenhängender Teilgraph mit minimalem Gewicht

Graph $G=(V,E)$ zusammenhängend

Kantengewichte: $\{w(e)\geq 0\}_{e\in E}$ (spanning)

aufspannende Kanten: $A\subseteq E$, Graph (V,A) z. hängend

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

können aufsp. Kanten mit min. Gewicht Kreis enthalten?

Beobachtung: kann aufspannende Kanten mit min. Gewicht immer ohne Kreis wählen

Beweis: entferne Kante auf Kreis solange bis kein Kreis

Spannbaum: $T\subseteq E$ aufspannend, kein Kreis enthalten

minimaler Spannbaum (MST): S. baum mit min. Gewicht

weitere Anwendungen

"clustering": zerlege Graph in k "gute" Teile

genauer: Teilgraph mit k ZHKs und min. Gewicht

- berechne MST
- entferne k-1 schwerste Kanten

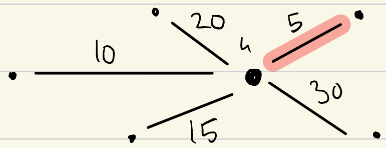
Korrektheit hier nicht besprochen

Annahme: alle Kantengewichte verschieden

(vereinfacht Analyse; Algorithmen auch sonst korrekt)

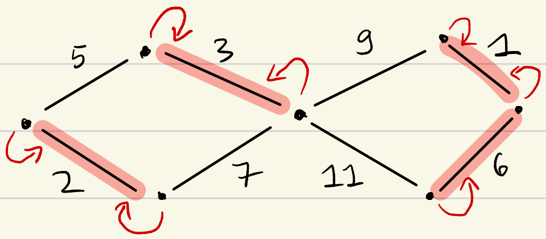
sichere Kante: enthalten in jedem MST

Beispiel: betrachte Knoten u; welche inzidente Kante sicher?



Vermutung: minimale Kante an Knoten sicher

bilden diese Kanten MST?

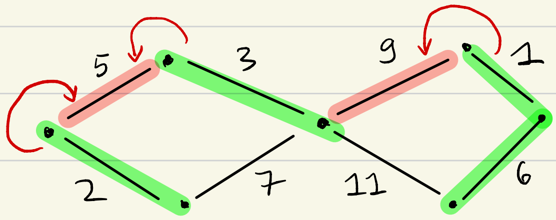


nicht aufspannend

⇝ erhalten Wald F (forest)

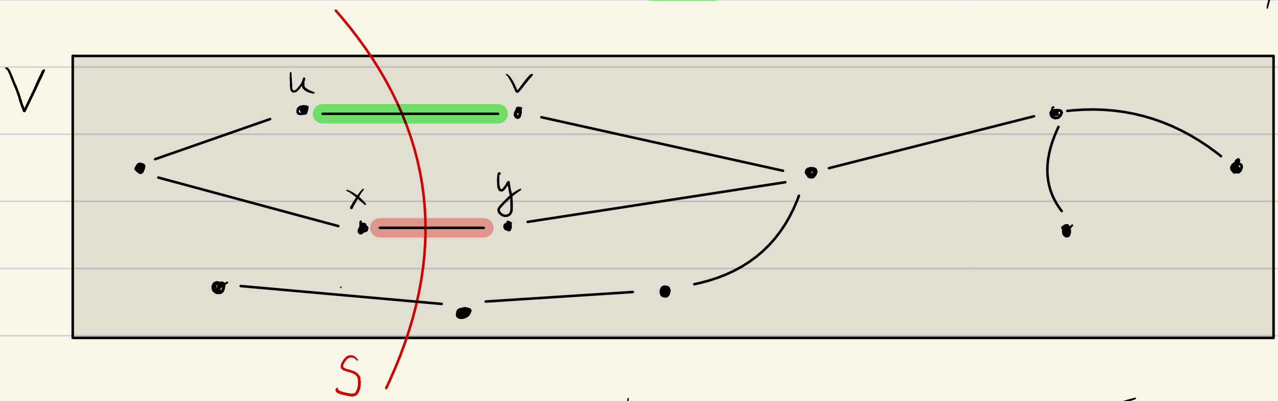
weitere sichere Kanten?

Vermutung: minimale Kante an ZHK von F sicher



Schnittprinzip (bestätigt unsere Vermutungen)

$\forall S \subseteq V.$ minimale Kante uv an S sicher ($u \in S, v \notin S$)



Beweis: (indirekt) betrachte Spannbaum T, $uv \notin T$

zu zeigen: T nicht MST

betrachte $xy \in T$ an S (also $w(xy) > w(uv)$)

ersetze xy durch uv in T

$\rightsquigarrow$ erhalte besseren Spannbaum — warum? aufspannend?

$\rightsquigarrow$ T nicht MST

wichtig (sonst Beweis falsch):

wähle xy auf Kreis in $T \cup \{uv\}$

Borurka (G):

$F \leftarrow \emptyset$ (sichere Kanten)

while F nicht Spannbaum:

$(S_1, \dots, S_k) \leftarrow$ ZHKs von F

$(e_1, \dots, e_k) \leftarrow$ minimale Kanten an $S_1, \dots, S_k$

$F \leftarrow F \cup \{e_1, \dots, e_k\}$

Laufzeit pro Iteration: $O(|V| + |E|)$

- berechne ZHKs mit DFS
- gehe einmal durch alle Kanten um minimale für jede ZHK zu finden

Anzahl Iterationen: $\leq \log |V|$

jede Iteration halbiert Anzahl ZHKs

$|\{e_1, \dots, e_k\}| \geq k/2$
da jede Kante für höchstens 2 ZHKs minimal sein kann

totale Laufzeit: $O((|V| + |E|) \cdot \log |V|)$

Variation 1: auf eine ZHK konzentrieren

Prim (G, s):

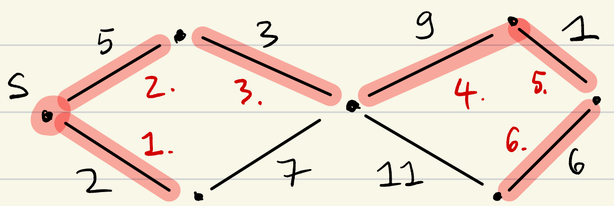
$F \leftarrow \emptyset$

$S \leftarrow \{s\}$ (ZHK von s in F)

while F nicht Spannbaum:

$\quad u^* v^* \leftarrow$ minimale Kante an S $(u^* \in S,\ v^* \notin S)$

$\quad F \leftarrow F \cup \{u^* v^*\}$; $S \leftarrow S \cup \{v^*\}$



Laufzeit:

wie minimale Kante schnell finden?

wie Dijkstra: priority queue (min-heap)

Prim (G, s): (mit min-heap)

$H \leftarrow$ make-heap (V, ∞), $S \leftarrow \emptyset$

$d[s] \leftarrow 0$, $d[v] \leftarrow \infty$ für $v \in V \setminus \{s\}$

decrease-key$(H, s, 0)$

while $H \neq \emptyset$:

$\quad v^* \leftarrow$ extract-min(H)

$\quad S \leftarrow S \cup \{v^*\}$

$\quad$ for $v^*v \in E, v \notin S$:

$\quad\quad d[v] \leftarrow \min\{d[v], \underline{w(v^*v)}\}$

$\quad\quad$ decrease-key$(H, v, d[v])$

Unterschied Dijkstra: $d[v^*] + w(v^*v)$

Laufzeit:

wie Dijkstra (und Boruvka)

$$O\left((|V| + |E|) \cdot \log |V|\right)$$

Variation 2: sichere Kanten sortiert nach Gewicht

Kruskal(G):

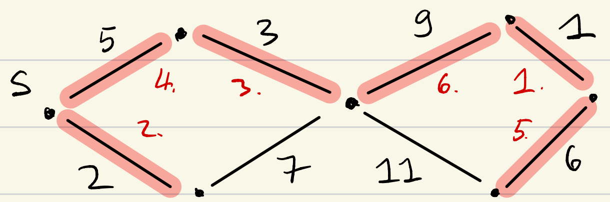
$F \leftarrow \emptyset$ (sichere Kanten)

for $uv \in E$, aufsteigend sortiert

  if u, v in verschiedenen ZHKs von F

    $F \leftarrow F \cup \{uv\}$

Beispiel



Laufzeit: pro Iteration: $O(|V|)$ (ZHKs berechnen)

Anzahl Iterationen: $\leq |E|$

geht es besser?

Idee: Datenstruktur für ZHKs von F

Union-find Datenstruktur $(|V| = n)$

Operationen:

make(V): erstelle Datenstruktur für $F = \emptyset$

same(u,v): teste ob u,v in derselben ZHK von F

union(u,v): vereinige ZHKs von u und v

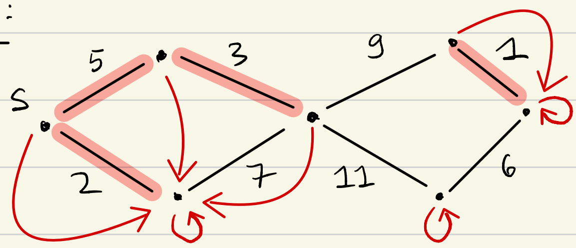
(füge Kante uv zu F hinzu)

Speicher:

rep[v]: eindeutiger Repräsentant der ZHK von V

$(\text{rep}[u] = \text{rep}[v] \iff \text{ZHK}(u) = \text{ZHK}(v))$

Beispiel:



Implementierung:

make(V): $rep[v] \leftarrow v$ für $v \in V$ $O(n)$

same(u,v): teste ob $rep[u] = rep[v]$ $O(1)$

union(u,v): for $x \in V$, $rep[x] = rep[u]$ $O(n)$

$rep[x] \leftarrow rep[v]$

geht es besser als $O(n)$ für union?

Idee: verwalte alle Knoten einer ZHK als Liste

Invariante:

members[rep[u]] ist Liste aller Knoten in ZHK(u)

union(u,v):

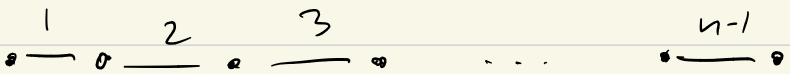
for $x \in$ members[rep[u]]:

$rep[x] \leftarrow rep[v]$

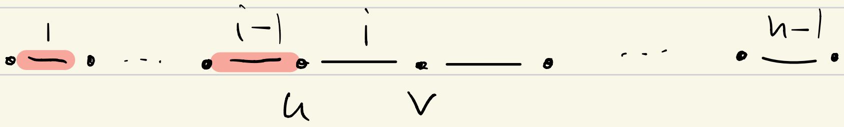
members[rep[v]] $\leftarrow$ members[rep[V]] $\cup \{x\}$

Laufzeit: $O(|ZHK(u)|)$

worst-case



i-ter union Aufruf:



$|ZHK(u)| = i \rightsquigarrow$ Laufzeit: $\Theta(i)$

total: $\Theta(1 + 2 + \dots + (n-1)) = \Theta(n^2)$

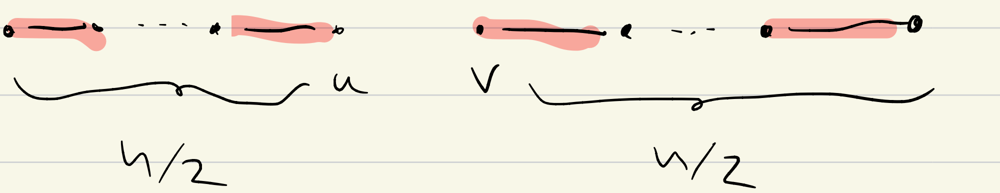
geht es besser?

Idee: durchlaufe nur Knoten der kleineren ZHK

union Laufzeit:

$$\Theta(\min\{|ZHK(u)|, |ZHK(v)|\})$$

worst-case $\Theta(n)$ für einzelne union Aufruf



$$\min\{ZHK(u), ZHK(v)|\} = n/2$$

Amortisierte Analyse

totale Laufzeit aller union Aufrufe $O(n \cdot \log n)$

$\rightsquigarrow$ im Mittel nur $O(\log n)$ pro union Aufruf

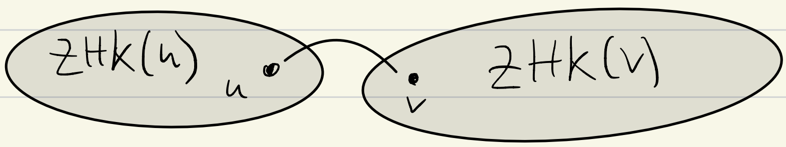
! Mittel nur über Aufrufe; Graph bleibt worst-case

Beweis:

$N_u :=$ Anzahl Änderungen von rep[u]

totale Laufzeit $O\left(\sum_{u \in V} N_u \right)$

Wie gross kann N_u sein?



rep[u] nur verändert wenn $|ZHK(u)| \leq |ZHK(v)|$

$\rightsquigarrow$ Grösse der ZHK von u mindestens verdoppelt

$\rightsquigarrow N_u \leq \log_2 n$

$\rightsquigarrow \sum_{u \in V} N_u \leq n \cdot \log_2 n$

Laufzeit Kruskal: $O\Big(\underbrace{|E| \cdot \log |E|}_{\text{Sortieren}} + \underbrace{|V| \cdot \log |V|}_{\text{union find}}\Big)$