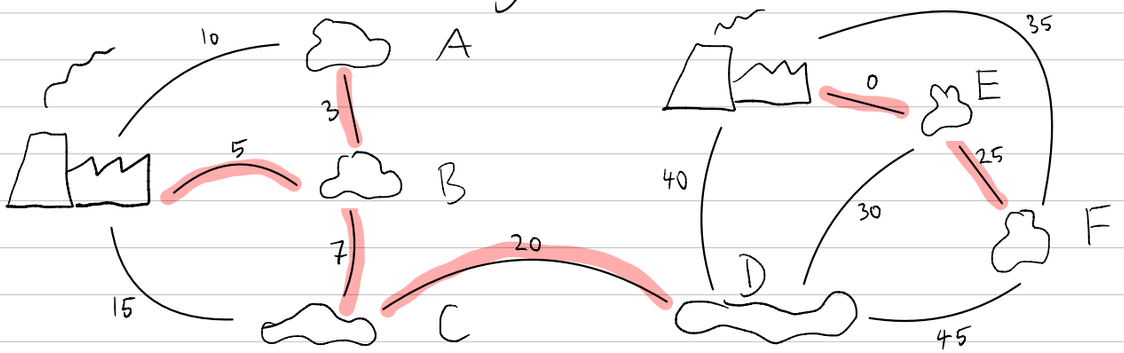
1926: Elektrifizierung Mährens (Region in Tschechien)

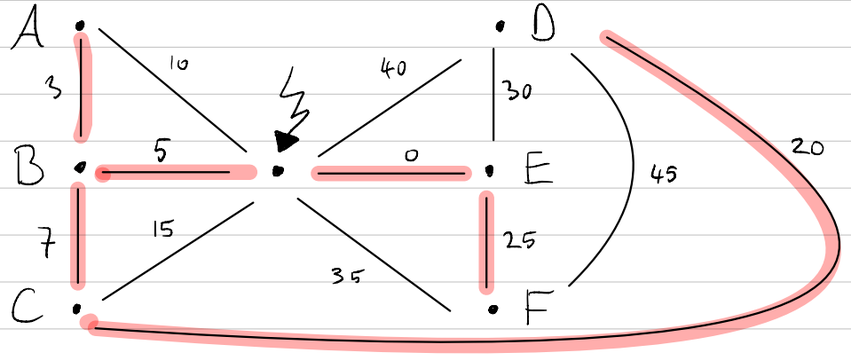


Frage an Otakar Borůvka (Mathematiker)

welche Stromleitungen bauen so dass

- alle Städte mit Strom versorgt
- Gesamtkosten minimal

gewichteter Graph:



Ziel: zusammenhängender Teilgraph mit min. 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

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

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

Beobachtung: aufspannende Kanten mit min. Gewicht können als Spannbaum gewählt werden

Beweis: entferne Kante aus Kreis solange bis S.baum

minimaler Spannbaum (MST) S.baum mit min. Gew.

weitere Anwendungen

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

genauer: Teilgraph mit k ZHK und min. Gew.

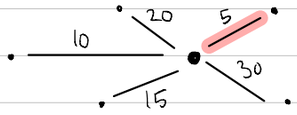
- berechne MST
- entferne k-1 schwerste Kanten

Annahme: alle Kantengew. verschieden

(vereinfacht Analyse; Algorithmen auch sonst korrekt)

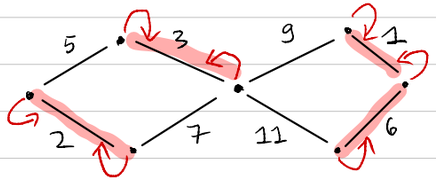
sichere Kante: enthalten in jedem MST

alle Kanten an einem Knoten



Vermutung: minimale Kante an Knoten sicher

bilden diese Kanten MST?

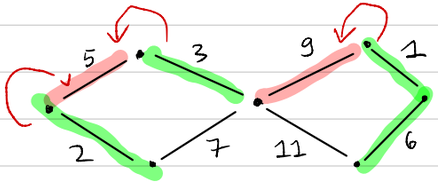


nicht aufspannend

$\rightsquigarrow$ erhalten Wald F (forest)

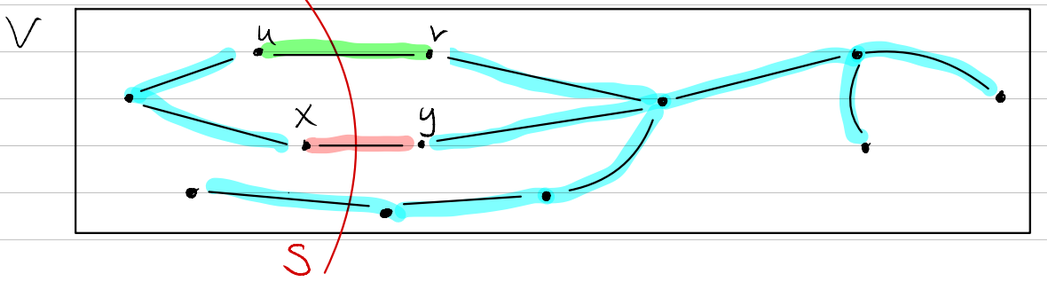
weitere sichere Kanten

Vermutung: minimale Kanten an ZHK von F sicher



Schnittprinzip:

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



Beweis: (indirekt) betrachte S.baum 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

$\rightsquigarrow$ erhalten besseren Spannbaum

$\rightsquigarrow$ T nicht MST

warum?

wichtig (sonst Beweis falsch):

wähle xy auf Kreis in $T \cup \{uv\}$

Boruvka (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: $O(|V|+|E|)$

Anzahl Iterationen: $\leq \log_2 |V|$

jede Iteration halbiert Anzahl ZHKs

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

Variation 1: auf eine ZHK konzentrieren

Prim (G, s):

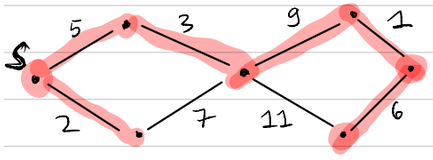
$F \leftarrow \emptyset$

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

while F nicht S-baum:

$u^* v^* \leftarrow$ min. Kante an S ($u^* \in S, v^* \notin S$)

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

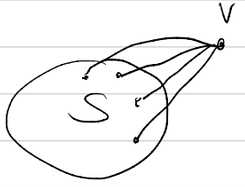


Laufzeit:

wie min. Kante an S 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 $H \neq \emptyset$:

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

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

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

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

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

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

Laufzeit:

wie Dijkstra (und Borůvka)

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

Variation 2: sichere Kanten sortiert nach Gew. (aufst.)

Kruskal(G):

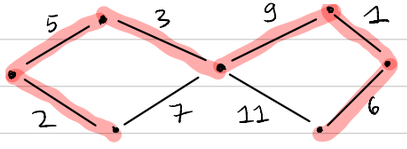
$F \leftarrow \emptyset$

for $uv \in E$, aufsteigend sortiert

  if u, v in verschiedenen ZHKs von F:

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

Beispiel:



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

Anzahl Iterationen: $\leq |E|$

total: $O(|E| \cdot |V| + |E| \cdot \log |E|)$ (sortieren)

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 ZHK von u und v
(füge Kante uv zu F hinzu)

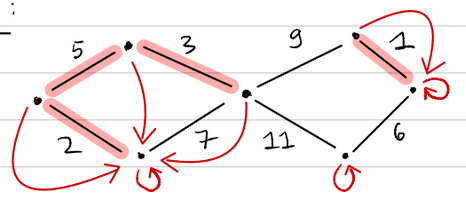
Speicher

rep[v]: eindeutiger Repräsentant der ZHK von v
($rep[u]=rep[v] \Leftrightarrow ZHK(u)=ZHK(v)$)

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

Beispiel:



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)

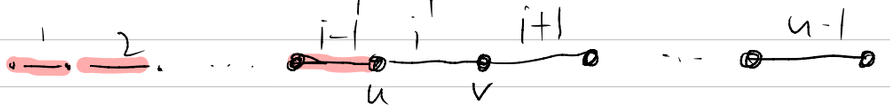
union(u,v):

```
for x ∈ members[rep[u]]:
    rep[x] ← rep[v]
    members[rep[v]] ← members[rep[v]] ∪ {x}
```

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

worst-case

i-ter union Aufruf:



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

total: $\Theta(1+2+3+\dots+(n-1)) = \Theta(n^2)$

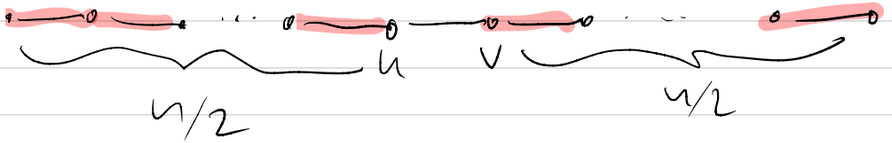
geht es besser?

Idee: durchlaufe nur Knoten der kleineren ZHK

union Laufzeit:

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

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



Amortisierte Analyse

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

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

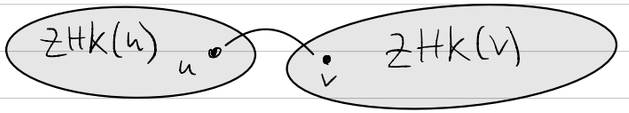
⚠ Mittel nur über Aufruf; Graph bleibt worst-case

Beweis:

$N_u :=$ Anzahl Änderungen von $rep[u]$

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

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: $O(\underbrace{|E| \log |E|}_{\text{sortieren}} + \underbrace{|V| \cdot \log |V|}_{\text{union find}})$