Dijkstra (s):

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

$S \leftarrow \emptyset$ [$H \leftarrow$ make-heap(V), decrease-key$(H, s, 0)$]

WHILE $S \neq V$:

wähle $v^* \in V \setminus S$ mit $d[v^*]$ minimal

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

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

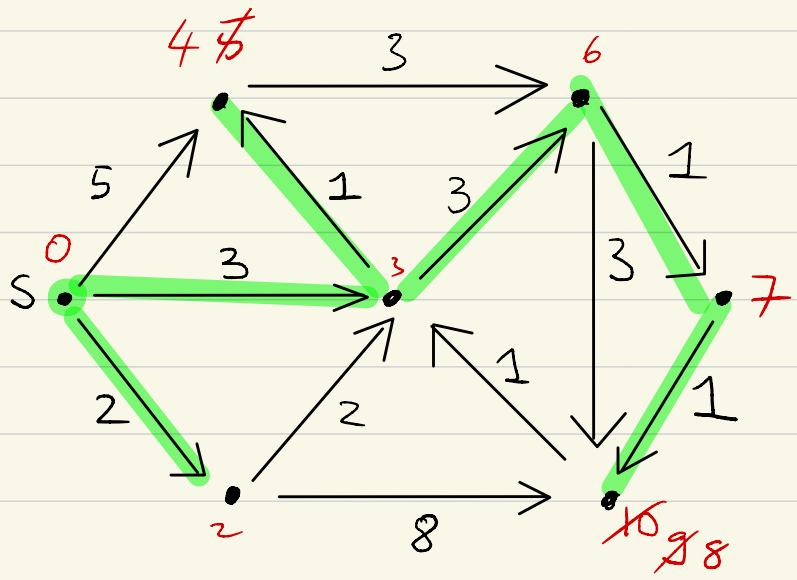
FOR $(v^*, v) \in E$, $v \notin S$:

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

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

Beispiel:

shortest-path tree



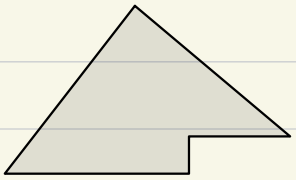
wie v^* schnell finden? alle Knoten in $V \setminus S$ aufzählen?

Gleiche Kantenkosten:

≤ 2 (endliche) Schrankenwerte: $d[v^*]$, $d[v^*]+1$

$\rightsquigarrow$ finde v^* in $O(1)$ mittels FIFO Queue (BFS)

Welche Datenstruktur für allgemeine nicht-neg. Kantenkosten?



Priority Queue z.B. Min Heap (vgl. HeapSort)

make-heap(V): setze alle Schlüssel auf ∞, $O(n)$

extract-min(H): entferne Minimum, $O(\log n)$ versickern um Heap Eigenschaft wieder herzustellen

decrease-key(H, v, c): verkleinere Schlüssel von v auf c, $O(\log n)$ Element aufsteigen lassen um Heap Eigenschaft wieder herzustellen

Laufzeit Dijkstra

extract-min $\leq n$ # decrease-key $\leq m$

total: $O(n + (\#\text{extract-min} + \#\text{decrease-key}) \cdot \log n)$

$= O((n + m) \cdot \log(n))$

Allgemeine Kantenkosten (negativ erlaubt)

Idee: sortiere nach Anzahl Kanten im günstigsten Weg

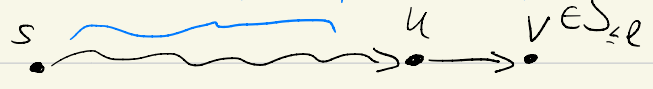
$S_{\leq \ell} := \{ v \in V \mid \exists \text{ günstigster Weg mit } \leq \ell \text{ Kanten} \}$

$S_{\leq 0} = \{s\}$, $S_{\leq n-1} = V$ Annahme: $\exists$ neg. Zyklus, alle Knoten von s erreichbar

Rekurrenz: $\forall v \in S_{\leq \ell} \setminus \{s\}$.

$$d(s,v) = \min \{ d(s,u) + c(u,v) \mid u \to v,\ u \in S_{\leq \ell-1} \} \quad (1)$$

Beweis: günstigster Weg nach v



wie $S_{\leq 1}$ berechnen? Unklar!

Aber: können Schranken berechnen, die genau sind für $S_{\leq 1}$! $\quad d[v] = \begin{cases} 0 & \text{falls } v = s \\ c(s,v) & \text{falls } s \to v \\ \infty & \text{sonst} \end{cases}$ $\quad \forall v \in S_{\leq 1}.\ d(s,v) = c(s,v)$

ℓ-gute Schranken: $d[v] \begin{cases} = d(s,v) & \text{falls } v \in S_{\leq \ell} \\ \geq d(s,v) & \text{sonst} \end{cases}$

Schranken verbessern: $(\ell-1)$-gute S. $\to$ ℓ-gute S.

For $v \in V$:

$$d[v] \leftarrow \min \{ d[v], \min_{u \to v} \underbrace{d[u]}_{= d(s,u) \text{ für } u \in S_{\leq \ell-1}} + c(u,v) \}$$

$= d(s,v)$ für $v \in S_{\leq \ell}$ (1)

Bellman - Ford

initialisiere $d[]$ wie Dijkstra (0-gute Schranken)

iteriere "Schranken verbessern" $(n-1)$-mal

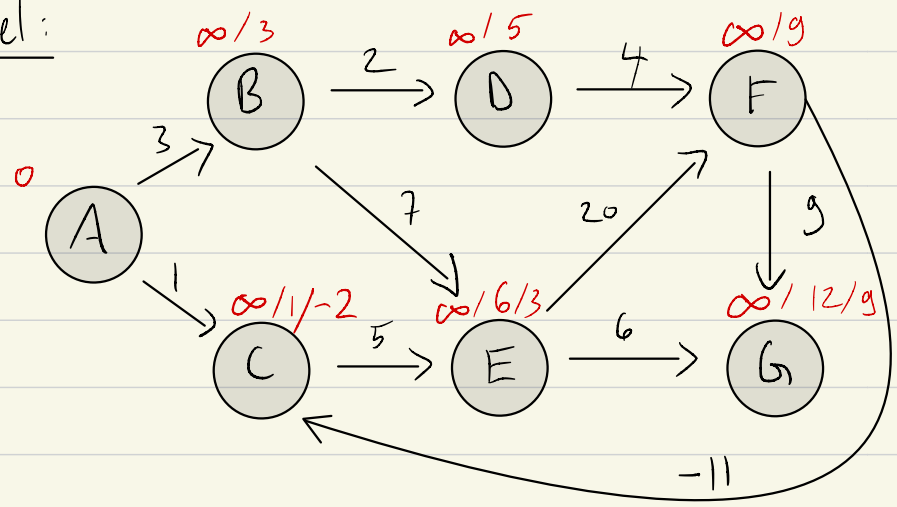
Korrektheit: Alg. berechnet $(n-1)$-gute Schranken $d[]$

$\rightsquigarrow d[v] = d(s,v) \quad \forall v \in S_{n-1} = V$

Laufzeit: pro "Schranken verbessern": $O(m)$

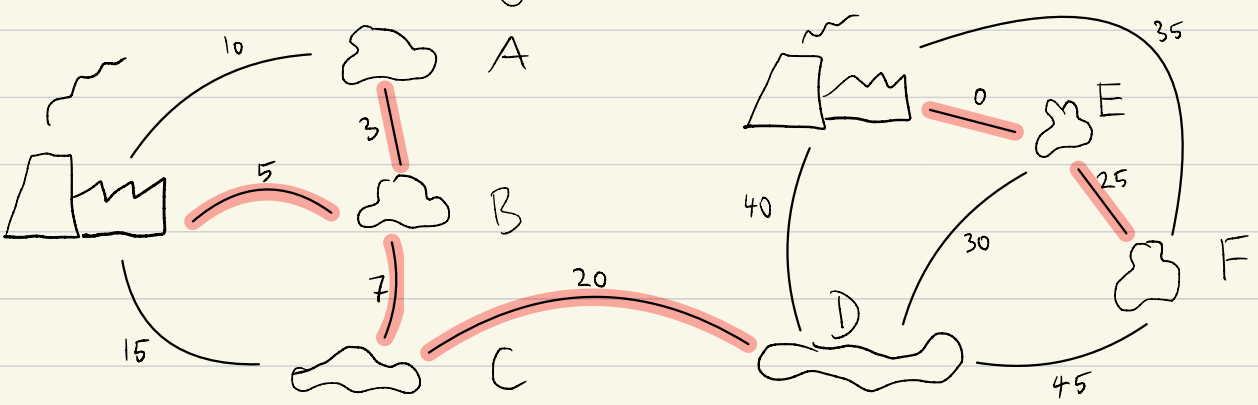
gesamt $O(n \cdot m)$

Beispiel:



d	A	B	C	D	E	F	G
	0	∞	∞	∞	∞	∞	∞
	"	3	1	5	6	9	12
	"	"	-2	"	3	"	9

① 1926: Elektrifizierung Mährens (Region in Tschechien)



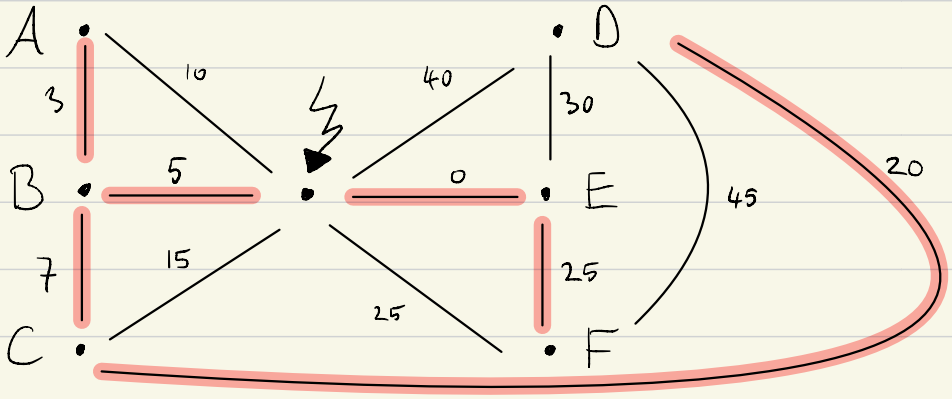
Frage an Otakar Boruvka (Mathematiker)

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

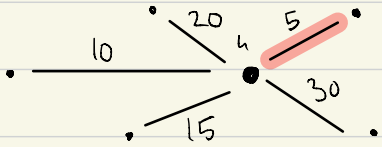
Korrektheit hier nicht besprochen

Annahme: alle Kantengewichte verschieden

(vereinfacht Analyse; Algorithmen auch sonst korrekt)

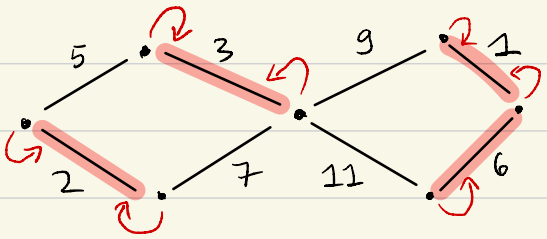
sichere Kante: enthalten in jedem MST

Beispiel: betrachte Knoten u; welche inzidente Kante sicher?



Vermutung: minimale Kante an Knoten sicher

bilden diese Kanten MST?

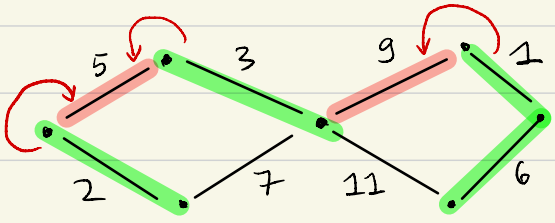


nicht aufspannend

⇝ erhalten Wald F (forest)

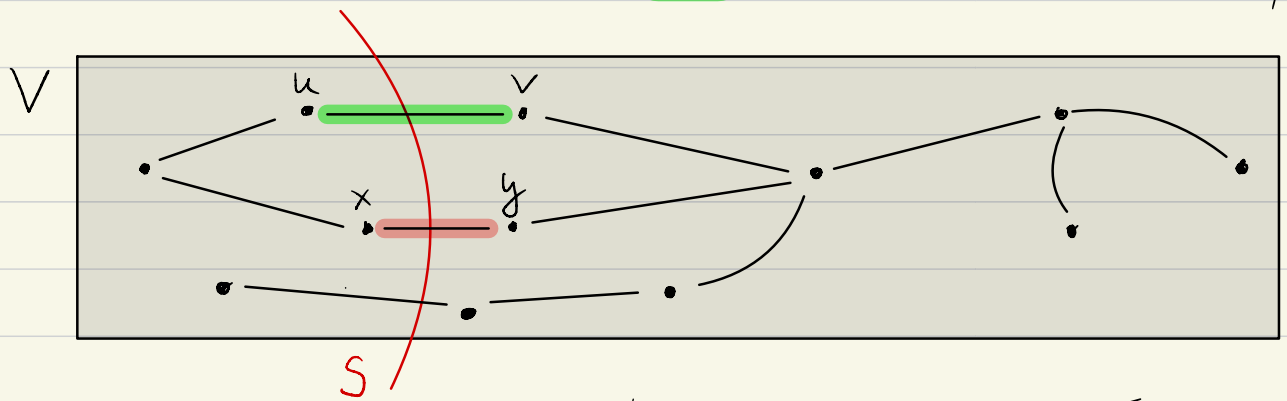
weitere sichere Kanten?

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 $T \cup \{uv\}$

Variation 1: auf eine ZHK konzentrieren

Prim (G, s):

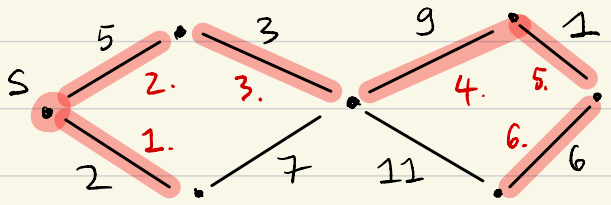
$F \leftarrow \emptyset$

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

while F nicht Spannbaum:

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

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



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 $H \neq \emptyset$:

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

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

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

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

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

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

Laufzeit:

wie Dijkstra (und Boruvka)

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