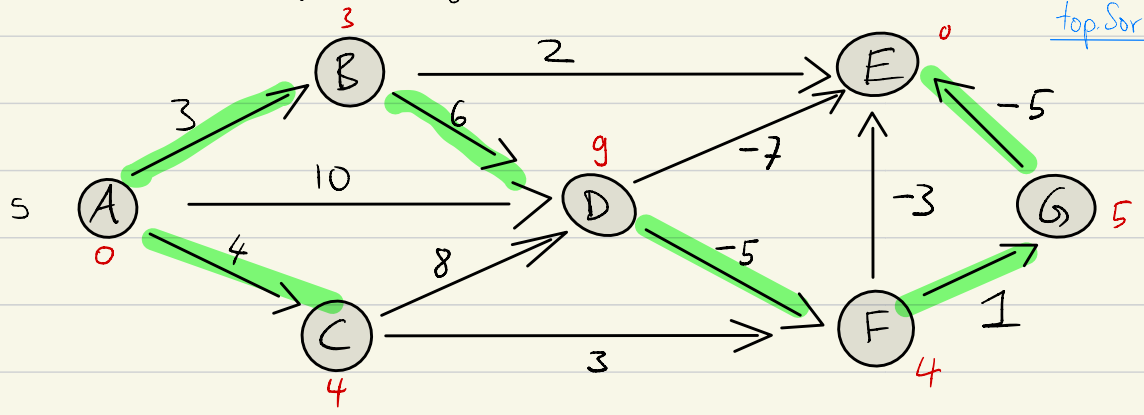
günstigste Wege in gewichteten Graphen

top. Sort.: A B C D F G E



gerichteter Graph $G=(V,E)$, Startknoten $s \in V$, $n=|V|$, $m=|E|$

Kantenkosten: $c(e) \in \mathbb{R}$, $e \in E$

negative Kosten erlaubt (algorithmisch aber schwerer)

Beispiel: E-Autos gewinnen Energie zurück wenn Strecke bergab geht

gesucht: günstigste Wege von s (shortest path)

Wegkosten: Summe der Kantenkosten

Weg $W = (v_0, v_1, \dots, v_\ell)$

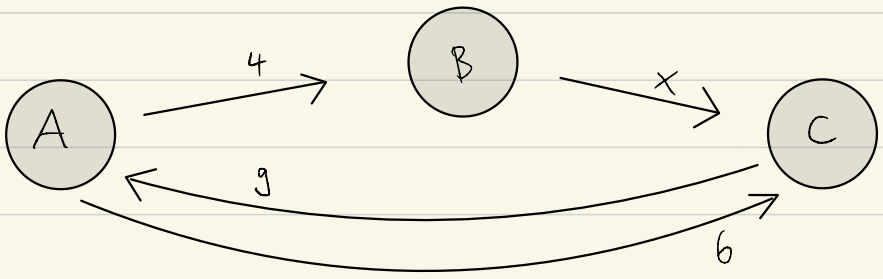
$$c(W) := c(v_0,v_1) + \dots + c(v_{\ell-1}, v_\ell)$$

Notationen: $u \xrightarrow{e} v$ wenn $e=(u,v) \in E$

$u \overset{W}{\rightsquigarrow} v$ wenn W Weg von u nach v

Distanz: $d(u,v) := \min \{ c(W) \mid u \overset{W}{\rightsquigarrow} v \}$

Beispiel:



$c(A \to C) = 6$

$c(A \to B \to C) = 4 + x$

$c(A \to B \to C \to A \to B \to C) = 4 + x + (13 + x)$

$$\text{dist}(A, C) = \begin{cases} 6 & \text{falls } x \geq 2 \\ 4 + x & \text{falls } -13 \leq x < 2 \\ -\infty & \text{falls } x < -13 \end{cases}$$

Zyklus $A \to B \to C \to A$ negativ

$\rightsquigarrow$ wiederhole Zyklus beliebig oft

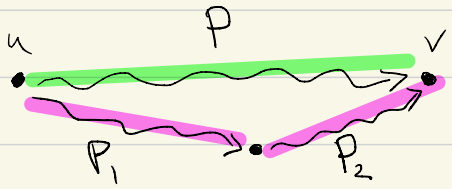
$\rightsquigarrow$ beliebig kleine negative Kosten

Annahme: $\nexists$ negativer Zyklus

$\rightsquigarrow$ kann jeden Weg zu Pfad abkürzen

$\rightsquigarrow$ $d(s, s) = 0$

Dreiecksungleichung: $d(u,v) \leq d(u,w) + d(w,v)$

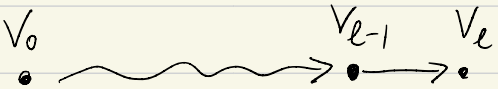


$\rightsquigarrow (P_1, P_2)$ ist Weg von u nach v

$\rightsquigarrow c(P_1, P_2) = d(u,w) + d(w,v)$

wähle günstigste Pfade P, P_1, P_2

Struktur günstigster Wege:



$(\ell \geq 1)$

Falls $v_0, \ldots, v_{\ell-1}, v_\ell$ günstigst, dann auch $v_0, \ldots, v_{\ell-1}$

Rekurrenz:

$$\forall v \neq s. \quad d(s,v) = \min_{u \to v} d(s,u) + c(u,v)$$

Dynamische Programmierung ? Berechnungsreihenfolge ?

Azyklische Graphen

Idee: topologische Sortierung

FOR $v \in V$ in topologischer Reihenfolge:

$$d[v] \leftarrow \begin{cases} 0 & \text{falls } v = s \\ \infty & \text{falls } v \neq s,\ \deg_{in}(v) = 0 \\ \min\limits_{u \to v} d[u] + c(u,v) & \text{sonst} \end{cases}$$

$d[u]$: u kommt vor v in top Sort. $\rightsquigarrow$ $d[u]$ schon berechnet

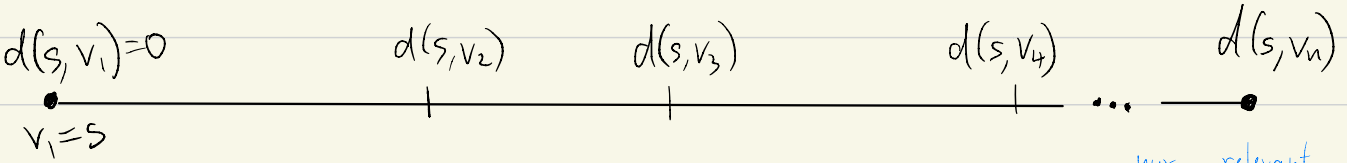
Beispiel: siehe S. 1

Laufzeit: $O(|V| + |E|)$ falls Adj.-liste gegeben

(Analyse wie DFS, BFS)

Nicht-negative Kantenkosten

Idee: sortiere Knoten nach Distanz von s



Annahme: alle Distanzen von s verschieden

nur relevant für Diskussion Algorithmus/Analyse funktioniert auch ohne Annahme

Rekurrenz: $d(s,v_k) = \min_{\substack{v_i \to v_k \\ i<k}} d(s,v_i) + c(v_i,v_k) \quad (k \geq 2)$

Beweis: $\forall i > k.\ \underbrace{d(s,v_i)}_{> d(s,v_k)} + \underbrace{c(v_i,v_k)}_{\geq 0} > d(s,v_k)$

$\rightsquigarrow$ Rekurrenz auf S.3: Vorgänger v_i mit $i > k$ irrelevant

wie Reihenfolge $v_1, \dots, v_n$ berechnen?

angenommen: $S = \{v_1, \dots, v_{k-1}\}$ bekannt

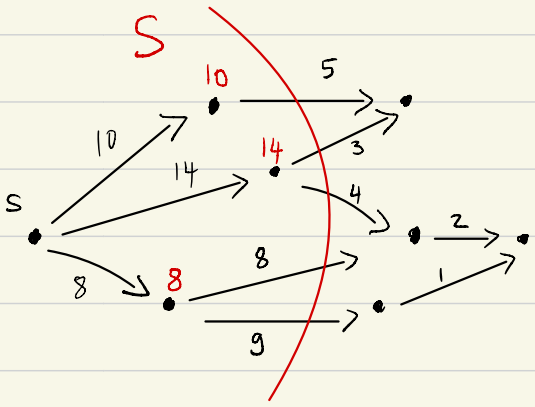
berechne v_k:

wähle Kante $u^* \to v^*$,

$u^* \in S$, $v^* \notin S$ mit

$d(s,u^*) + c(u^*,v^*)$ minimal

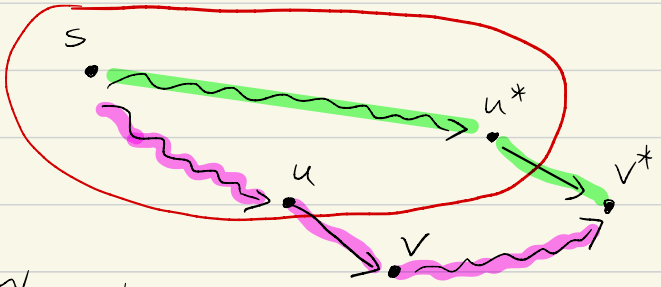
$\rightsquigarrow$ $v_k = v^*$



neu berechnet zuvor bekannt

Behauptung: $d(s,v^*) = d(s,u^*) + c(u^*,v^*)$

Beweis



W muss S an einer Stelle verlassen da $s \in S$ und $v^* \notin S$

$\forall W, s \overset{W}{\rightsquigarrow} v^*$.

$$c(W) \geq d(s,u) + c(u,v) + \underbrace{d(v,v^*)}_{\geq 0} \quad (u \in S, v \notin S)$$

$$\geq d(s,u) + c(u,v)$$

$$\geq d(s,u^*) + c(u^*,v^*) \quad \text{(Wahl von } u^* \to v^*\text{)}$$

$\rightsquigarrow$ kein Weg günstiger ▨

Schnelle Laufzeit: wie v^* schnell berechnen?

alle Kanten $u \to v$ mit $u \in S$, $v \in V \setminus S$ aufzählen?

verwalte Array $d[\,]$ von (oberen) Schranken

$$d[v] = \min_{\substack{u \to v \\ u \in S}} d(s,u) + c(u,v) \qquad (v \neq s)$$

- Wahl von v^*: Knoten in $V \setminus S$ mit minimaler Schranke
- v^* zu S hinzugefügt: alle Schranken bleiben gleich bis auf Nachfolger von v^*

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)]

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

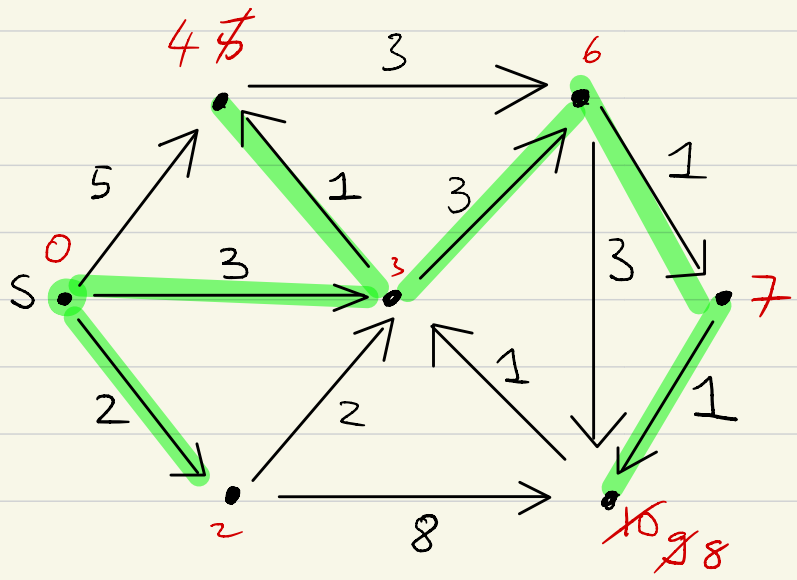
FOR $(v^*, v) \in E$, $v \notin S$:

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

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

Beispiel:

shortest-path tree



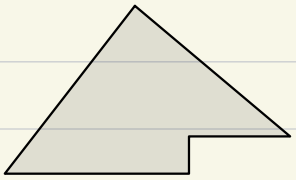
wie v^* schnell finden? alle Knoten in $V \setminus S$ aufzählen?

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

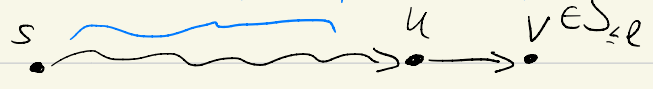
$S_{\leq \ell} := \{ v \in V \mid \exists \text{ günstigster Weg mit } \leq \ell \text{ Kanten} \}$

$S_{\leq 0} = \{s\}$, $S_{\leq n-1} = V$ 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

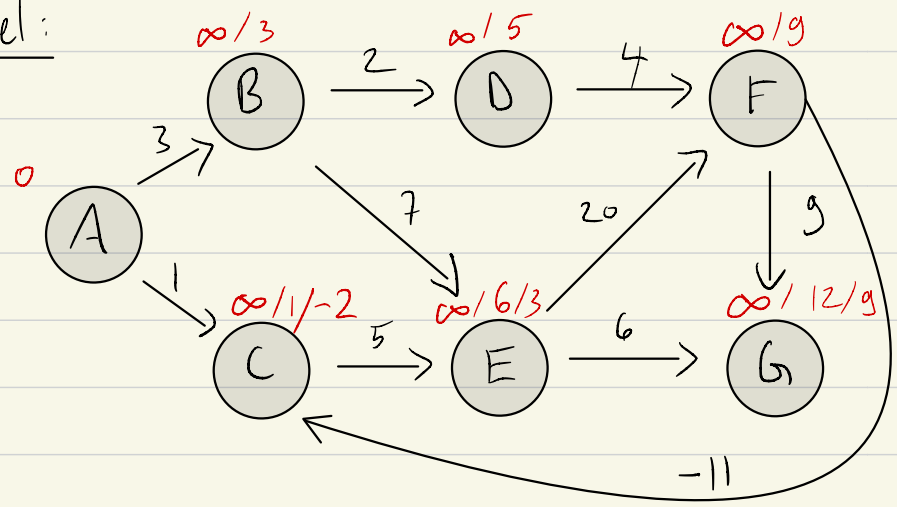
Korrektheit: Alg. berechnet $(n-1)$-gute Schranken $d[]$

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

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

negative Zyklen — wie testen ob existent?

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

Korrektheit:

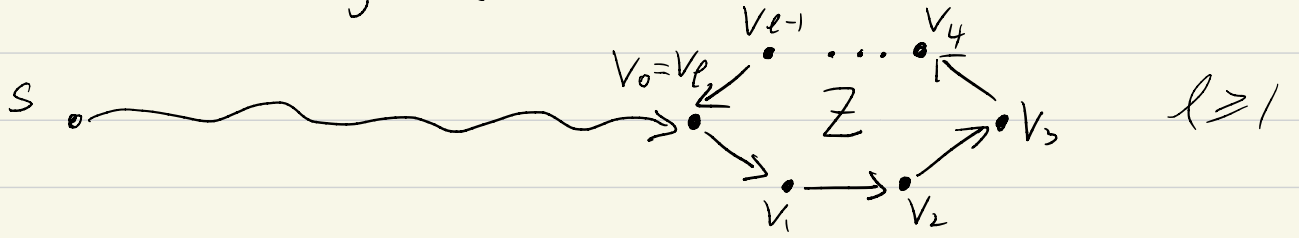
$d[\,]$: Schranken nach $n-1$ Iterationen

$d'[\,]$: 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



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

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

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

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

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