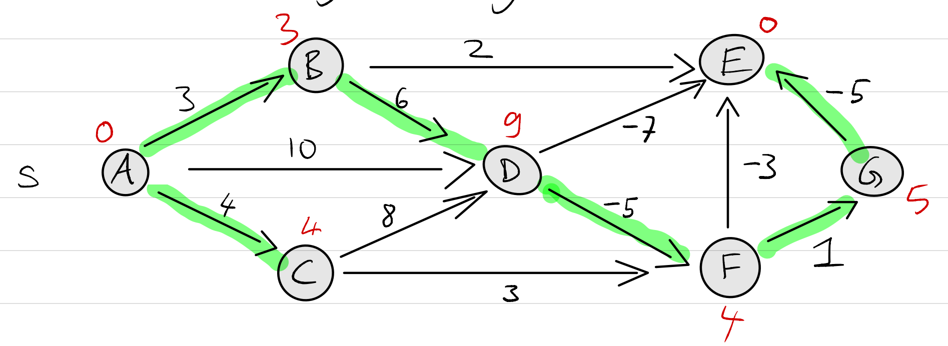
1

Kürzeste Wege in gewichteten Graphen



gerichteten Graph $G=(V,E)$, Startknoten s, $n=|V|$, $m=|E|$

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

gesucht: günstigste Wege von s (shortest paths)

Weg kosten: Summe der Kantenkosten

Weg $W=(v_0, v_1, \dots, v_e)$ **kurz:** $v_0 \xrightarrow{W} v_e$

$$c(W) = c(v_0,v_1) + c(v_1,v_2) + \dots + c(v_{e-1},v_e)$$

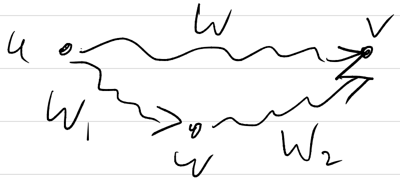
neg. Kantenkosten: erlaubt, aber bringen Komplikation (zu besprechen)

2

Definition: Distanz

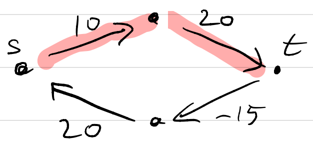
$$d(u,v) = \min \{ c(W) \mid u \xrightarrow{W} v \}$$

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

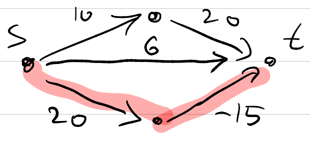


(W_1, W_2) ist möglicher Weg

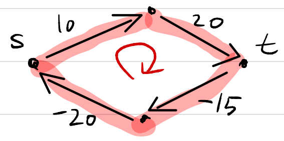
Beispiele:



$d(s,t)=30$



$d(s,t)=5$

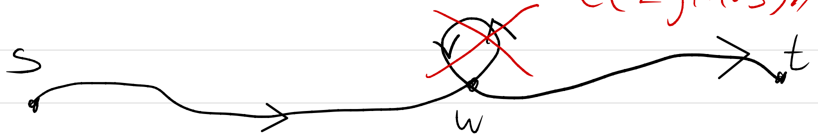


$d(s,t) = -\infty$!

$\rightsquigarrow \nexists$ günstigster Weg

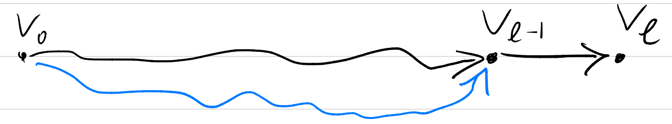
Annahme: $\nexists$ neg. Zyklus

Lemma: kann Weg zu Pfad abkürzen



Also: $d(s,s)=0$

Beobachtung: Falls $\overbrace{v_0, \dots, v_{\ell-1}, v_\ell}^{W}$ günstigst, dann auch $\overbrace{v_0, \dots, v_{\ell-1}}^{W'}$ günstigst



$c(W) = c(W') + c(v_{\ell-1}, v_\ell)$

Rekurrenz: $d(s,v) = \min_{(u,v)\in E} d(s,u) + c(u,v) \quad (v \neq s)$

Dynamische Programmierung? Berechnungsreihenfolge?

Azyklische Graphen

berechne Rekurrenz entlang topologischen Sortierung

$d[s] \leftarrow 0$, $d[v] \leftarrow \infty$ für $v \in V$, nicht erreichbar von s

For $v \in V \setminus \{s\}$, erreichbar von s, topologische Reihenfolge

$$d[v] \leftarrow \min_{(u,v)\in E} \underbrace{d[u]}_{\text{schon berechnet}} + c(u,v)$$

u kommt vor v in top. Sort

Laufzeit: $O(n+m)$ falls Adjazenzliste geg.

Nicht-negative Kantenkosten

Idee: sortiere nach Distanz von s

Annahme: alle Distanzen von s verschieden

Rekurrenz: $d(s,v_k) = \min_{\substack{(v_i,v_k)\in E \\ i<k}} d(s,v_i) + c(v_i,v_k)$

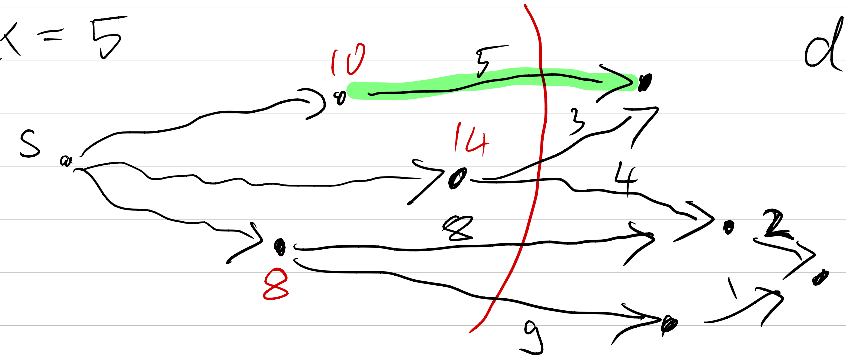
Beweis: $\forall i>k$: $\underbrace{d(s,v_i)}_{>d(s,v_k)} + \underbrace{c(v_i,v_k)}_{\geq 0} > d(s,v_k)$

⇝ vorherige Rekurrenz: nur v_i mit $i<k$ relevant

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

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

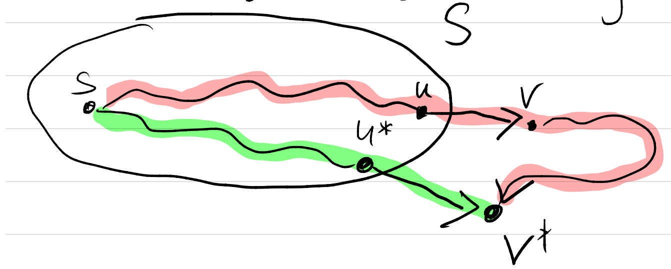
$k = 5$ $\qquad d(s,v_5) = 15$



Lemma: Für $S \subseteq V$ mit $s \in S$, sei $(u^*,v^*) \in E$, $u^* \in S$, $v^* \notin S$ mit $d(s,u^*) + c(u^*,v^*)$ minimal. Dann,

$$\underbrace{d(s,v^*)}_{\text{neu bekannt}} = \underbrace{d(s,u^*)}_{\text{zuvor bekannt}} + c(u^*,v^*)$$

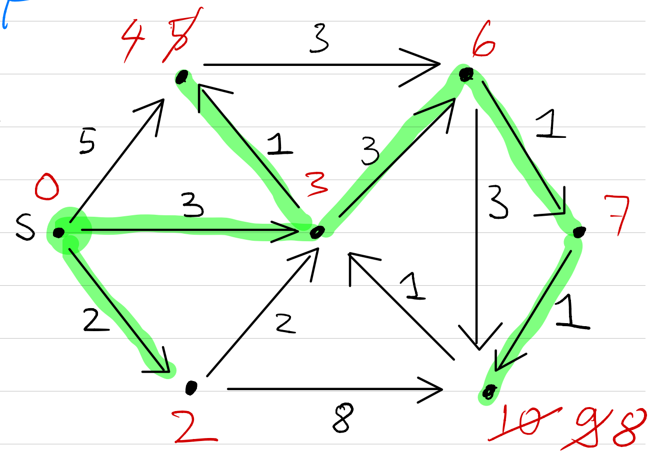
Beweis: günstigster Weg W nach v^* muss S verlassen



$$d(s,v^*) = c(W) \geq d(s,u) + c(u,v) \geq d(s,u^*) + c(u^*,v^*) \geq d(s,v^*)$$

alles gleich!

shortest path tree



Algorithmus (Dijkstra):

$S \leftarrow \{s\}$

while $S \neq V$:

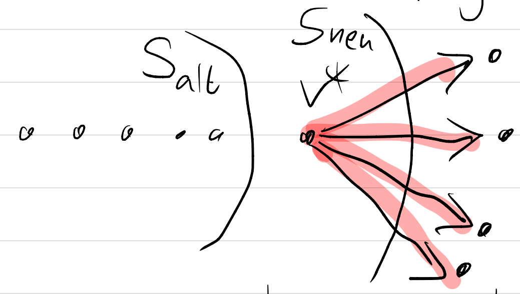
  wähle $v^* \notin S$ mit kleinstem $\min_{\substack{(u,v^*)\in E \\ u\in S}} d(s,u) + c(u,v^*)$

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

Schnelle Laufzeit

verwalte Array d[] mit $d[v] = \min_{(u,v)\in E,\, u\in S} d(s,u) + c(u,v)$

- v^* gewählt $\overset{\text{Lemma}}{\Longrightarrow}$ $d[v^*] = d(s,v^*)$
- v^* zu S hinzufügen $\Rightarrow$ d[v] bleibt gleich bis auf Nachfolger von v^*

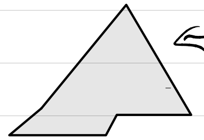


wie v^* schnell finden?

verwende Datenstruktur für $V \setminus S$:

- d[v] Werte als Schlüssel
- schnelles Entfernen des Minimums

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



heap Eigenschaft

Laufzeit Dijkstra mit Heap:

$$O(n + (\underbrace{\#\text{extract-min}}_{\le n} + \underbrace{\#\text{decrease-key}}_{\le 1+m}) \cdot \log(n)) = O(n+m) \cdot \log(n)$$

Operationen

make-heap(V): Setze alle Schlüssel auf ∞; $O(n)$

extract-min(H): entferne Min. $O(\log n)$

decrease-key(H, v, c): verkleinere key von v auf c; $O(\log n)$

Dijkstra(G, s):

```
H ← make-heap(V), S ← ∅
d[s] ← 0; d[v] ← ∞ für v ∈ V \ {s}
decrease-key(H, s, 0)
while S ≠ V:
    v* ← extract-min(H)
    S ← S ∪ {v*}
    for (v*, v) ∈ E, v ∉ S:
        d[v] ← min{ d[v], d[v*] + c(v*, v) }
        decrease-key(H, v, d[v])
```

Allgemeine Kantengewichte (neg. erlaubt)

Idee: sortiere Knoten nach Anzahl Kanten im günstigsten W.

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

$S_0 = \{s\} \qquad S_{n-1} = V$ (Annahme: $\nexists$ neg. Zyklus)

Rekurrenz: $\forall v \in S_\ell \setminus \{s\}.\ d(s,v) = \min\limits_{(u,v) \in E,\, u \in S_{\ell-1}} d(s,u) + c(u,v)$

Wie S_ℓ berechnen (selbst wenn $S_{\ell-1}$ bekannt)?

Rekurrenz zeigt wie Distanz berechnen! Reicht!

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

Schranken verbessern: $(\ell-1)$-gut $\rightarrow$ ℓ-gut

For $v \in V$:

$$d[v] \leftarrow \underbrace{\min \Big\{ d[v],\ \min_{(u,v) \in E} \underbrace{d[u] + c(u,v)}_{\geq d(s,v)} \Big\}}_{= d(s,v) \text{ falls } v \in S_\ell} \quad \Bigg\} \ O(m+n)$$

Bellmann-Ford:

totale Laufzeit: $O(n \cdot m)$