Algorithmen

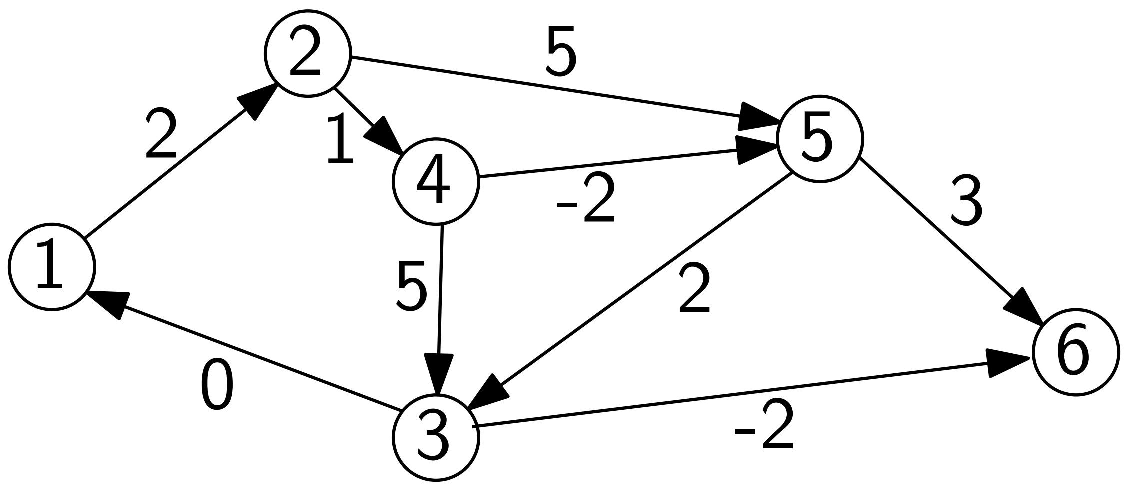
Michael Kaufmann

16/11/2021

III.b Kürzeste (billigste) Wege

Definitionen

Gegeben sei Graph $G = (V, E)$ sowie Kantenkosten $c\colon E \to \mathbb{R}$.
Name: Netzwerk (V, E, c).



Kosten des Pfades $c(v_0, v_1, \ldots, v_k) := \sum_{i=1}^{k} c(v_{i-1}, v_i)$

Betrachten Varianten 'single source shortest paths'
sowie 'all pairs shortest paths', zuerst SSSP.

Single Source Shortest Paths

Sei $s \in V$ der Startpunkt der Pfade (Quelle/Source).
Für $u \in V$ sei $P(s,u)$ die Menge aller Pfade von s nach u.

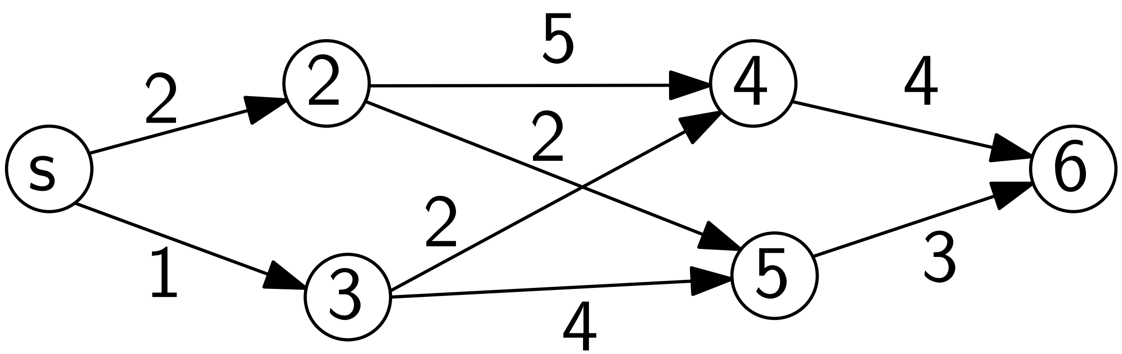
$$\text{Sei } \delta(u) := \begin{cases} \infty & \text{falls } P(s,u) = \emptyset \\ inf\{c(p) \mid p \in P(s,u)\} & \text{sonst} \end{cases}$$

Ein zyklischer Pfad p mit $c(p) < 0$ heißt 'negativer Zykel'.

Lemma:
Sei $u \in V$.
a) $\delta(u) = -\infty \Leftrightarrow u$ erreichbar von negativem Zykel, der von s aus erreichbar ist.
b) $\delta(u) \in \mathbb{R} \Rightarrow \exists$ billigsten Weg von s nach u mit Kosten $\delta(u)$.

1. Graph ist DAG.

DAG $\rightarrow$ topologische Sortierung (wird angenommen)



Algorithmus:

Distanzen $d(s) \leftarrow 0$; $\texttt{Pfad}(s) \leftarrow (s)$;

for all $(v \in V \backslash \{s\})$ $\{d(v) \leftarrow \infty; \}$

for $(v \leftarrow s + 1$ to $n)$ {

$$d(v) \leftarrow \min_{u \geq s}\{d(u) + c(u, v) \mid (u, v) \in E\};$$

$\texttt{Pfad}(v) \leftarrow \texttt{concat}(\texttt{Pfad}(u), v))$;

}

DAG (Korrektheit, Laufzeit)

Lemma 1: Nach Ausführung des Algorithmus gilt
$d(v) = \delta(v)$ für alle $v \in V$

Beweis mit Fallunterscheidung:

$v < s$**:** $\delta(v) = \infty = d(v)$

$v \geq s$**:** Induktion über i:

IA: $i = 0$: $\delta(s + i) = \delta(s) = 0 = d(s) = d(s + i)$

IV: $\delta(s + j) = d(s + j)$ für alle $j < i$

IS:
$$\begin{aligned}
\delta(s + i) &= \min_{v \in V \setminus \{s+i\}} \{\delta(v) + c(v, s + i)\} \\
&= \min_{v < s+i} \{\delta(v) + c(v, s + i)\} \\
&\overset{\text{IV}}{=} \min_{v < s+i} \{d(v) + c(v, s + i)\} \\
&= d(s + i)
\end{aligned}$$

DAG (Korrektheit, Laufzeit)

Lemma 2: Nach Ausführung des Algorithmus gilt:
$d(v) < \infty \Rightarrow \texttt{Pfad}(v)$ ist billigster Weg von s nach v

Beweis:
Aus Lemma 1 folgt, dass $d(v) = \delta(v)\ \forall v \in V$.
Außerdem $c(\texttt{Pfad}(v)) = d(v)$. Also $c(\texttt{Pfad}(v)) = \delta(v)$

Laufzeit: $\sum_{v \in V} |\texttt{InAdj}(v)| = \sum_{v \in V} \texttt{indeg}(v) = O(m)$
Konkatenation jeweils in $O(1)$.
insgesamt: d-Werte in $O(n+m)$ und Pfade in $O(n)$.

Ab sofort berechnen wir nur d-Werte. Pfade analog.

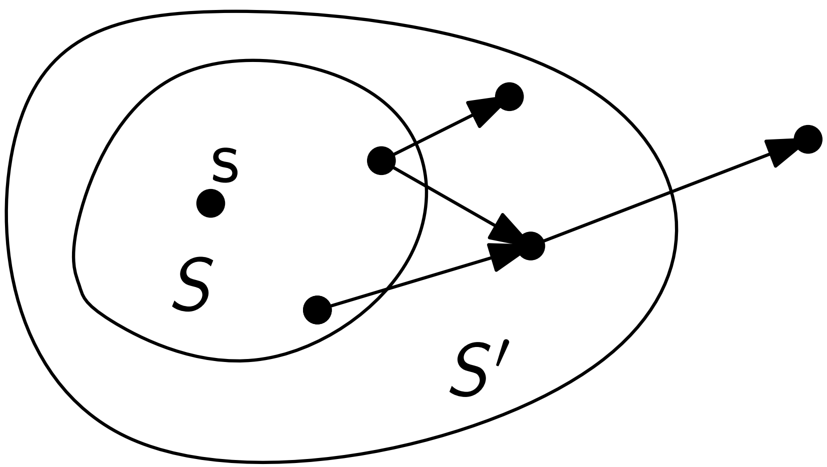
SSSP auf DAGs

Satz:
Distanzen von Single Source Shortest Paths können auf azyklischen Netzwerken in Zeit $O(n + m)$ berechnet werden.

2. nichtneg. Kantenkosten ($c(e) \geq 0 \forall e \in E$)

Name: Dijkstra Algorithmus

- Starte an s. Halte Mengen $S = \{v \in V | d(v) = \delta(v)\}$, sowie $S' = \{v \in V \backslash S \mid v \text{ hat Nachbar in } S\}$
- Vergrößere S sukzessive, bis $S = V$



Dijkstra Algorithmus

$S \leftarrow \{s\}; d(s), d'(s) \leftarrow 0;$
$S' \leftarrow \texttt{OutAdj}(s)$; for all $u \in S'$ $\{d'(u) = c(s,u);\}$
for all $(u \in V \backslash (S' \cup \{s\}))$ $\{d'(u) \leftarrow \infty;\}$

while $(S \neq V)$ {
  wähle $w \in S'$ geeignet // ('`findmin`')
  $d(w) \leftarrow d'(w);$
  $S \leftarrow S \cup \{w\};\ S' \leftarrow S' \backslash \{w\};$

  for all $(u \in \texttt{OutAdj}(w))$ {
    if $(u \notin (S \cup S'))$ $\{S' \leftarrow S' \cup \{u\};\}$
    $d'(u) \leftarrow min\{d'(u), d'(w) + c(w,u)\}$
  }
}

d' sind Distanzzwischenwerte

Dijkstra Algorithmus: Korrektheit

Lemma:
Wird $w \in S'$ gewählt, dass $d'(w)$ minimal, ist $d'(w) = \delta(w)$.

Beweis:
Sei $p = (s, s_1, \ldots, s_k, w)$ billigster Weg von s nach w mit $s_i \in S$. Nimm an, es gäbe einen billigeren Weg q von s nach w. q hat ersten Knoten v in $V \backslash S$.
Es gilt: $d'(v) \geq d'(w)$.
Da alle Kantenkosten nichtnegativ, gilt:
$c(q) \geq d'(v) \geq d'(w) = c(p)$. Wid.

Fazit: Wir können uns also wirklich auf die Menge S' beschränken.

Dijkstra Algorithmus: Laufzeit

Wie implementieren wir S und S', d und d'?

Antwort: S und S' als Bitvektoren der Länge $|V|$, d, d' als Integervektoren

'**for all** $(u \in \texttt{OutAdj}(w))$': Durchlaufen der Adjazenzliste in $O(\sum_w \texttt{OutDeg}(w)) = O(n+m)$

$n \times$ Minimumssuche in S': insgesamt $O(n^2)$

$\Rightarrow$ **Laufzeit**: $O(n^2 + m)$, ok für dichte Graphen.

Verbesserung:
Minimumssuche geht in $O(\log n)$ jeweils mit S' als Heap.
Einfügen, Minimumssuchen/Löschen in $O(\log n)$

d'-Werte müssen verringert werden! Decrease-Key! Geht!

Laufzeit: $O(n \log n + m \log n)$

Dijkstra Algorithmus

Satz:
Single Source Shortest Paths können auf Netzwerken mit nichtnegativen Kantenkosten in Zeit $O((n+m)\log n)$ berechnet werden.

Bemerkung: Es geht auch in $O(n\log n + m)$.