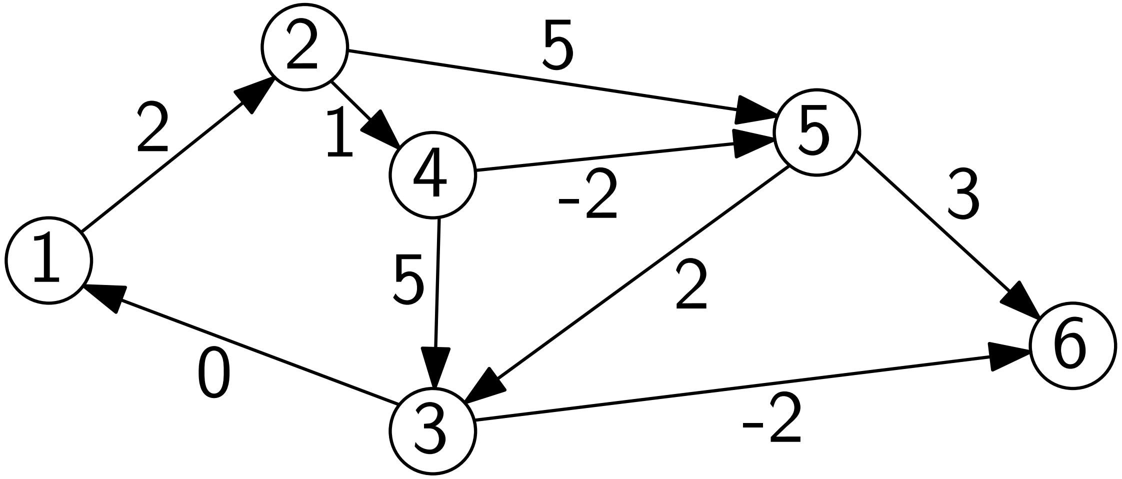
Billigste Wege II

Michael Kaufmann

22+23/11/2021

Rückblick:

Definition, negative Zykel, DAGs, Dijkstra



3. Neg. Kantenkosten erlaubt, keine Zykel

Name: Bellman/Ford - Algorithmus

Idee: benutze iterativ Operation $Relax(v, w)$ durch
$d(w) \leftarrow \min\{d(w), d(v) + c(v, w)\}$

Beobachtung:

- Relax-Operation erhöht keine d-Werte.
- Auch nach Relax-Operation gilt $d(v) \geq \delta(v)$, wenn es vorher galt.

Algorithmus:

- Sei $d(v) = 0$, falls $v = s$, und $d(v) = \infty$ sonst, für alle v
- **Iteriere**: Relaxiere Kanten

d-Werte erniedrigen sich, bis δ's erreicht.
Frage: Wie schnell? Reihenfolge der Relaxierungen?

Bellman-Ford Algorithmus

Lemma:
Sei $w \in V$ mit $\delta(w) < \infty$. Sei (v, w) die letzte Kante auf billigstem Pfad von s nach w. Dann gilt:
Falls (v, w) relaxiert wird, und vorher $d(v) = \delta(v)$, ist danach $d(w) = \delta(w)$.

Algorithmus:

$d(s) \leftarrow 0$
for all $(v \neq s)$ $\{d(v) \leftarrow \infty;\}$
for $(i \leftarrow 1$ to $n - 1)$ $\{$
    for all $((v, w) \in E)$ $\{$
        Relax(v, w)
    $\}$
$\}$

Bellman-Ford Algorithmus

Lemma:
Für $i = 0, ..., n-1$ gilt: Nach Phase i ist $d(w) = \delta(w)$ für alle $w \in V$, für die es einen billigsten Pfad von s nach w der Länge i gibt.

Beweis: Per Induktion

$i = 0$: $\quad d(s) = \delta(s) = 0 \quad$ (kein neg. Zykel)

$i \leftarrow i+1$: $\quad w$ ist Knoten mit billigstem Weg der Länge $i+1$ mit letzter Kante (v, w). Also gibt es billigsten Weg von s nach v der Länge i.

Nach Induktionsannahme gilt nach Phase i: $d(v) = \delta(v)$.
In Phase $i+1$ wird (v, w) relaxiert:
$d(w) = \min\{d(v) + c(v, w)\} = \delta(v) + c(v, w) = \delta(w)$.

Bellman-Ford Algorithmus

Satz:
Nach Phase $n - 1$ gilt für alle $v \in V$: $d(v) = \delta(v)$.

Laufzeit: $O(n \cdot m)$ (Implementierung ganz einfach!)

4. Negative Zykel erlaubt

Idee:

Bei negativem Zykel erniedrigen sich δ-Werte immer weiter.

1. Führe zuerst $n-1$ Phasen von Bellman-Ford aus, merke letzte d-Werte (d_1).

2. Führe weitere n Phasen aus. Ergibt neue d-Werte (d_2).

Lemma: Sei $w \in V$.

$d_2(w) = d_1(w) \Rightarrow d_1(w) = \delta(w)$.
$d_2(w) < d_1(w) \Rightarrow \delta(w) = -\infty$.

Frage:

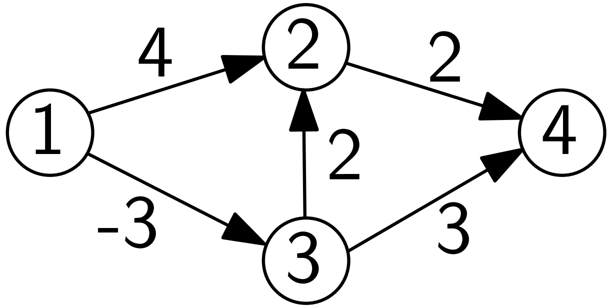
Warum im 2. Schritt n Iterationen und nicht bloss eine?

All Pairs Shortest Paths (APSP)

Name: Floyd/Warshall

Sei (V, E, c) ein Netzwerk, ohne neg. Zykel und sei $V = \{1, 2, ..., n\}$.

Wir definieren für Knoten i, j und $0 \leq k \leq n$ die Kosten $\delta_k(i, j)$ als die des billigsten Weges von i nach j mit inneren Knoten zwischen 0 und k.



Was ist $\delta_0(1, 4)$?, $\delta_2(1, 4)$?, $\delta_4(1, 4)$?

$\delta_0(1, 4) = \infty$, $\delta_2(1, 4) = 6$, $\delta_4(1, 4) = 1$

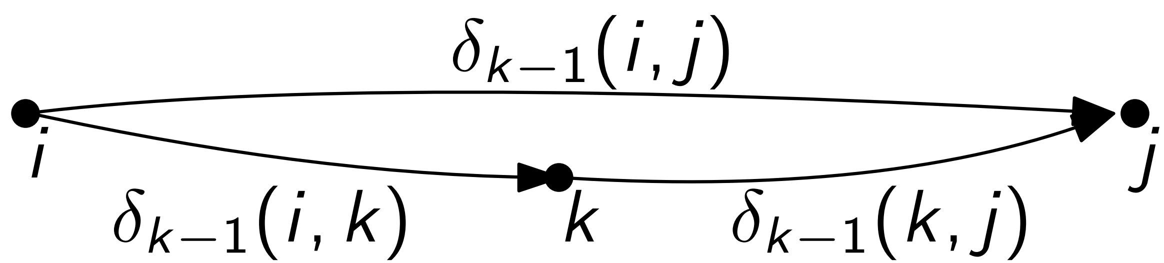
All Pairs Shortest Paths

Wollen $\delta_k(i,j)$ berechnen:

Für $k = 0$ initial: $\delta_0(i,j) = \begin{cases} c(i,j) & \text{falls } (i,j) \in E \\ 0 & \text{falls } i = j \\ \infty & \text{sonst} \end{cases}$

Für $k = n$ gilt: $\delta_n(i,j) = \delta(i,j) =$ Kosten des billigsten Weges von i nach j

Allgemein rekursiv:
$\delta_k(i,j) = \min\{\delta_{k-1}(i,j), \delta_{k-1}(i,k) + \delta_{k-1}(k,j)\}$



Floyd/Warshall's Algorithmus

for all $(i \neq j \in V)$ {
    $\delta_0(i,j) = c(i,j)$ falls $(i,j) \in E$ und $\delta_0(i,j) = \infty$ sonst
}

for all $(i = j)$ {
    $\delta_0(i,j) = 0$
}

for $(k = 1$ to $n)$ {
    for all $(i, j \in V)$ {
        $\delta_k(i,j) = \min\{\delta_{k-1}(i,j), \delta_{k-1}(i,k) + \delta_{k-1}(k,j)\}$
    }
}

Laufzeit: $O(n^3)$ (3-fache for-Schleife)

Alternativer Algorithmus

1. **Mache $n \times$Bellman/Ford**, jeder Knoten ist 1x die Quelle s
   Laufzeit: $O(n \cdot n \cdot m)$

2. **Besser**: $n \times$Dijkstra $\rightarrow$ Ziel: $O(n \cdot (n \log n + m))$.
   Aber: brauchen dazu nichtnegative Kantenkosten!

Es muss gelten für modifizierte Kosten c':
Lemma:
Für alle $u, v \in V$: Pfad p zwischen u und v billigster Weg für originales $c \Leftrightarrow p$ billigster Weg für Kosten c'.

Wahl der Kantenkosten c':

Wählen $c'(u,v) = c(u,v) + h(u) - h(v)$ für Kanten $(u,v) \in E$, und $h : V \to R$.

Für Pfad $p = (v_0, ..., v_k)$:

$$\begin{aligned} c'(p) &= \sum_i c'(v_i, v_{i+1}) \\ &= \sum_i (c(v_i, v_{i+1}) + h(v_i) - h(v_{i+1})) \\ &= c(p) + h(v_0) - h(v_k). \end{aligned}$$

Somit ist

$c(p) = \delta(v_0, v_k) \Leftrightarrow c'(p) = \delta(v_0, v_k) + h(v_0) - h(v_k)$

und für Zykelfall ($v_0 = v_k$): $c(p) = c'(p)$.

Somit ist das Lemma ok.
Nun wähle h, so dass $c' \geq 0$ für alle Kanten.

Wahl von c':

Wahl von h:

- Erweitere G mit neuer Quelle $s \notin V$ und Kanten (s, v) für alle $v \in V$ mit $c(s, v) = 0$.
- Berechne SSSP von s aus und setze $h(v) = \delta(s, v)$ für alle $v \in V$.

Wieso ist das gut?

Dann gilt: $h(v) \leq h(u) + c(u, v)$ für alle $(u, v) \in E$.
und somit $0 \leq c(u, v) + h(u) - h(v) = c'(u, v)$.

Berechne h mit $1\times$Bellman/Ford in $O(nm)$
und dann $n\times$Dijkstra in $O(n \cdot (n \log n + m))$.

Zusammenfassung: APSP

Satz:
Das All-Pairs-Shortest-Paths Problem kann in Zeit $O(n^2 \log n + nm)$ gelöst werden.