

Algoritmen & Datenstrukturen

Herfst 2019

Vorlesung 13

Kürzeste Wege, one-to-all (single source), Überblick

G gewichteter oder ungewichteter Graph, $|V|=n$, $|E|=m$
ich schreibe $G = (V, E, c)$ wenn c Gewichtsfunktion

Graph Algorithmus Laufzeit

$G = (V, E)$ Breitensuche $O(m+n)$

↓
ausgewählter

$G = (V, E, c)$ Dijkstra $O((m+n) \log n)$
 $c: E \rightarrow \mathbb{R}^+$

$O(m+n \log n)$ mit
Fibonacci-Heaps, war
nicht in Vorlesung

$G = (V, E, c)$ Bellman-Ford $O(mn)$
 $c: E \rightarrow \mathbb{R}$

- nur one-to-one gibt es nicht da alle one-to-all berechnen
- wenn man weiß das G keine Zyklen hat gilt es in $O(m+n)$ (kopologische Sortierung + DP, Vorlesung 11). Das kann man also immer zuerst prüfen:
 - 1.) DPS \rightarrow Zyklen? + Kop. Sortierung
 - 2.) keine Zyklen \rightarrow DP

Problem: Finde welche Personen in einem sozialen Netzwerk,

"wichtig"? Ein Ausdruck: Person liegt auf vielen kürzesten Wegen zwischen 2 Personen ("betweenness centrality")

braucht all-to-all (all pairs) shortest path

Graph	Algorithmus	Laufzeit
$G = (V, E)$	$n \times$ Breitensuche	$O(mn + n^2)$
$G = (V, E, c)$ $c: E \rightarrow \mathbb{R}^+$	$n \times$ Dijkstra	$O(mn + n^2 \log n)$
$G = (V, E, c)$ $c: E \rightarrow \mathbb{R}$	$n \times$ Bellman-Ford Floyd-Warshall Johnson	$O(mn^2)$ $O(n^3)$ $O(mn + n^4 \log n)$

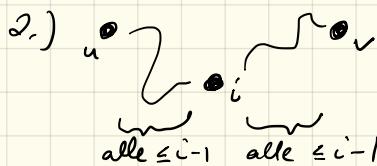
aufwändiger ↓

Floyd-Warshall algorithm: all-to-all (all pairs)
shortest path

Idee: nummeriere Knoten 1..n

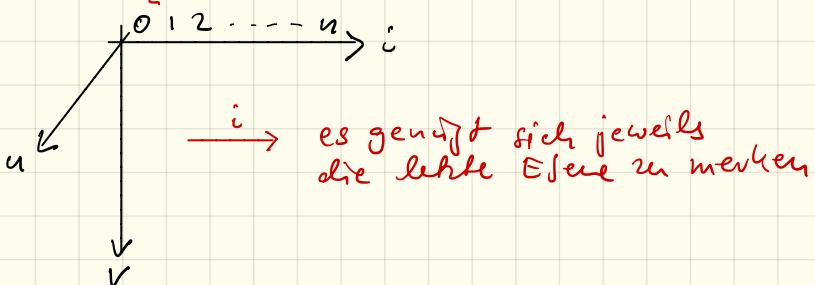
d_{uv}^i = Kosten günstigster Weg $u \rightsquigarrow v$
mit Zwischenknotennummern $\leq i$

JP! $d_{uv}^i = ?$ 2 Möglichkeiten:



$$\Rightarrow d_{uv}^i = \min(d_{uv}^{i-1}, d_{ui} + d_{iv}^{i-1})$$

3D-Tafelan:



$FW(G)$

// initialisiere Tableau

for $u \in V$: $d_{uu}^0 = 0$

for $(u, v) \in E$: $d_{uv}^0 = c(u, v)$; else $d_{uv}^0 = \infty$

// DP

for $i = 1 \dots n$

for $u = 1 \dots n$

for $v = 1 \dots n$

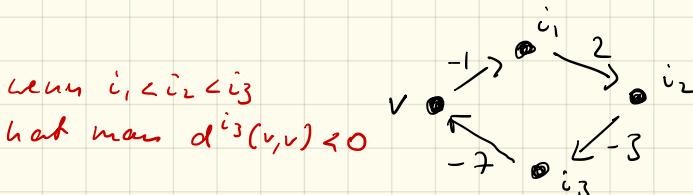
$$d_{uv}^i = \min(d_{uv}^{i-1}, d_{ui}^{i-1} + d_{iv}^{i-1})$$

return d

wenn inplace lässt man den Index weg

Funktioniert wenn keine negativen Zyklen

Test: v ist in negativem Zyklus $\Leftrightarrow d_{vv}^n < 0$



Kürzeste Distanz marken: $\mathcal{O}(n^2)$ Extrapunkt

(ähnlich Fords Algorithmus, Vorlesung 11)

$\Rightarrow \mathcal{O}(n^3)$

Variante: Transitive Closure für Relationen

$G = (V, E)$ beschreibt Relation ρ auf V :

$$u \rho v \Leftrightarrow (u, v) \in E$$

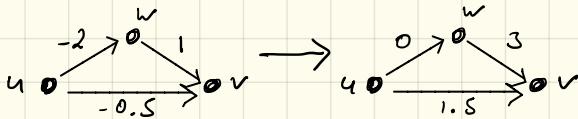
$$d_{uv}^i = d_{uv}^{i-1} \vee (d_{ui}^{i-1} \wedge d_{iv}^{i-1})$$

hier: $d_{uv}^i = v$ ist erreichbar von u mit Knoten $\leq i$

Johnson's Algorithm

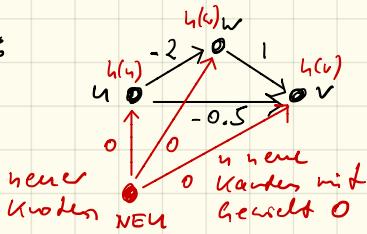
Idee: mache alle Kanten gewichtete positiv
dann mit Dijkstras

Ausatz 1: Subtrahierend kleinstes Gewicht



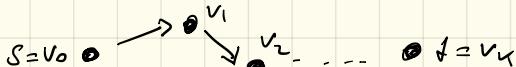
ändert den kürzesten Weg (hier: $u \rightsquigarrow v$) da
Subtraktion von Anzahl der Pfeile abhängt

Ausatz 2:



- mache zu jedem Knoten v eine "Höhe" $h(v)$
- neue Gewichte:
 $\hat{c}(u, v) = c(u, v) + h(u) - h(v)$
- Ziel: $\hat{c}: E \rightarrow \mathbb{R}^+$

Zweck: 1.) Blätter kürzeste Wege kürzen:



$$\text{Länge vorher: } c(s \rightsquigarrow t) = \sum_{i=0}^{k-1} c(v_i, v_{i+1})$$

$$\begin{aligned} \text{Länge jetzt: } \hat{c}(s \rightsquigarrow t) &= \sum_{i=0}^{k-1} \hat{c}(v_i, v_{i+1}) \\ &= \sum_{i=0}^{k-1} (c(v_i, v_{i+1}) + h(v_i) - h(v_{i+1})) \end{aligned}$$

Hängt nur von
s und t ab

$$= c(s \rightsquigarrow t) + h(s) - h(t)$$

2.) Zykluskosten bleiben: $\hat{c}(s \rightsquigarrow s) = c(s \rightsquigarrow s)$

Wie definieren wir h so daß \hat{c} positiv?

$h(u) = \text{Länge kürzester Weg NEU} \rightarrow u$

Dann dann für jede $(u, v) \in E$:

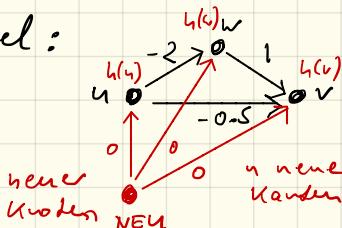
$$h(v) \leq h(u) + c(u, v)$$

$$\Leftrightarrow c(u, v) + h(u) - h(v) \geq 0$$

$$\Leftrightarrow \hat{c}(u, v) \geq 0$$



In Beispiel:

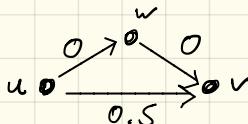


$$h(u) = 0$$

$$h(v) = -1$$

$$h(w) = -2$$

\Rightarrow



Analyse:

Nener Knoten und Pfade	$O(n)$
h -Werte: Bellman-Ford	$O(nm)$
n Mal Dijkstra	$O(nm + n^2 \log n)$

Insgesamt $O(nm + n^2 \log n)$

(besser als Floyd-Warshall
für dünn gesetzte Graphen)

Floyd-Warshall nochmal:

$$d_{uv} = \min(d_{uv}, d_{ui} + d_{iv})$$

$$d_{uv} = d_{uv} \vee (d_{ui} \wedge d_{iv})$$

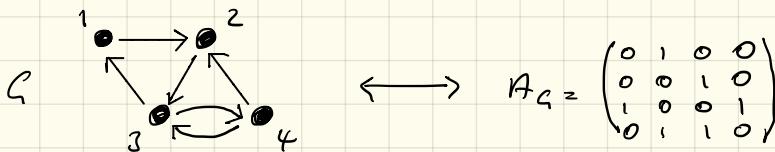
$$d_{uv} = d_{uv} + d_{ui} \cdot d_{iv}$$

// all pairs shortest path

// transitive closure

// Matrix multiplikation

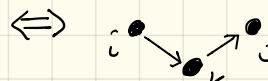
interessant! hat Matrixmultiplikation eine Bedeutung für Bspn?



$$A_G^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}, \quad A_G^3 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 1 & 1 \end{pmatrix}$$

Bedeutung?

$$A_G^2 = B = [s_{ij}] \quad s_{ij} = \sum_{k=1}^n \underbrace{a_{ik} a_{kj}}_{=1 \Leftrightarrow a_{ik}=1 \text{ und } a_{kj}=1}$$



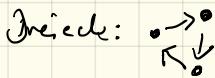
zählt Wege der Länge 2!

Satz: Das Element (i,j) in A_G^k ist die Anzahl der Wege $i \rightarrow j$ der Länge k .

Zewer: Inklusiver über k . Übung!

Anderer herum: Potenzierung von A_G ist ein DP zur Lösung von:
 Wiederhole Weg gibt es von u nach v , für alle $u, v \in V$

Anwendung 1: Anzahl Dreiecke im gewichteten Graphen ohne Schleifen ($\Leftrightarrow (u, u) \notin E$)

Dreieck: 

$$\text{Spur}(A_G^3) / 3$$

Anwendung 2: Kürzester Weg von i nach j ?

Potenziere A_G bis Eintrag (i, j) $\neq 0$
 Potenziieren bis $n-1$ reicht, also
 $O(n)$ Matrix multiplikation erforderlich

\Rightarrow Laufzeit $O(n^4)$

all-pairs shortest path: auch $O(n^4)$

nicht konspektiv

oder ggf Matrix multiplikationen vielfach besser als $\Theta(n^3)$?

Matrixmultiplikation nach Strassen (1969)

ähnliche Idee wie Karatsuba

Blöcke für Rekurrenz: alle Matrixen $n \times n$

$$\begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array} \cdot \begin{array}{|c|c|} \hline e & f \\ \hline g & h \\ \hline \end{array} = \begin{array}{|c|c|} \hline u & v \\ \hline w & x \\ \hline \end{array}$$

A B C

$$\left. \begin{array}{l} u = ae + bf \\ v = af + bd \\ w = ce + dg \\ x = cf + dh \end{array} \right\} \begin{array}{l} 8 \text{ Matrizen} \\ \text{halber Grösse} \\ \text{Schnell nicht} \end{array} \quad T(n) = 8T(\frac{n}{2}) + c \cdot n^2 = \Theta(n^3)$$

Schassen:

$$t_1 = (a+d)(e+h)$$

$$t_2 = (c+d)e$$

$$t_3 = a(f-h)$$

7 Matrizen
halber Grösse

$$t_4 = d(g-e)$$

$$t_5 = (a+b)h$$

$$t_6 = (c-a)(e+f)$$

$$t_7 = (b-d)(g+h)$$

$$u = t_1 + t_4 - t_5 + t_7$$

$$v = t_3 + t_5$$

$$w = t_2 + t_4$$

$$x = t_1 - t_2 + t_3 + t_6$$

$$= a(f-h) + (a+b)h = af + bh \quad \checkmark$$

$$T(n) = 7T(\frac{n}{2}) + c \cdot n^2 = \Theta(n^{\log_2 7}) \approx \Theta(n^{2.8...})$$