

Dynamic Shortest Path

VO Algorithm Engineering

Professor Dr. Petra Mutzel

Lehrstuhl für Algorithm Engineering, LS11

10. VO

28.11.2005

Überblick

Probleme im Voll-Dynamischen Fall

Einführung von Locally Historical Paths

Eigenschaften von LHPs

Algorithmus Fully-Dynamic APSP

Analyse: Korrektheit + Laufzeit

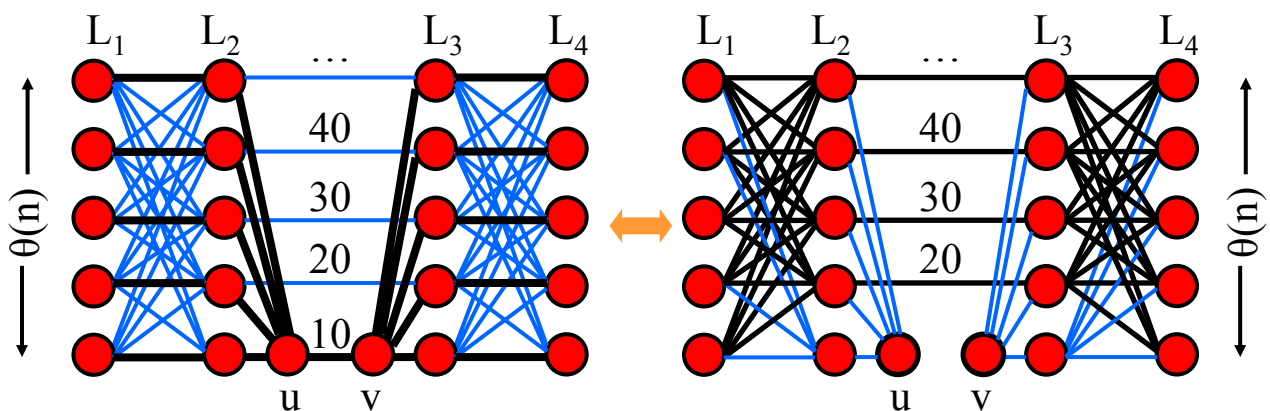
Dynamische APSP

- Geg.: Gerichteter Graph $G=(V,E)$ mit nicht-negativen Kantenkosten $w(e) \in \mathbb{R}$, sowie eine Sequenz der folgenden Operationen:
 - **update(v,w')**: ändere die Kosten aller zu Knoten v inzidenten Kanten zur neuen Kostenfunktion w' (verallgemeinerte Version von dyn. APSP)
 - **distance(x,y)**: gib die Distanz zwischen Knoten x und y zurück
 - **path(x,y)**: gib den kürzesten Weg von x nach y aus, falls einer existiert.

Decrease-only APSP

- Beobachtung: Algorithmus Increase-only APSP funktioniert auch für Decrease-only Sequenzen mit gleicher Laufzeit $O(n^2 \log n)$, denn
 - ein neuer LSP wg. Decrease in Vorwärtsrichtung entspricht einem Herausfallen eines LSP Weges bei Increase in Rückwärtsrichtung.
-
- Decrease-only ist jedoch einfacher lösbar, denn:
 - alle neuen SPs nach $\text{update}(v, w')$ müssen über v laufen
 - löse ein SSSP von v zu allen anderen Knoten und
 - ein SSSP zu v von allen Knoten
 - Falls für ein Knotenpaar die verschmolzenen Teilwege kürzer sind als die vorherigen Wege vor Änderung, dann: NEUER Weg gefunden!

Beispiel für Worst Case



Kanten sind von links nach rechts gerichtet und besitzen kleine Kantengewichte $\epsilon_{ps} > 0$ so dass SP eindeutig und wie in Abb.

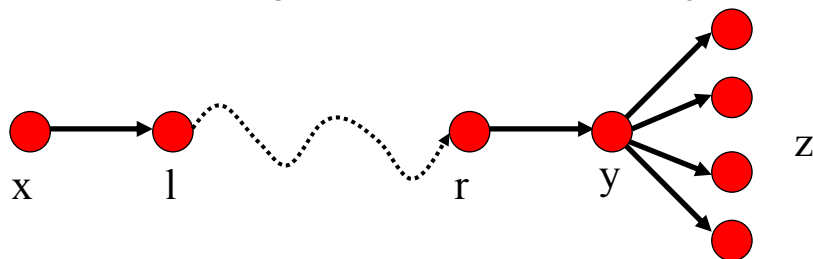
Update-Sequenz: Iteratives Einfügen und Entfernen von (u,v)
 Betrachte: LSPs zwischen Knoten in L_1 und L_4

→ führt zu $\theta(n^3)$ Änderungen in der Menge \mathcal{LSP}

→ Aktualisierung von \mathcal{LSP} nach jedem Update zu teuer

Locally Historical Paths

Idee: Aktualisierung von \mathcal{LSP} nicht nach jedem Update



Betrachte vertex update auf v zur Zeit t' :

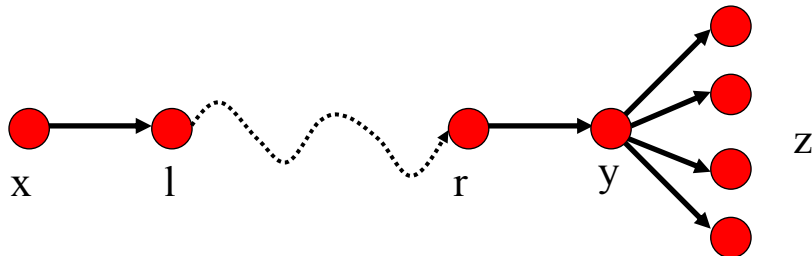
- Laut Algorithmus Increase-only APSP werden in `cleanup()` alle Wege aus DS entfernt, die v enthalten. In `fixup()` werden diese wieder neu (korrekt) eingefügt.
- Betrachte nun einen neuen SP-Weg $P(x \rightarrow y)$

Betrachte nun die Operation `Decrease(w)` von $w \notin P(x \rightarrow y)$ zum Zeitpunkt $t > t'$.

- Evtl. ist $P(x \rightarrow y)$ nun kein SP-Weg mehr

Locally Historical Paths

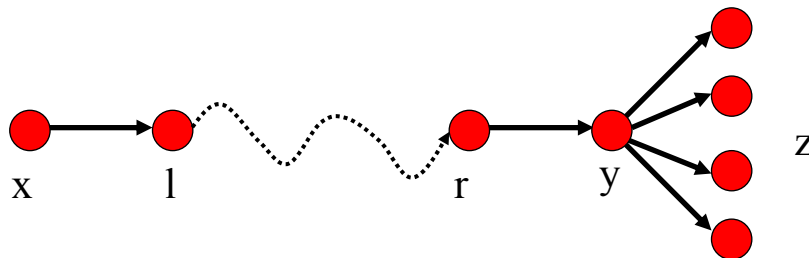
Idee: Aktualisierung von \mathcal{LSP} nicht nach jedem Update



Sei $P(x \rightarrow y)$ ein Weg in G zur Zeit t , und sei $t' \leq t$ der Zeitpunkt des letzten vertex updates von $P(x \rightarrow y)$.

Wir bezeichnen $P(x \rightarrow y)$ als **historisch (historical)** zur Zeit t , wenn er im Zeitintervall $[t', t]$ mindestens einmal kürzester Weg gewesen ist.

Locally Historical Paths



- Ein Weg $P(x \rightarrow z)$ heißt „Lokal Historischer Weg“ („Locally Historical Path“, LHP) in G zur Zeit t , falls entweder
 - $P(x \rightarrow z)$ aus einem einzigen Knoten besteht, oder
 - jeder echte Teilweg von $P(x \rightarrow z)$ ein historischer Weg in G zur Zeit t ist.

- Achtung: LHP Wege müssen nicht unbedingt irgendwann LSP Wege gewesen sein.

- Denn: Im Gegensatz zu LSP, müssen die Teilwege nicht zur gleichen Zeit kürzeste Wege sein.

Eigenschaften von LHPs (1)

Lemma: Bezeichnen SP , HP bzw. LHP die Menge aller SPs, HPs bzw. LSPs in G zur Zeit t in einer Sequenz von Updates. Dann gilt: $SP \subseteq HP \subseteq LHP$.

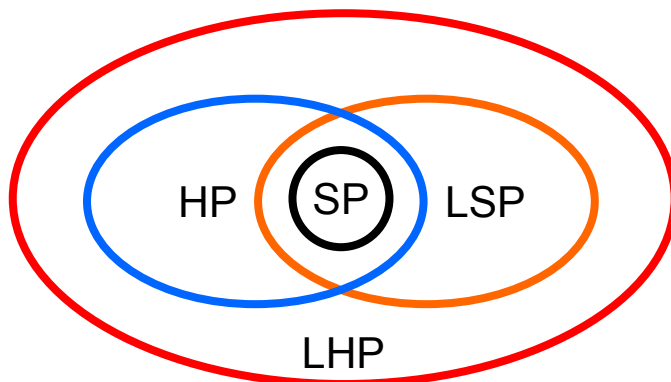
Beweis: $SP \subseteq HP$: aus Definition.

$HP \subseteq LHP$: Sei P ein HP zur Zeit t und sei $t' \leq t$ der Zeitpunkt des letzten vertex updates auf P . Für jeden echten Teilweg $P^i \subset P$, sei $t^i \leq t' \leq t$ der Zeitpunkt des letzten vertex updates von P^i .

Laut Def. war P mind. einmal kürzester Weg im Intervall $[t', t]$. Wegen der optimalen Teilweg Eigenschaft war P^i auch mind. einmal kürzester Weg im Intervall $[t^i, t] \supseteq [t', t]$, also historisch zur Zeit t . Daraus folgt $HP \subseteq LHP$.

Eigenschaften von LHPs (1)

Lemma: Bezeichnen SP , HP bzw. LHP die Menge aller SPs, HPs bzw. LSPs in G zur Zeit t in einer Sequenz von Updates. Dann gilt: $SP \subseteq HP \subseteq LHP$.



Eigenschaften von LHPs (2)

- Sei G ein Graph mit Update-Sequenz Σ . Wenn zu jeder Zeit t höchstens z HPs zwischen jedem Knotenpaar existieren und $G(t)$ m Kanten besitzt, dann können zur Zeit t höchstens zmn LHPs existieren.

Beweis: Fixiere Kante (x,u) und einen Knoten y . Da jeder echte Teilweg eines LHPs historisch sein muss, existieren höchstens z historische Wege zwischen u und y .

Es gibt m Möglichkeiten zur Wahl der ersten Kante und n Möglichkeiten zur Wahl des letzten Knotens.

Eigenschaften von LHPs (3)

- Sei G Graph mit einer Update-Sequenz Σ und seien x, y zwei Knoten in G und sei $\sigma(P)$ der letzte Knoten-Update auf Weg P und $v(\sigma)$ der dazugehörige Knoten des Updates σ .

Falls SP eindeutig sind, gilt für je zwei verschiedene historische Wege $P=P(x \rightarrow y)$ und $P'=P'(x \rightarrow y)$ in G zu jeder Zeit t : $v(\sigma(P)) \neq v(\sigma(P'))$.

- Beweis Indirekt: Annahme: Es gilt Gleichheit.
- Beide Wege waren mind. einmal SP im Intervall $[t(\sigma(P)), t]$.
- Allerdings wird keiner im Intervall $(t(\sigma(P)), t]$ berührt
- Widerspruch zu: Eindeutigkeit von SP.

Eigenschaften von LHPs (4)

- Sei G Graph mit einer Update-Sequenz Σ und sei v ein Knoten in G . Falls SP eindeutig sind und es zu jeder Zeit höchstens z HPs zwischen jedem Knotenpaar gibt, dann:
 - gibt es höchstens $O(zn^2)$ LHPs mit v als Endknoten.

Beweis: Betrachte Wege der Form (v, x, \dots, y) : Es gibt $O(n^2)$ Möglichkeiten für x und y ; es gibt $\leq z$ verschiedene $(x \rightarrow y)$ HP Wege.

Eigenschaften von LHPs (4)

- Sei G Graph mit einer Update-Sequenz Σ und sei v ein Knoten in G . Falls SP eindeutig sind und es zu jeder Zeit höchstens z HPs zwischen jedem Knotenpaar gibt, dann:
 - gibt es höchstens $2z$ LHPs mit v als internen Knoten.

Beweis: Sei \mathcal{P} Menge der LHP Wege der Form (x, \dots, v, \dots, y) und $\mathcal{Q} \subseteq \mathcal{P}$ der Form $Q_i = (x, \dots, v_i, \dots, v, \dots, y)$, wobei v_i ist der letzte update vertex von Q_i (außer x, y).

Betrachte \mathcal{Q}' : Menge der Teilwege der Form $(x, \dots, v_i, \dots, v)$ in \mathcal{Q} . Beh.: $|\mathcal{Q}| = |\mathcal{Q}'|$.

Denn: Betrachte Teilwege der Form (v_i, v, \dots, y) . Wenn es verschiedene solche Wege gäbe, dann Zeitpunkt des letzten Updates aller solcher Wege war v_i bzw. y .
Widerspruch zu vorherigem Lemma (S. 12).

Eigenschaften von LHPs (4)

- Sei G Graph mit einer Update-Sequenz Σ und sei v ein Knoten in G . Falls SP eindeutig sind und es zu jeder Zeit höchstens z HPs zwischen jedem Knotenpaar gibt, dann:
 - gibt es höchstens $2z$ LHPs mit v als internen Knoten.

Beweis: Sei \mathcal{P} Menge der LHP Wege der Form (x, \dots, v, \dots, y) und $\mathcal{Q} \subseteq \mathcal{P}$ der Form $Q_i = (x, \dots, v_i, \dots, v, \dots, y)$, wobei v_i ist der letzte update vertex von Q_i (außer x, y).

Betrachte \mathcal{Q}' : Menge der Teilwege der Form $(x, \dots, v_i, \dots, v)$ in \mathcal{Q} . Beh.: $|\mathcal{Q}| = |\mathcal{Q}'|$. (gezeigt)

Ind. Annahme: $|\mathcal{P}| > 2z$, dann o.E. $|\mathcal{Q}| \geq |\mathcal{P}|/2$, also $|\mathcal{Q}| > z$.

Alle Wege in Menge \mathcal{Q}' sind historisch zur Zeit t (s. Def. \mathcal{P}).

Weil aber $|\mathcal{Q}| = |\mathcal{Q}'|$ gibt es zur Zeit t mehr als z HPs zwischen x und v in G . Widerspruch.

Eigenschaften von LHPs (5)

- Sei G Graph mit einer Update-Sequenz Σ . Falls SP eindeutig sind und es zu jeder Zeit höchstens z HPs zwischen jedem Knotenpaar gibt, dann ist die Anzahl der Wege, die aus der Menge der LHPs per Update herausfallen höchstens $O(zn^2)$.

Beweis: Ein Weg P kann nur durch vertex update an einem Knoten in P aus LHP-Menge herausfallen.

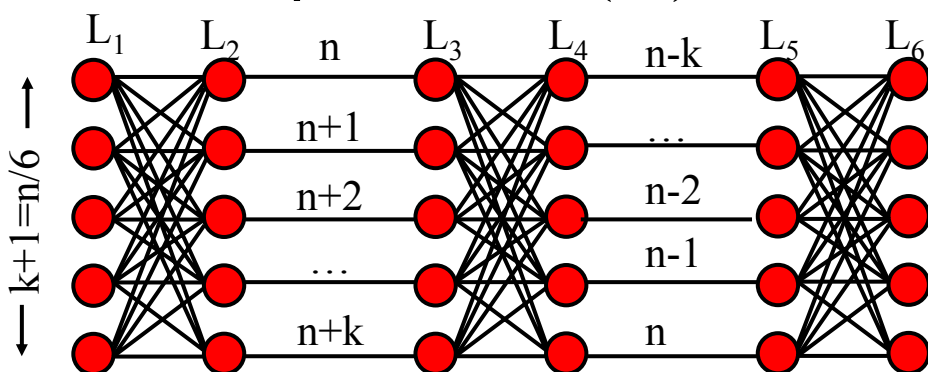
Dann folgt die Behauptung direkt aus den Definitionen und Eigenschaften von LHPs (4).

Eigenschaften von LHPs (6)

- Theorem: Sei G Graph mit einer Update-Sequenz Σ und sei m die maximale Kantenanzahl in G . Falls SP eindeutig sind und es zu jeder Zeit höchstens z HPs zwischen jedem Knotenpaar gibt, dann ist die Anzahl der Wege, die neu in die Menge der LHPs per Update hineinkommt
 - $O(zmn)$ im Worst Case
 - $O(zn^2)$ amortisiert über $\Omega(m/n)$ Update Operationen

Beweis: wie bei LSPs

Beispiel mit $\Omega(n^3)$ HPs



Kanten von links nach rechts gerichtet und Kosten 1 bzw. Bild.

Phase 1 (L_2, L_3): Phase 2 (L_4, L_5): Update-Sequenz: Ph1, Ph2,

(1) $n+1 \rightarrow n-1$ (1) $n-k \rightarrow n+k$

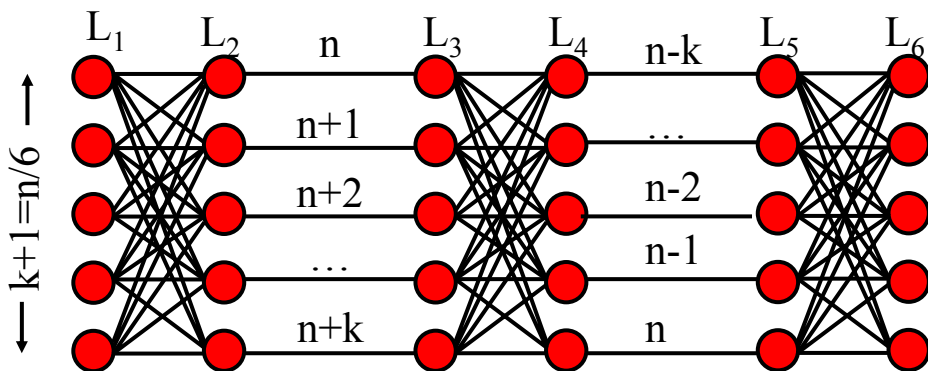
(2) $n+2 \rightarrow n-2$ (2) $n-(k-1) \rightarrow n+(k-1)$

... ..

(k) $n+k \rightarrow n-k$ (k) $n-1 \rightarrow n+1$

Phase 1: $(k+1)^2$ Wege zwischen L_1 und L_5 werden HP je
Decrease \rightarrow führt zu $\theta(k^3)$ Änderungen in der Menge \mathcal{HP} 8

Beispiel mit $\Omega(n^3)$ HPs



Kanten von links nach rechts gerichtet und Kosten 1 bzw. Bild.

Phase 1 (L_2, L_3): Phase 2 (L_4, L_5): Update-Sequenz: Ph1, Ph2,

(1) $n+1 \rightarrow n-1$ (1) $n-k \rightarrow n+k$

(2) $n+2 \rightarrow n-2$ (2) $n-(k-1) \rightarrow n+(k-1)$

... ..

(k) $n+k \rightarrow n-k$ (k) $n-1 \rightarrow n+1$

Alle LHPs zu aktualisieren
kann zu lange dauern:
 $\Omega(n^3)$ per Update

Phase 2: $\Omega(k)$ neue LHP-Wege zwischen jedem Knotenpaar
per Increase $\rightarrow \Omega(n^3)$ Änderungen

Idee: Glättung (Smoothing)

- Alle LHPs in DS zu halten kann zu lange dauern
- Aber auch: keine Information über die Historie zu haben (LSPs) kann auch zu lange dauern.
- Idee: Finde eine Lösung, die dazwischen liegt:
- Reduziere die Anzahl der LHPs
- Dies gelingt durch „Schein“-Updates (Cleanup Updates) von Wegen.
- Idee: Transformiere die gegebene Update Sequenz Σ in eine geglättete Sequenz $F(\Sigma)$, die Cleanup-Updates enthält.
- Ein Cleanup-Update ändert die Kosten nicht.
- Es werden zur Zeit t immer ein Update und mehrere Cleanup-Updates gemacht

Datenstrukturen

- $P(x,y)$: Menge aller LHPs von x nach y (in PrioQ)
- $P^*(x,y)$: Menge aller HPs von x nach y (in PrioQ)
- $L(P(x,y))$: Liste der möglichen $P(x,y)$ path extension Wege nach vorn (*left*), die LHP sind
- $L^*(P(x,y))$: Liste der möglichen $P(x,y)$ path extension Wege nach vorn (*left*), die HP sind
- $R(P(x,y))$: Liste der möglichen $P(x,y)$ path extension Wege nach hinten (*right*), die LHP sind
- $R^*(P(x,y))$: Liste der möglichen $P(x,y)$ path extension Wege nach hinten (*right*), die HP sind
- $t(v)$: Speichert für jeden Knoten Zeitpunkt des letzten Knoten-Updates

21

Algorithmus Fully-Dynamic APSP

Fully-Update(v, w')

- $\text{time} \leftarrow \text{time} + 1$
- $t(v) \leftarrow \text{time}$
- $\text{Update}(v, w')$
- Für jeden Knoten $u \in V$:
 - Falls $\log_2(\text{time} - t(u))$ ganzzahlig ist: $\text{Update}(u, w)$

Wenn Knoten v zur Zeit $t(v)$ einen echten Knoten-Update hatte, dann erhält er zu den folgenden Zeiten einen Cleanup-Update:

$t+1, t+2, t+4, t+8, \dots, t+2^p$

Zur Erinnerung: Algorithmus Update()

Algorithmus Update(v, w'):

1. Cleanup(v): entfernt alle Wege, die v enthalten
2. Fixup(v, w'): Addiere alle neuen SP's und LSP's

Analyse Korrektheit

- Falls die kürzeste Wege eindeutig sind, dann werden die Mengen $P(x,y)$ und $P^*(x,y)$ für jedes Knotenpaar x und y korrekt aktualisiert.

Beweis: genau wie für LSP letztes Mal.

Analyse Laufzeit

- Sei Σ eine Update-Sequenz der Länge k . Die geglättete Sequenz $F(\Sigma)$ hat die folgenden Eigenschaften:
 - Es gibt pro Original-Update höchstens $O(\log k)$ von ihm initiierte Schein-Updates (Cleanup-Updates).
 - Die Graphen, die durch Σ und durch $F(\Sigma)$ produziert werden sind gleich.
 - Es gibt zu jedem Zeitpunkt höchstens $O(\log k)$ HPs zwischen jedem Knotenpaar im Graphen mit Sequenz $F(\Sigma)$.

Beweis: Die ersten beiden Behauptungen sind klar.

Dritte Behauptung: z.z. ist: Seien P^1, \dots, P^z HPs zur Zeit t , dann müssen z Updates $\sigma^1, \dots, \sigma^z \in \Sigma$ existieren mit $t(\sigma^i) + 2 \lfloor \log(t - t(\sigma^i)) \rfloor \leq t(\sigma^{i+1}) \leq t$ für $1 \leq i \leq z-1$. (o.Bw.)

Daraus folgt $z = O(\log t) = O(\log k)$

Analyse Laufzeit

- In einer Update-Sequenz von $\Omega(m/n)$ Operationen wird, falls es zu jedem Zeitpunkt höchstens z HPs zwischen jedem Knotenpaar gibt, jede Update-Operation in $O(zn^2 \log n)$ amortisierter Zeit durchgeführt.
- Der Speicherplatz beträgt $O(zmn)$.

Beweis: wie letztes Mal bei LSPs:

- Cleanup(v): maximal $O(zn^2)$ LHPs können durch v laufen; pro Iteration $O(\log n)$: $O(zn^2 \log n)$
- Fixup():
 - Phase 1: $O(n \log n)$
 - Phase 2: $O(n^2 \log n)$
 - Phase 3: $O(zn^2 \log n)$

Analyse Laufzeit

- In einer Update-Sequenz von $\Omega(m/n)$ Operationen wird jede Update-Operation in $O(n^2 \log^3 n)$ amortisierter Zeit und jede Distanz und Weg-Nachfrage in optimaler Zeit durchgeführt.
- Wegen der Glättung existieren bei einer Sequenz von k Updates höchstens $z=O(\log k)$ HPs zwischen jedem Knotenpaar zu jedem Zeitpunkt.
- Jeder Update Aufruf kostet $O(n^2 \log n \log k)$ amortisierte Zeit
- Jeder Fully-Update Aufruf zieht $O(\log k)$ Update-Aufrufe nach sich
- Das sind insgesamt $O(n^2 \log n \log^2 k)=O(n^2 \log^3 n)$ für jede Sequenz der Länge k mit k ist polynomiell in n .
- Restart alle $\theta(n^2)$ Operationen.

Neuere Entwicklungen

- Mikkel Thorup erreicht mit einer neuen Glättungs-Strategie eine Laufzeit von $O(n^2(\log n + \log^2(m/n)))$

Offene Probleme

- Speicherplatzreduktion
- Fully-Dynamic Algorithms für Single-Source Shortest Path?

Literatur für diese VO

- C. Demetrescu und G.F. Italiano: A new approach to dynamic all pairs shortest paths, Proc. 35th Annual ACM Symposium on Theory of Computing (STOC '03), San Diego, 2003, 159-166.
- Zeitschriftenversion: Journal of the Association for Computing Machinery (JACM), vol. 51 (6), 2004, 968-992.

