Vorlesungstermin 11:

Dynamische Programmierung III

Markus Püschel
David Steurer

talks2.dsteurer.org/dp3.pdf

Algorithmen und Datenstrukturen, Herbstsemester 2018, ETH Zürich

Plan für heute

zurück zu den Graphen:

(kürzeste) Wege mit dynamischer Programmierung

Wege in Graphen zählen

gegeben: ger. Graph G, Knoten $u, v \in V(G)$, Länge $L \in \mathbb{N}$

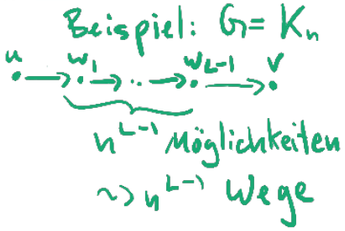
gesucht: Anzahl der Wege in G von u nach v der Länge L

Wege in Graphen zählen

gegeben: ger. Graph G, Knoten $u, v \in V(G)$, Länge $L \in \mathbb{N}$

gesucht: Anzahl der Wege in G von u nach v der Länge L

es kann exponentiell viele solcher Wege geben!

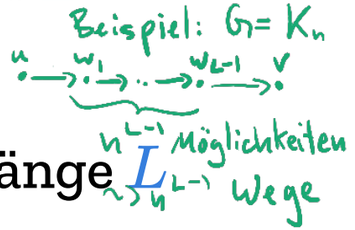


Wege in Graphen zählen

gegeben: ger. Graph G, Knoten $u, v \in V(G)$, Länge $L \in \mathbb{N}$

gesucht: Anzahl der Wege in G von u nach v der Länge L

es kann exponentiell viele solcher Wege geben!



Struktur der Lösungen: jeder Weg von u nach v der Länge L besteht aus einem Weg der Länge $L-1$ zu einem Knoten w und einer Kante von w nach v

Wege in Graphen zählen

gegeben: ger. Graph G, Knoten $u, v \in V(G)$, Länge $L \in \mathbb{N}$

gesucht: Anzahl der Wege in G von u nach v der Länge L

es kann exponentiell viele solcher Wege geben!

Beispiel: $G = K_n$

n^{L-1} Möglichkeiten

L_n^{L-1} Wege

Struktur der Lösungen: jeder Weg von u nach v der Länge L besteht aus einem Weg der Länge $L-1$ zu einem Knoten w und einer Kante von w nach v

Zerlegung in Teilprobleme:
$T^{(\ell)}(i,k) =$ Anzahl der Wege von i nach k der Länge ℓ

Rekurrenz zwischen Teilproblemen:
$T^{(\ell)}(i,k) = \sum_{j:\ (j,k)\in E(G)} T^{(\ell-1)}(i,j)$

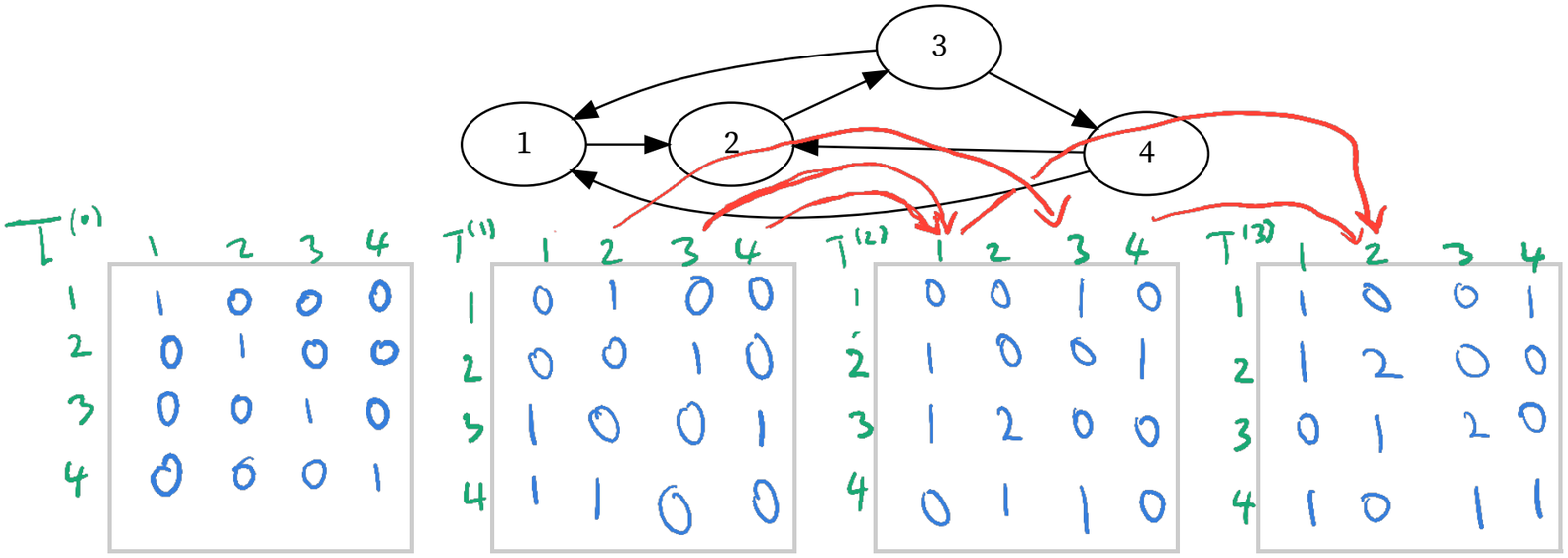
← jeder L-Weg wird in der Summe genau einmal gezählt

Zerlegung in Teilprobleme:

$T^{(\ell)}(i,k) =$ Anzahl der Wege von i nach k der Länge ℓ

Rekurrenz zwischen Teilproblemen:

$T^{(\ell)}(i,k) = \sum_{j:\ (j,k)\in E(G)} T^{(\ell-1)}(i,j)$

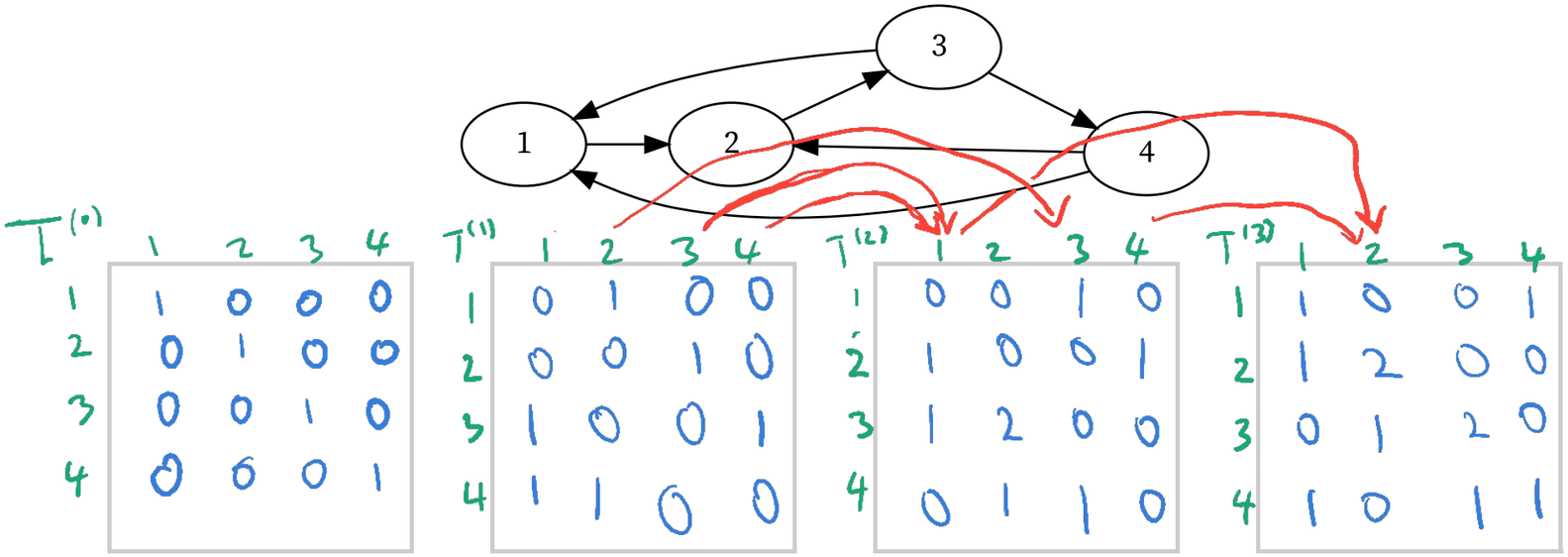


Zerlegung in Teilprobleme:

$T^{(\ell)}(i,k) =$ Anzahl der Wege von i nach k der Länge ℓ

Rekurrenz zwischen Teilproblemen:

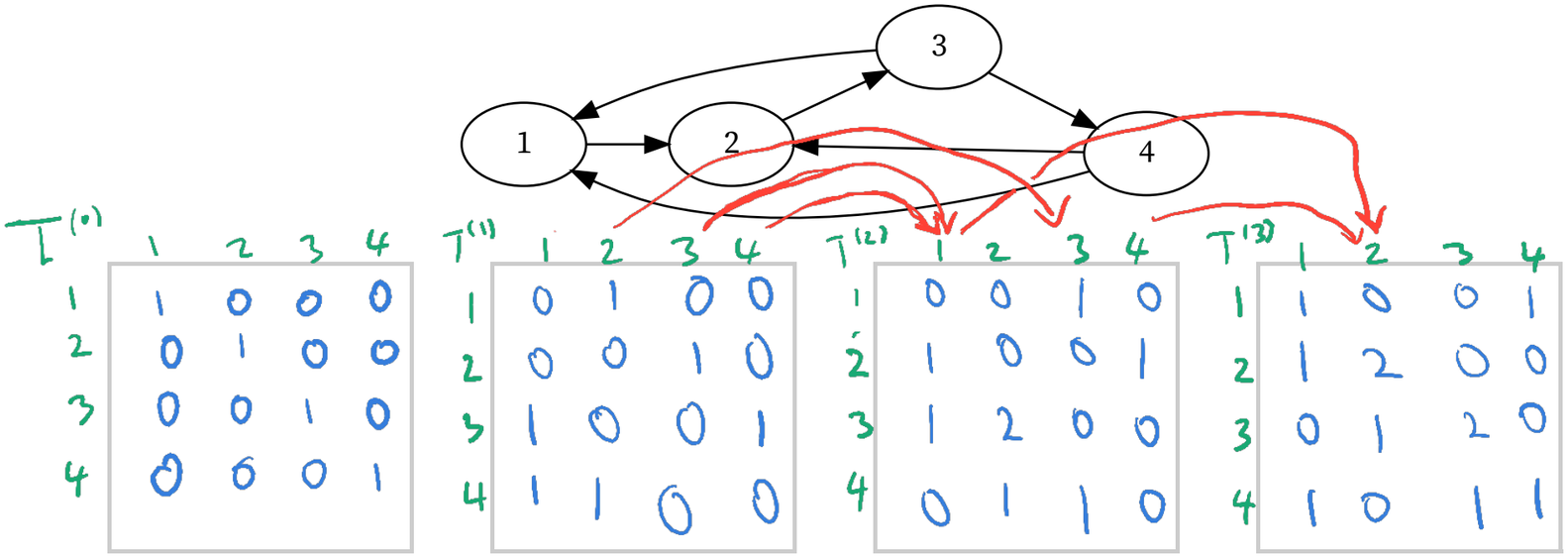
$T^{(\ell)}(i,k) = \sum_{j:\ (j,k)\in E(G)} T^{(\ell-1)}(i,j)$



Beobachtung: $T^{(\ell)}$ ist ℓ-te Potenz der Adjazenzmatrix

Zerlegung in Teilprobleme:
$T^{(\ell)}(i,k) =$ Anzahl der Wege von i nach k der Länge ℓ

Rekurrenz zwischen Teilproblemen:
$T^{(\ell)}(i,k) = \sum_{j:\ (j,k)\in E(G)} T^{(\ell-1)}(i,j)$



Beobachtung: $T^{(\ell)}$ ist ℓ-te Potenz der Adjazenzmatrix

Fazit: Matrix Multiplikation ist einfache Form der dynamischen Programmierung ...

kürzeste Wege in Graphen

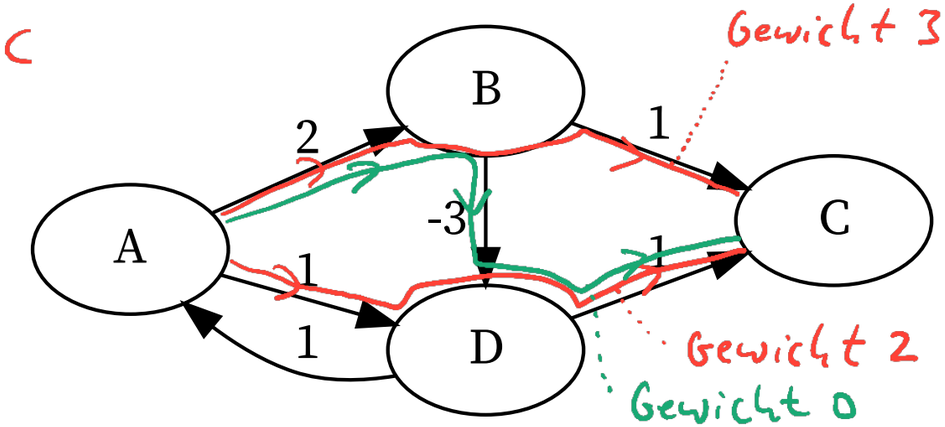
gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

kürzeste Wege in Graphen

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

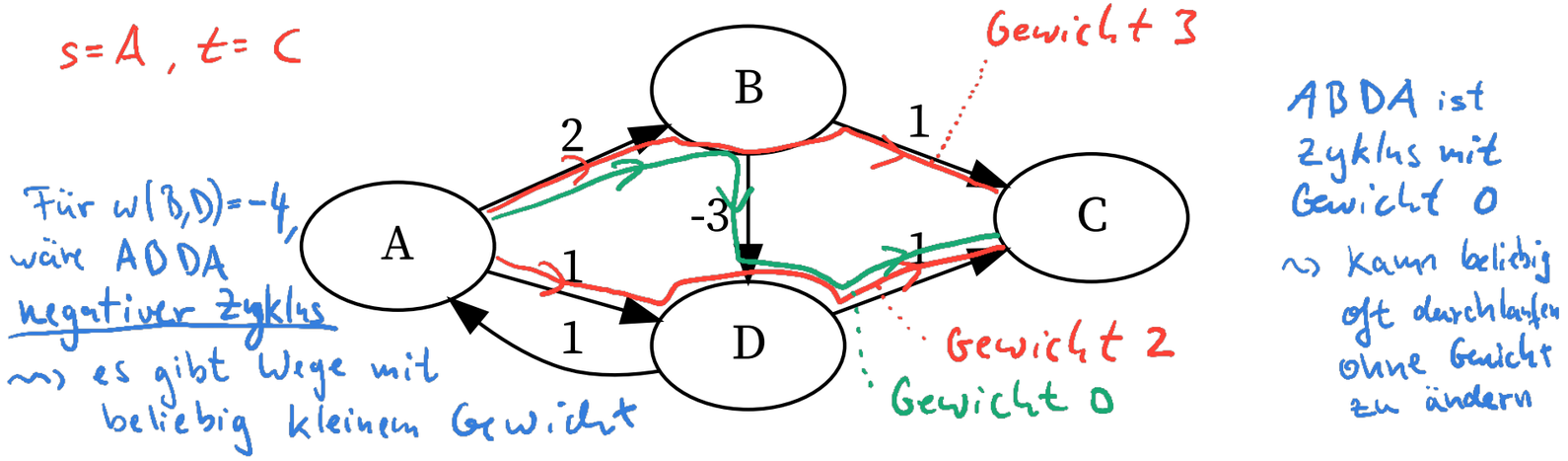
gesucht: Weg von s nach t in G mit kleinstem Gewicht



kürzeste Wege in Graphen

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

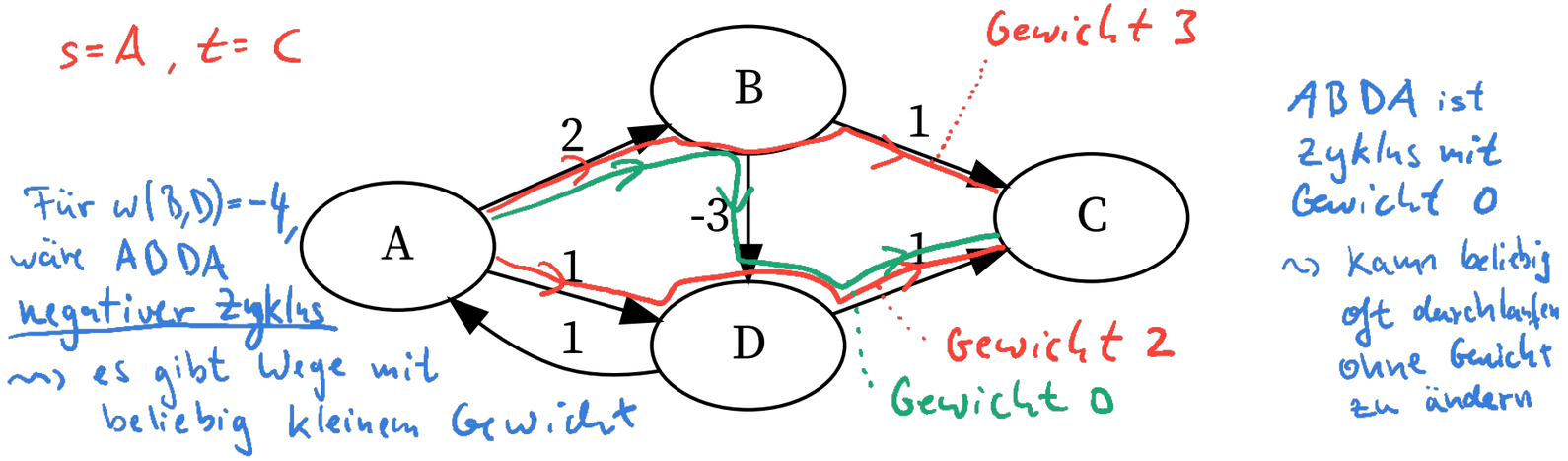
gesucht: Weg von s nach t in G mit kleinstem Gewicht



kürzeste Wege in Graphen

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht



Zur Erinnerung: *Breitensuche* funktioniert, wenn alle Gewichte *positiv* und *gleich* sind (d.h. wir suchen Weg mit wenigsten Kanten)

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w \colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

Struktur: optimaler Weg von s nach t besteht aus optimalem Weg von s zu einem Knoten i und der Kante von i nach t

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

Struktur: optimaler Weg von s nach t besteht aus optimalem Weg von s zu einem Knoten i und der Kante von i nach t

Teilprobleme: $T(j) =$ min. Gewicht eines Wegs von s zu j

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

Struktur: optimaler Weg von s nach t besteht aus optimalem Weg von s zu einem Knoten i und der Kante von i nach t

Teilprobleme: $T(j) =$ min. Gewicht eines Wegs von s zu j

Rekurrenz: $T(j) = \min_{i\colon (i,j)\in E} T(i) + w(i,j)$ ← min wird erreicht falls i Vorgänger von j im optimalen Weg ist

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

Struktur: optimaler Weg von s nach t besteht aus optimalem Weg von s zu einem Knoten i und der Kante von i nach t

Teilprobleme: $T(j) =$ min. Gewicht eines Wegs von s zu j

Rekurrenz: $T(j) = \min_{i\colon (i,j)\in E} T(i) + w(i,j)$ ← min wird erreicht falls i Vorgänger von j im optimalen Weg ist

welche Einträge können wir direkt berechnen?
welche Reihenfolge für die anderen Einträge?

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

Struktur: optimaler Weg von s nach t besteht aus optimalem Weg von s zu einem Knoten i und der Kante von i nach t

Teilprobleme: $T(j) =$ min. Gewicht eines Wegs von s zu j

Rekurrenz: $T(j) = \min_{i\colon (i,j)\in E} T(i) + w(i,j)$ ← min wird erreicht falls i Vorgänger von j im optimalen Weg ist

welche Einträge können wir direkt berechnen?
welche Reihenfolge für die anderen Einträge?

Fehlschlag: Rekurrenz bezieht sich nicht auf *"kleinere" Teilprobleme* ⇝ keine natürliche Reihenfolge ("Rekursion terminiert nicht")

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

Struktur: optimaler Weg von s nach t besteht aus optimalem Weg von s zu einem Knoten i und der Kante von i nach t

Teilprobleme: $T(j) =$ min. Gewicht eines Wegs von s zu j

Rekurrenz: $T(j) = \min_{i\colon (i,j)\in E} T(i) + w(i,j)$ ← min wird erreicht falls i Vorgänger von j im optimalen Weg ist

welche Einträge können wir direkt berechnen?
welche Reihenfolge für die anderen Einträge?

Fehlschlag: Rekurrenz bezieht sich nicht auf *"kleinere" Teilprobleme* ⇝ keine natürliche Reihenfolge ("Rekursion terminiert nicht")

können wir den Ansatz retten?

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

Struktur: optimaler Weg von s nach t besteht aus optimalem Weg von s zu einem Knoten i und der Kante von i nach t

dabei hat der optimale Weg von s zu i eine Kante weniger

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

Struktur: optimaler Weg von s nach t besteht aus optimalem Weg von s zu einem Knoten i und der Kante von i nach t

dabei hat der optimale Weg von s zu i eine Kante weniger

Teilprobleme:
$T(j, \ell)$ = min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten
(= ∞ falls kein solcher Weg existiert)

Rekurrenz:
$$T(j, \ell) = \min\left\{T(j, \ell-1), \min_{(i,j)\in E} T(i, \ell-1) + w(i, j)\right\}$$

gegeben: ger. Graph $G = (V, E)$, Knoten $s, t \in V$,
Gewichte $w\colon E \to \mathbb{R}$

gesucht: Weg von s nach t in G mit kleinstem Gewicht

Struktur: optimaler Weg von s nach t besteht aus optimalem Weg von s zu einem Knoten i und der Kante von i nach t

dabei hat der optimale Weg von s zu i eine Kante weniger

Teilprobleme:
$T(j, \ell)$ = min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten
(= ∞ falls kein solcher Weg existiert)

Rekurrenz:
$$T(j, \ell) = \min\left\{T(j, \ell - 1), \min_{(i,j)\in E} T(i, \ell - 1) + w(i, j)\right\}$$

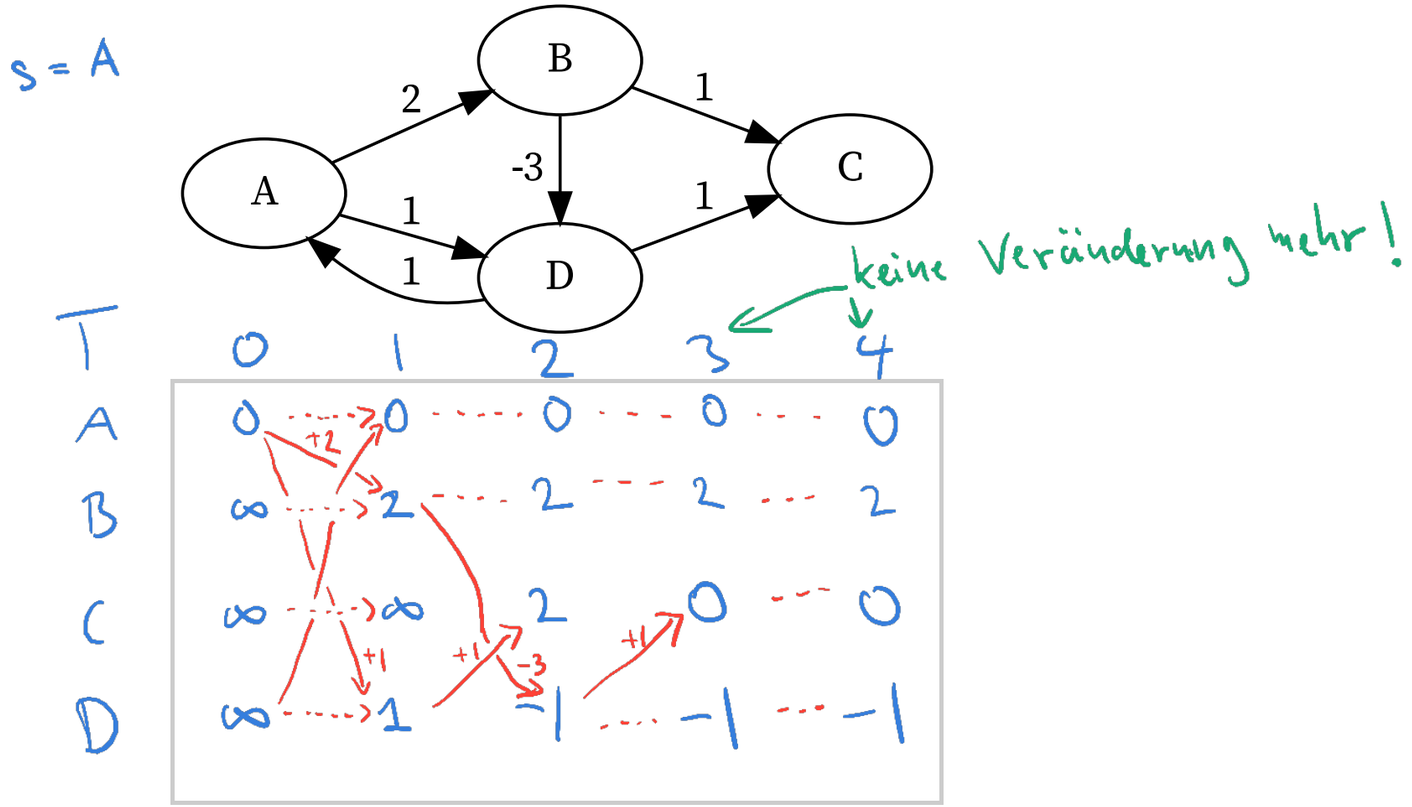
Berechnung: initialisiere Einträge der Form $T(j, 0)$; berechne Einträge der Form $T(j, 1)$ via Rekurrenz; danach $T(j, 2)$ …
[Bellman–Ford Algorithmus]

Teilprobleme:

$T(j, \ell) =$ min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten

Rekurrenz:

$$T(j, \ell) = \min \left\{ T(j, \ell - 1), \min_{(i,j) \in E} T(i, \ell - 1) + w(i, j) \right\}$$



Teilprobleme:

$T(j, \ell) =$ min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten

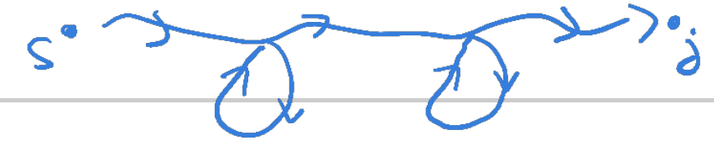
Rekurrenz:

$$T(j, \ell) = \min \left\{ T(j, \ell - 1), \min_{(i,j) \in E} T(i, \ell - 1) + w(i, j) \right\}$$

sei $n = |V|$

Satz: $T(j, 2n) < T(j, n-1)$ genau dann wenn es Wege von s nach j mit *beliebig kleinem (negativen) Gewicht* gibt

angenommen $T(j,2n) < T(j,n-1)$
betrachte opt. Weg mit $2n$ Kanten; besteht aus Pfad von s nach j und (mehreren Zyklen)



da Pfad $\leq n-1$ Kanten hat, gilt
Gewicht(Pfad + Zyklen) < Gewicht(Pfad)

Also: Zyklen haben neg. Gewicht!

Teilprobleme:

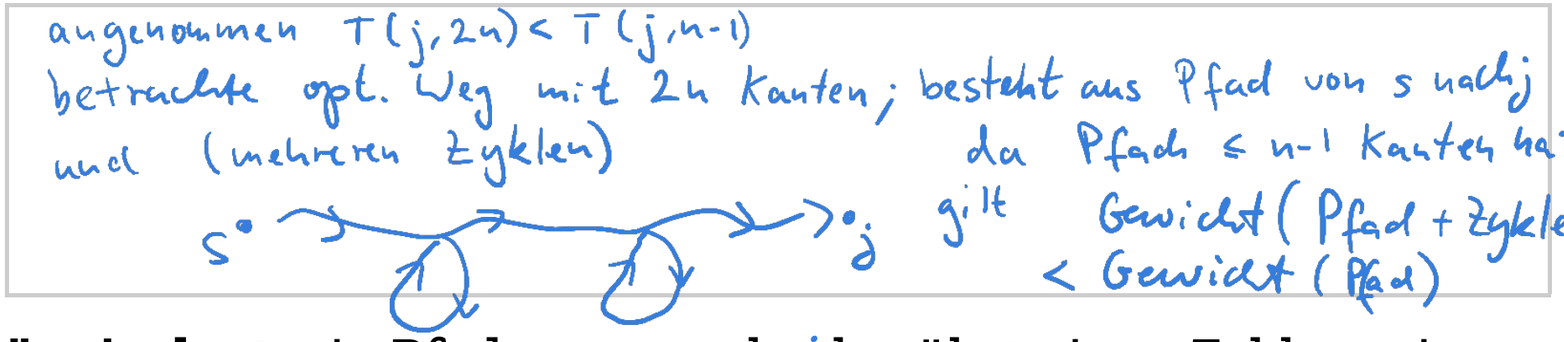
$T(j, \ell) =$ min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten

Rekurrenz:

$$T(j, \ell) = \min \left\{ T(j, \ell - 1), \min_{(i,j) \in E} T(i, \ell - 1) + w(i, j) \right\}$$

sei $n = |V|$

Satz: $T(j, 2n) < T(j, n-1)$ genau dann wenn es Wege von s nach j mit *beliebig kleinem (negativen) Gewicht* gibt



Also: Zyklen haben neg. Gewicht!

äquivalent: ein Pfad von s nach j berührt einen Zyklus mit negativem Gewicht

Teilprobleme:
$T(j, \ell) =$ min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten

Rekurrenz:
$T(j, \ell) = \min \left\{ T(j, \ell - 1), \min_{(i,j) \in E} T(i, \ell - 1) + w(i, j) \right\}$

sei $n = |V|$ und $m = |E|$

Laufzeit: Anzahl der Einträge ist ;

Teilprobleme:
$T(j, \ell) =$ min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten

Rekurrenz:
$T(j, \ell) = \min \left\{ T(j, \ell - 1), \min_{(i,j) \in E} T(i, \ell - 1) + w(i, j) \right\}$

sei $n = |V|$ und $m = |E|$

Laufzeit: Anzahl der Einträge ist $O(n^2)$;

Teilprobleme:
$T(j, \ell) =$ min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten

Rekurrenz:
$T(j, \ell) = \min \left\{ T(j, \ell - 1), \min_{(i,j) \in E} T(i, \ell - 1) + w(i, j) \right\}$

sei $n = |V|$ und $m = |E|$

Laufzeit: Anzahl der Einträge ist $O(n^2)$; Berechnung eines Eintrags $T(j, \ell)$ braucht Schritte

Teilprobleme:
$T(j, \ell) =$ min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten

Rekurrenz:
$T(j, \ell) = \min \left\{ T(j, \ell - 1), \min_{(i,j) \in E} T(i, \ell - 1) + w(i, j) \right\}$

sei $n = |V|$ und $m = |E|$

Laufzeit: Anzahl der Einträge ist $O(n^2)$; Berechnung eines Eintrags $T(j, \ell)$ braucht $O(\deg^-(j))$ Schritte

Teilprobleme:
$T(j, \ell) =$ min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten

Rekurrenz:
$T(j, \ell) = \min \left\{ T(j, \ell - 1), \min_{(i,j) \in E} T(i, \ell - 1) + w(i, j) \right\}$

sei $n = |V|$ und $m = |E|$

Laufzeit: Anzahl der Einträge ist $O(n^2)$; Berechnung eines Eintrags $T(j, \ell)$ braucht $O(\deg^-(j))$ Schritte

naive Abschätzung: $O(n^3)$ Gesamtlaufzeit (da $\deg^-(j) \leqslant n$)

Teilprobleme:
$T(j, \ell) =$ min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten

Rekurrenz:
$T(j, \ell) = \min \left\{ T(j, \ell - 1), \min_{(i,j) \in E} T(i, \ell - 1) + w(i, j) \right\}$

sei $n = |V|$ und $m = |E|$

Laufzeit: Anzahl der Einträge ist $O(n^2)$; Berechnung eines Eintrags $T(j, \ell)$ braucht $O(\deg^-(j))$ Schritte

naive Abschätzung: $O(n^3)$ Gesamtlaufzeit (da $\deg^-(j) \leqslant n$)

bessere Abschätzung: Gesamtlaufzeit $O(\sum_{\ell=1}^{n} \sum_{j \in V} N_{j,\ell})$
wobei $N_{j,\ell} = \deg^-(j)$;

$$\sum_{\ell=1}^{n} \sum_{j \in V} N_{j,\ell} = \sum_{\ell=1}^{n} \underbrace{\sum_{j \in V} \deg^-(j)}_{= m} = \sum_{\ell=1}^{n} m = n \cdot m$$

Teilprobleme:
$T(j, \ell) =$ min. Gewicht eines Wegs von s zu j mit $\leqslant \ell$ Kanten

Rekurrenz:
$T(j, \ell) = \min \left\{ T(j, \ell - 1), \min_{(i,j) \in E} T(i, \ell - 1) + w(i, j) \right\}$

sei $n = |V|$ und $m = |E|$

Laufzeit: Anzahl der Einträge ist $O(n^2)$; Berechnung eines Eintrags $T(j, \ell)$ braucht $O(\deg^-(j))$ Schritte

naive Abschätzung: $O(n^3)$ Gesamtlaufzeit (da $\deg^-(j) \leqslant n$)

bessere Abschätzung: Gesamtlaufzeit $O(\sum_{\ell=1}^{n} \sum_{j \in V} N_{j,\ell})$
wobei $N_{j,\ell} = \deg^-(j)$; da $\sum_{\ell=1}^{n} \sum_{j \in V} N_{j,\ell} = n \cdot m$, hat
Bellman–Ford Laufzeit $O(n \cdot m)$

$$\sum_{\ell=1}^{n} \sum_{j \in V} N_{j,\ell} = \sum_{\ell=1}^{n} \underbrace{\sum_{j \in V} \deg^-(j)}_{= m} = \sum_{\ell=1}^{n} m = n \cdot m$$

alle kürzesten Wege

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

alle kürzesten Wege

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

wir nehmen an $V = \{1, \dots, n\}$

baseline:

alle kürzesten Wege

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

wir nehmen an $V = \{1, \dots, n\}$

baseline: Bellman–Ford berechnet für ein $i \in V$ alle Einträge $D(i, 1), \dots, D(i, n)$ in Zeit $O(n \cdot m)$;
mit n Aufrufen erhalten wir gesamte Tabelle in Zeit $O(n^2 \cdot m)$

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Verbesserungsidee:

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

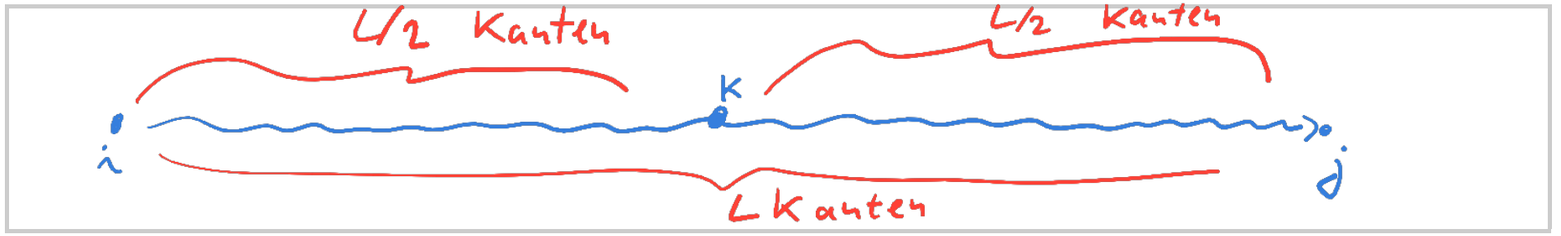
Verbesserungsidee: berechne alle Einträge gleichzeitig und stelle "bessere" Rekurrenz auf

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Verbesserungsidee: berechne alle Einträge gleichzeitig und stelle "bessere" Rekurrenz auf

Struktur: optimaler Weg von i zu j besteht aus optimalen Wegen von i zu k und von k zu j mit *halb so vielen Kanten*

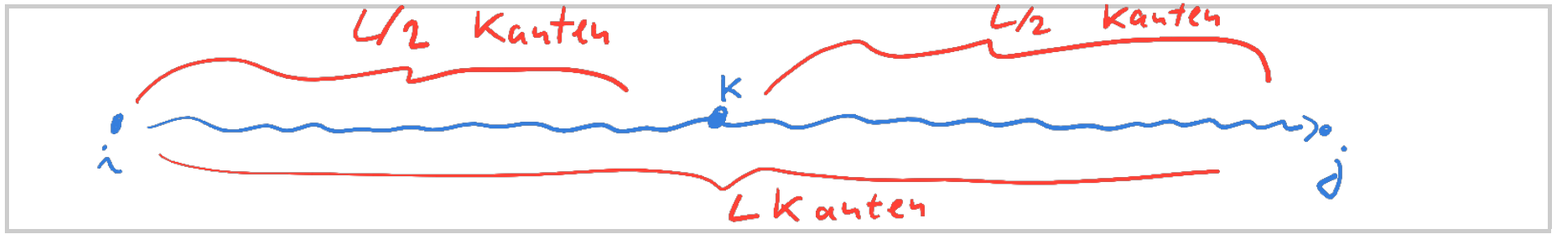


gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Verbesserungsidee: berechne alle Einträge gleichzeitig und stelle "bessere" Rekurrenz auf

Struktur: optimaler Weg von i zu j besteht aus optimalen Wegen von i zu k und von k zu j mit *halb so vielen Kanten*



Teilprobleme:
$T(i, j, r) =$ min. Gewicht eines i-j Wegs mit $\leqslant 2^r$ Kanten

Rekurrenz:
$T(i, j, r) = \min_{k \in V} T(i, k, r-1) + T(k, j, r-1)$

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Teilprobleme:
$T(i, j, r) =$ min. Gewicht eines i-j Wegs mit $\leqslant 2^r$ Kanten

Rekurrenz:
$T(i, j, r) = \min_{k \in V} T(i, k, r-1) + T(k, j, r-1)$

Berechnung:

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Teilprobleme:
$T(i, j, r) =$ min. Gewicht eines i-j Wegs mit $\leqslant 2^r$ Kanten

Rekurrenz:
$T(i, j, r) = \min_{k \in V} T(i, k, r-1) + T(k, j, r-1)$

Berechnung: initialisiere Einträge mit $r = 0$ (wie?);
verwende Rekurrenz für Einträge mit $r = 1$; dann $r = 2$, ...

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Teilprobleme:
$T(i, j, r) =$ min. Gewicht eines i-j Wegs mit $\leqslant 2^r$ Kanten

Rekurrenz:
$T(i, j, r) = \min_{k \in V} T(i, k, r-1) + T(k, j, r-1)$

Berechnung: initialisiere Einträge mit $r = 0$ (wie?);
verwende Rekurrenz für Einträge mit $r = 1$; dann $r = 2$, ...

Laufzeit:

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Teilprobleme:
$T(i, j, r) =$ min. Gewicht eines i-j Wegs mit $\leqslant 2^r$ Kanten

Rekurrenz:
$T(i, j, r) = \min_{k \in V} T(i, k, r-1) + T(k, j, r-1)$

Berechnung: initialisiere Einträge mit $r = 0$ (wie?);
verwende Rekurrenz für Einträge mit $r = 1$; dann $r = 2$, ...

Laufzeit: es reicht aus $r \leqslant \lceil \log n \rceil$ zu betrachten
Anzahl der Einträge ist $O(n^2 \cdot \log n)$
$O(n)$ Schritte pro Eintrag;

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Teilprobleme:
$T(i, j, r) =$ min. Gewicht eines i-j Wegs mit $\leqslant 2^r$ Kanten

Rekurrenz:
$T(i, j, r) = \min_{k \in V} T(i, k, r-1) + T(k, j, r-1)$

Berechnung: initialisiere Einträge mit $r = 0$ (wie?);
verwende Rekurrenz für Einträge mit $r = 1$; dann $r = 2$, ...

Laufzeit: es reicht aus $r \leqslant \lceil \log n \rceil$ zu betrachten
Anzahl der Einträge ist $O(n^2 \cdot \log n)$
$O(n)$ Schritte pro Eintrag;
Gesamtlaufzeit $O(n^3 \cdot \log n)$

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Teilprobleme:
$T(i, j, r) =$ min. Gewicht eines i-j Wegs mit $\leqslant 2^r$ Kanten

Rekurrenz:
$T(i, j, r) = \min_{k \in V} T(i, k, r-1) + T(k, j, r-1)$

Berechnung: initialisiere Einträge mit $r = 0$ (wie?);
verwende Rekurrenz für Einträge mit $r = 1$; dann $r = 2$, ...

Laufzeit: es reicht aus $r \leqslant \lceil \log n \rceil$ zu betrachten
Anzahl der Einträge ist $O(n^2 \cdot \log n)$
$O(n)$ Schritte pro Eintrag;
Gesamtlaufzeit $O(n^3 \cdot \log n)$
(besser als baseline falls $m > n \cdot \log n$)

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Teilprobleme:
$T(i, j, r) =$ min. Gewicht eines i-j Wegs mit $\leqslant 2^r$ Kanten

Rekurrenz:
$T(i, j, r) = \min_{k \in V} T(i, k, r-1) + T(k, j, r-1)$

Berechnung: initialisiere Einträge mit $r = 0$ (wie?);
verwende Rekurrenz für Einträge mit $r = 1$; dann $r = 2$, ...

Laufzeit: es reicht aus $r \leqslant \lceil \log n \rceil$ zu betrachten
Anzahl der Einträge ist $O(n^2 \cdot \log n)$
$O(n)$ Schritte pro Eintrag;
Gesamtlaufzeit $O(n^3 \cdot \log n)$
(besser als baseline falls $m > n \cdot \log n$)

es geht noch besser! [Floyd–Warshall Algorithmus]

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Idee: mehr Teilprobleme; weniger Zeit pro Teilproblem

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Idee: mehr Teilprobleme; weniger Zeit pro Teilproblem

Teilprobleme: $T^{(k)}(i, j) =$ min. Gewicht eines i-j-Pfads mit allen Zwischenknoten in $\{1, \ldots, k\}$

gegeben: ger. Graph $G = (V, E)$, Gewichte $w\colon E \to \mathbb{R}$

gesucht: Tabelle $D(i,j) =$ min. Gewicht eines i-j-Wegs

Idee: mehr Teilprobleme; weniger Zeit pro Teilproblem

Teilprobleme: $T^{(k)}(i,j) =$ min. Gewicht eines i-j-Pfads mit allen Zwischenknoten in $\{1,\dots,k\}$

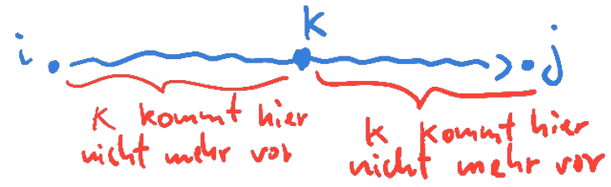
kein Knoten kommt mehrmals vor

Rekurrenz:

$$T^{(k)}(i,j) = \min\{T^{(k-1)}(i,j), T^{(k-1)}(i,k) + T^{(k-1)}(k,j)\}$$

falls k nicht vorkommt in opt. Lösung für $T^{(k)}(i,j)$ gilt $T^{(k)}(i,j) = T^{(k-1)}(i,j)$

ansonsten:



hier gilt: $T^{(k)}(i,j) = T^{(k-1)}(i,k) + T^{(k-1)}(k,j)$

gegeben: ger. Graph $G = (V, E)$, Gewichte $w\colon E \to \mathbb{R}$

gesucht: Tabelle $D(i,j) =$ min. Gewicht eines i-j-Wegs

Idee: mehr Teilprobleme; weniger Zeit pro Teilproblem

Teilprobleme: $T^{(k)}(i,j) =$ min. Gewicht eines i-j-Pfads mit allen Zwischenknoten in $\{1,\dots,k\}$

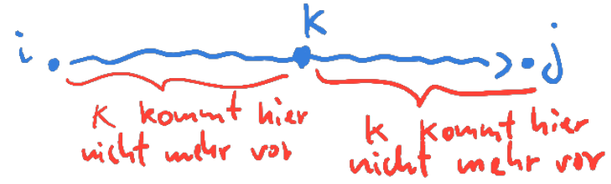
kein Knoten kommt mehrmals vor

Rekurrenz:

$$T^{(k)}(i,j) = \min\{T^{(k-1)}(i,j), T^{(k-1)}(i,k) + T^{(k-1)}(k,j)\}$$

falls k nicht vorkommt in opt. Lösung für $T^{(k)}(i,j)$ gilt $T^{(k)}(i,j) = T^{(k-1)}(i,j)$

ansonsten:



hier gilt: $T^{(k)}(i,j) = T^{(k-1)}(i,k) + T^{(k-1)}(k,j)$

Laufzeit:

gegeben: ger. Graph $G = (V, E)$, Gewichte $w \colon E \to \mathbb{R}$

gesucht: Tabelle $D(i, j) =$ min. Gewicht eines i-j-Wegs

Idee: mehr Teilprobleme; weniger Zeit pro Teilproblem

Teilprobleme: $T^{(k)}(i, j) =$ min. Gewicht eines i-j-Pfads mit allen Zwischenknoten in $\{1, \dots, k\}$

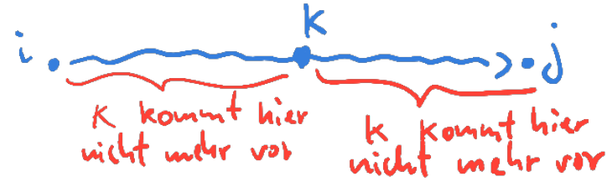
kein Knoten kommt mehrmals vor

Rekurrenz:

$$T^{(k)}(i, j) = \min\{T^{(k-1)}(i, j), T^{(k-1)}(i, k) + T^{(k-1)}(k, j)\}$$

falls k nicht vorkommt in opt. Lösung für $T^{(k)}(i,j)$ gilt $T^{(k)}(i,j) = T^{(k-1)}(i,j)$

ansonsten:



hier gilt: $T^{(k)}(i,j) = T^{(k-1)}(i,k) + T^{(k-1)}(k,j)$

Laufzeit: $O(n^3)$ Einträge; $O(1)$ Zeit pro Eintrag

⇝ Gesamtlaufzeit $O(n^3)$ des Floyd–Warshall Algorithmus

Ende