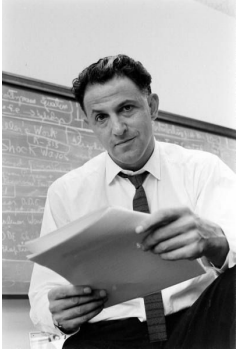
Agenda

3. Graphentheorie

Richard Bellman

Leben:

- *29.08.1920 (New York) †19.03.1984 (Los Angeles)
- Studium der Mathematik am Brooklyn College und der University of Wisconsin
- 1946: Promotion an der Princeton University
- forschte an der Wasserstoffbombe und beteiligte sich am „Manhatten Project“ (Bau der Atombombe)



Hauptwerk:

- „Dynamic Programming“ (1957)

Wirkung:

- Erfinder der Dynamischen Optimierung
- Bellman-Ford-Algorithmus
- Auszeichnungen:
  - Norbert-Wiener-Preis (1970)
  - Dickson Prize in Science (1970)
  - John-von-Neumann-Theorie-Preis (1976)

Lester Randolph Ford junior

Leben:

- *23.09.1927 (Houston) †27.02.2017
- Sohn von Lester Randolph Ford senior (Mathematiker)
- Arbeitete für CEIR Inc. und Rand Corporation

Hauptwerk:

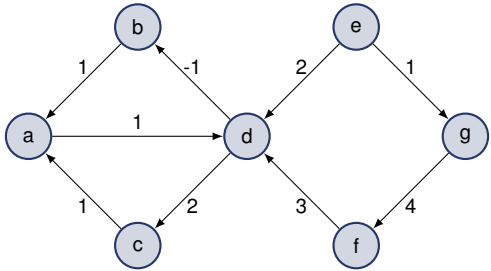
- „Flows in Networks“ mit D.R. Fulkerson (1962)

Wirkung:

- Algorithmus von Ford und Fulkerson
- Bellman-Ford-Algorithmus

Bellman-Ford-Algorithmus

Der Bellman-Ford-Algorithmus hat gegenüber dem Dijkstra-Algorithmus den Vorteil, dass er auch bei Graphen angewendet werden kann, welche negative Kantengewichte haben. Der Algorithmus terminiert jedoch nicht, wenn ein negativer Zyklus existiert.



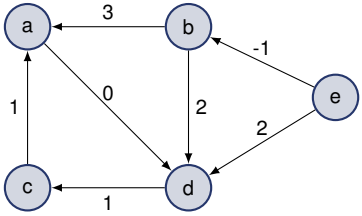
Bewertungsmatrix

Die Bewertungsmatrix $B(g) = b_{ij}$ des Graphen $g(V,E,w)$ gibt an, welches Gewicht eine Kante zwischen den Knoten i und j hat.

- Es handelt sich um eine [n x n]-Matrix.

Die Einträge b_{ij} der Matrix ergeben sich wie folgt:

- Existiert die Kante $e(i,j)$, $b_{ij} = w(i,j)$
- Falls $i = j$, $b_{ij} = 0$
- Sonst $b_{ij} = \infty$



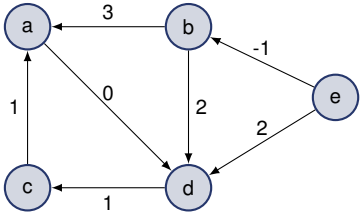
Bewertungsmatrix

Die Bewertungsmatrix $B(g) = b_{ij}$ des Graphen $g(V,E,w)$ gibt an, welches Gewicht eine Kante zwischen den Knoten i und j hat.

- Es handelt sich um eine [n x n]-Matrix.

Die Einträge b_{ij} der Matrix ergeben sich wie folgt:

- Existiert die Kante $e(i,j)$, $b_{ij} = w(i,j)$
- Falls $i = j$, $b_{ij} = 0$
- Sonst $b_{ij} = \infty$



$$B(g) = \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix}$$

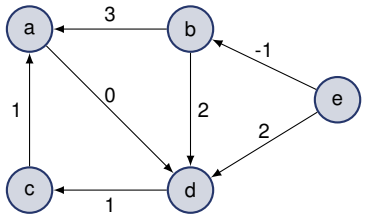
Tree-Matrix

Die Tree-Matrix $T(g) = t_{ij}$ des Graphen $g(V,E,w)$ speichert den direkten Vorgängerkonten auf dem kürzesten Weg von Knoten i zu Knoten j.

- Es handelt sich um eine [n x n]-Matrix.

Die Einträge der Matrix bei der Initialisierung ergeben sich wie folgt:

- Falls $(i,j) \in E$, $t_{ij} = i$
- Sonst -1



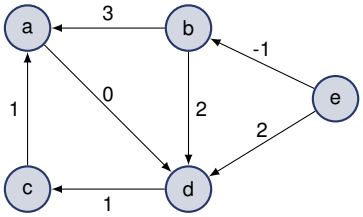
Tree-Matrix

Die Tree-Matrix $T(g) = t_{ij}$ des Graphen $g(V,E,w)$ speichert den direkten Vorgängerkonten auf dem kürzesten Weg von Knoten i zu Knoten j.

- Es handelt sich um eine [n x n]-Matrix.

Die Einträge der Matrix bei der Initialisierung ergeben sich wie folgt:

- Falls $(i,j) \in E$, $t_{ij} = i$
- Sonst -1



$$T(g) = \begin{pmatrix} -1 & -1 & -1 & a & -1 \\ b & -1 & -1 & b & -1 \\ c & -1 & -1 & -1 & -1 \\ -1 & -1 & d & -1 & -1 \\ -1 & e & -1 & e & -1 \end{pmatrix}$$

Bellman-Ford-Algorithmus

Voraussetzungen:

- Gewichteter Digraph (auch negative Kantengewichte sind zugelassen)

Variablen:

- Anzahl der betrachteten Kanten m
- Matrix der kürzesten Wege mit maximal m Kanten $U^{(m)}(g)$ mit den Einträgen u_{ij}
- Tree-Matrix nach Betrachtung der Wege mit maximal m Kanten $T^{(m)}(g)$

Initialisierung:

- $m = 0$
- $U^{(0)}$ mit $u_{ij} = \infty$
- $T^{(0)}$ mit $t_{ij} = -1$

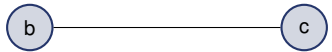
Ziel:

- Berechnung aller kürzesten Weglängen u_{ij}, sofern diese existieren, sowie die zugehörigen kürzesten Wege

Prinzip der optimalen Substruktur

Ein kürzester Weg zwischen zwei beliebigen Knoten i und j in einem Graphen $g(V,E,w)$ setzt sich immer aus kürzesten Teilwegen zusammen.

kürzester Weg von a nach d



kürzester Weg von b nach c

Dies gilt nur bei Graphen ohne negativen Zyklus!
Das Prinzip der optimalen Substruktur wird im Bellman-Ford-Algorithmus ausgenutzt, um u_{ij} zu bestimmen.

Die Bellman-Gleichung

Mit Hilfe der Bellman-Gleichung können die kürzesten Weglängen $u_{ij}^{(m)}$ mit maximal m Kanten zwischen den zwei Knoten i und j iterativ berechnet werden, sofern der kürzeste Weg existiert.

Der Wert $u_{ij}^{(m+1)}$ ergibt sich wie folgt:

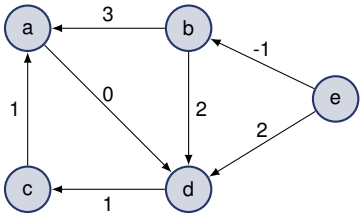
- $u_{ij}^{(1)} = b_{ij}$
- $u_{ij}^{(m+1)} = \min_k \left[u_{ik}^{(m)} + b_{kj} \right] \forall m \geq 1$

Interpretation:

- Die Länge des kürzesten Wegs von i nach j mit höchstens $m + 1$ Kanten ergibt sich aus der kürzesten Weglänge zu allen Vorgängerknoten k von j mit höchstens m Kanten plus die Länge der Kante von k nach j.
- $\rightarrow$ Ausnutzung des Prinzips der optimalen Substruktur

Matrixschreibweise der Bellman-Gleichung

Gegeben sei folgender Graph, dann ergibt sich für $U^{(1)}$, die Matrix aller kürzesten Weglängen:



$$U^{(1)} = B = \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix}$$

Um nun z. B. die kürzeste Weglängen von *e* nach *d* zu berechnen, geht man nach der Bellman-Gleichung wie folgt vor:

$$\begin{aligned} U_{ed}^{(2)} &= \min_k \left[u_{ek}^{(1)} + b_{kd} \right] \\ &= \min \left[u_{ea}^{(1)} + b_{ad}; u_{eb}^{(1)} + b_{bd}; u_{ec}^{(1)} + b_{cd}; u_{ed}^{(1)} + b_{dd}; u_{ee}^{(1)} + b_{ed} \right] \end{aligned}$$

Matrixschreibweise der Bellman-Gleichung

Die Berechnung von $u_{ed}^{(2)}$ entspricht einer Verknüpfung von Zeile „e“ aus $U^{(1)}$ mit Spalte „d“ aus B.

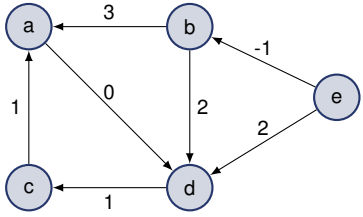
$$U^{(1)} \otimes B = \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix}$$

Somit ergibt sich $U^{(2)}$ als Verknüpfung der Matrizen $U^{(1)}$ und B.

- Die Vorgehensweise der Verknüpfung erfolgt durch die Bellman-Multiplikation.
- $X \otimes Y = C$ mit $c_{ij} = \min_k [x_{ik} + y_{kj}]$

Um nicht nur die kürzesten Weglängen zu berechnen, sondern auch den kürzesten Weg, muss t_{ij} immer dann mit den verwendeten Vorgängerknoten k aktualisiert werde, wenn sich u_{ij} verbessert.

Der Bellman-Ford-Algorithmus – Beispiel



1. Iteration: $U^{(1)} = B$
(wie im obigen Beispiel)

$$U^{(1)} = \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix}$$

$$T^{(1)} = \begin{pmatrix} -1 & -1 & -1 & a & -1 \\ b & -1 & -1 & b & -1 \\ c & -1 & -1 & -1 & -1 \\ -1 & -1 & d & -1 & -1 \\ -1 & e & -1 & e & -1 \end{pmatrix}$$

2. Iteration: $U^{(2)} = U^{(1)} \otimes B$

$$U^{(1)} \otimes B = \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} =$$

2. Iteration: $U^{(2)} = U^{(1)} \otimes B$

$$U^{(1)} \otimes B = \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} = \begin{pmatrix} u_{aa}^{(2)} & u_{ab}^{(2)} & u_{ac}^{(2)} & u_{ad}^{(2)} & u_{ae}^{(2)} \\ u_{ba}^{(2)} & u_{bb}^{(2)} & u_{bc}^{(2)} & u_{bd}^{(2)} & u_{be}^{(2)} \\ u_{ca}^{(2)} & u_{cb}^{(2)} & u_{cc}^{(2)} & u_{cd}^{(2)} & u_{ce}^{(2)} \\ u_{da}^{(2)} & u_{db}^{(2)} & u_{dc}^{(2)} & u_{dd}^{(2)} & u_{de}^{(2)} \\ u_{ea}^{(2)} & u_{eb}^{(2)} & u_{ec}^{(2)} & u_{ed}^{(2)} & u_{ee}^{(2)} \end{pmatrix}$$

Beispiel:

$$\begin{aligned} u_{bc}^{(2)} &= \min_k \left[u_{bk}^{(1)} + b_{kc} \right] \\ &= \min \left[u_{ba}^{(1)} + b_{ac}; u_{bb}^{(1)} + b_{bc}; u_{bc}^{(1)} + b_{cc}; u_{bd}^{(1)} + b_{dc}; u_{be}^{(1)} + b_{ec} \right] \\ &= \min \left[3 + \infty; 0 + \infty; \infty + 0; 2 + 1; \infty + \infty \right] \\ &= u_{bd}^{(1)} + b_{dc} = 3 \end{aligned}$$

2. Iteration: $U^{(2)} = U^{(1)} \otimes B$

$$U^{(1)} \otimes B = \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 3 & 1 & 0 \end{pmatrix}$$

Beispiel:

$$\begin{aligned} U_{bc}^{(2)} &= \min_k \left[u_{bk}^{(1)} + b_{kc} \right] \\ &= \min \left[u_{ba}^{(1)} + b_{ac}; u_{bb}^{(1)} + b_{bc}; u_{bc}^{(1)} + b_{cc}; u_{bd}^{(1)} + b_{dc}; u_{be}^{(1)} + b_{ec} \right] \\ &= \min \left[3 + \infty; 0 + \infty; \infty + 0; 2 + 1; \infty + \infty \right] \\ &= u_{bd}^{(1)} + b_{dc} = 3 \end{aligned}$$

2. Iteration: $U^{(2)} = U^{(1)} \otimes B$

$$U^{(1)} \otimes B = \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 3 & 1 & 0 \end{pmatrix}$$

$$T^{(2)} = \begin{pmatrix} -1 & -1 & d & a & -1 \\ b & -1 & d & b & -1 \\ c & -1 & -1 & a & -1 \\ c & -1 & d & -1 & -1 \\ b & e & d & b & -1 \end{pmatrix}$$

3. Iteration: $U^{(3)} = U^{(2)} \otimes B$

$$U^{(3)} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 3 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} =$$

3. Iteration: $U^{(3)} = U^{(2)} \otimes B$

$$U^{(3)} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 3 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & \boxed{2} & 1 & 0 \end{pmatrix}$$

3. Iteration: $U^{(3)} = U^{(2)} \otimes B$

$$U^{(3)} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 3 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & \boxed{2} & 1 & 0 \end{pmatrix}$$

$$T^{(3)} = \begin{pmatrix} -1 & -1 & d & a & -1 \\ b & -1 & d & b & -1 \\ c & -1 & -1 & a & -1 \\ c & -1 & d & -1 & -1 \\ b & e & \boxed{d} & b & -1 \end{pmatrix}$$

4. Iteration: $U^{(4)} = U^{(3)} \otimes B$
(letzte Iteration, da 5 Knoten)

$$U^{(4)} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 2 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} =$$

4. Iteration: $U^{(4)} = U^{(3)} \otimes B$
(letzte Iteration, da 5 Knoten)

$$U^{(4)} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 2 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 2 & 1 & 0 \end{pmatrix}$$

4. Iteration: $U^{(4)} = U^{(3)} \otimes B$
(letzte Iteration, da 5 Knoten)

$$U^{(4)} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 2 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & \infty & \infty & 0 & \infty \\ 3 & 0 & \infty & 2 & \infty \\ 1 & \infty & 0 & \infty & \infty \\ \infty & \infty & 1 & 0 & \infty \\ \infty & -1 & \infty & 2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 2 & 1 & 0 \end{pmatrix}$$

Es gibt keine Aktualisierung

- $T^{(4)} = T^{(3)}$
- Abbruch des Algorithmus
- Kürzeste Wege wurden gefunden (es gibt keine negativen Zykel)

Interpretation

- Die Länge des kürzesten Weges zwischen allen Punkten des Graphen kann aus der $U^{(|V|-1)}$-Matrix abgelesen werden, sofern dieser existiert.
- Für die Darstellung des Weges müssen die Vorgänger aus der Tree-Matrix abgelesen werden.

$$U^{(|V|-1)} = \begin{pmatrix} 0 & \infty & 1 & 0 & \infty \\ 3 & 0 & 3 & 2 & \infty \\ 1 & \infty & 0 & 1 & \infty \\ 2 & \infty & 1 & 0 & \infty \\ 2 & -1 & 2 & 1 & 0 \end{pmatrix}$$

$$T^{(|V|-1)} = \begin{pmatrix} -1 & -1 & d & a & -1 \\ b & -1 & d & b & -1 \\ c & -1 & -1 & a & -1 \\ c & -1 & d & -1 & -1 \\ b & e & d & b & -1 \end{pmatrix}$$

Abbruchkriterien

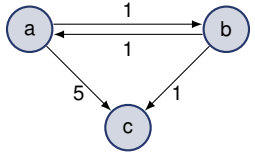
- Treten auf der Hauptdiagonalen negative Einträge auf, kann der Algorithmus abgebrochen werden, da dies ein Hinweis auf einen negativen Zyklus ist.
- Verändern sich die Einträge der U-Matrix von einer Iteration zur nächsten nicht, so kann der Algorithmus abgebrochen werden, da bereits eine optimale Lösung vorliegt.
- Der Algorithmus muss maximal $|V| - 1$ mal durchlaufen werden, danach noch ein weiteres Mal zum Test auf einen negativen Zyklus.

Agenda

3. Graphentheorie

K-Kürzeste Wege

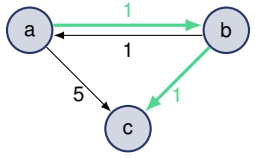
Der Dijkstra- und der Bellman-Ford-Algorithmus berechnen den kürzesten Weg.
In der Realität könnten aber auch der zweit- und/oder drittkürzeste Weg eine Rolle spielen:
$\Rightarrow$ k-kürzeste-Wege-Problem



Iteration	Weg	Kosten	Bemerkung

K-Kürzeste Wege

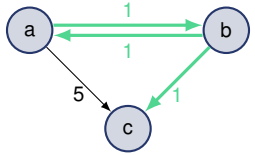
Der Dijkstra- und der Bellman-Ford-Algorithmus berechnen den kürzesten Weg.
In der Realität könnten aber auch der zweit- und/oder drittkürzeste Weg eine Rolle spielen:
$\Rightarrow$ k-kürzeste-Wege-Problem



Iteration	Weg	Kosten	Bemerkung
1	$a \rightarrow b \rightarrow c$	2	

K-Kürzeste Wege

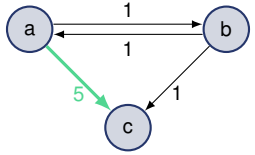
Der Dijkstra- und der Bellman-Ford-Algorithmus berechnen den kürzesten Weg.
In der Realität könnten aber auch der zweit- und/oder drittkürzeste Weg eine Rolle spielen:
$\Rightarrow$ k-kürzeste-Wege-Problem



Iteration	Weg	Kosten	Bemerkung
1	$a \to b \to c$	2	
2	$a \to b \to a \to b \to c$	4	Zyklus

K-Kürzeste Wege

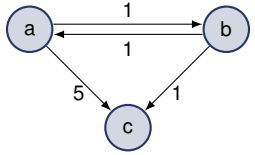
Der Dijkstra- und der Bellman-Ford-Algorithmus berechnen den kürzesten Weg.
In der Realität könnten aber auch der zweit- und/oder drittkürzeste Weg eine Rolle spielen:
$\Rightarrow$ k-kürzeste-Wege-Problem



Iteration	Weg	Kosten	Bemerkung
1	$a \to b \to c$	2	
2	$a \to b \to a \to b \to c$	4	Zyklus
3	$a \to c$	5	

K-Kürzeste Wege

Der Dijkstra- und der Bellman-Ford-Algorithmus berechnen den kürzesten Weg.
In der Realität könnten aber auch der zweit- und/oder drittkürzeste Weg eine Rolle spielen:
$\Rightarrow$ k-kürzeste-Wege-Problem



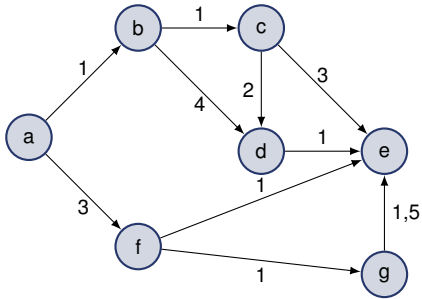
Iteration	Weg	Kosten	Bemerkung
1	$a \to b \to c$	2	
2	$a \to b \to a \to b \to c$	4	Zyklus
3	$a \to c$	5	

Mögliche Fragestellungen

- Zyklen/Schlingen zulässig $\to$ unendlich viele Alternativwege möglich
- Zyklen/Schlingen unzulässig $\to$ Yen-Algorithmus anwendbar

Algorithmus von Yen

Idee: Berechne minimale Wegalternative (Deviation) eines Weges bezüglich eines Abzweigpunktes und einer Ausschlussmenge an Wegen.



Yen算法

思路：计算相对于分支点和一组排除路径的路径的最小偏差（Deviation）。

Jin Y. Yen

Hauptwerk:

- 1971 “Finding the k Shortest Loopless Paths in a Network“ herausgegeben von der Monterrey California Naval Postgraduate School

Wirkung:

- Yen-Algorithmus zur Bestimmung k-kürzester Wege als Erweiterung des Bellman-Ford-Algorithmus

Algorithmus von Yen

Variablen:

- Liste der kürzesten Wege $P = \{p_1, p_2, \ldots, p_k\}$
- Kandidatenliste K für den nächstkürzesten Weg
- Iteration k (k-kürzester Weg)

Initialisierung:

- $k = 1$
- $K = p_1$ (kürzester Weg, ermittelt mit bekanntem Algorithmus)
- $P = \emptyset$

1. 设置 $k = 1$ 来表示当前正在查找的是最短路径。
2. 用某个已知的算法（如Dijkstra算法）计算最短路径 p_1。
3. 初始化路径列表 P 为一个空集，它将用来存储找到的最短路径。

Algorithmus von Yen

Iteration für $k > 1$:

- Wähle denjenigen Weg aus K mit minimaler Weglänge als k-kürzesten Weg p_k
- $P = P \cup \{p_k\}$
- $K = K \setminus \{p_k\}$
- Für jeden Knoten von p_k, mit Ausnahme des Zielknotens (Abzweigknoten):
  - ▷ Berechne minimale schleifenlose Wegalternative p' zu p_{k-1} für Abzweigknoten und Ausschlussmenge $\{p_1, p_2, \ldots, p_{k-1}\}$
  - ▷ Wenn p' existiert: $K = K \cup \{p'\}$

Abbruchkriterien:

- $k =$ Anzahl gewünschter kürzester Wege
- $K = \emptyset$

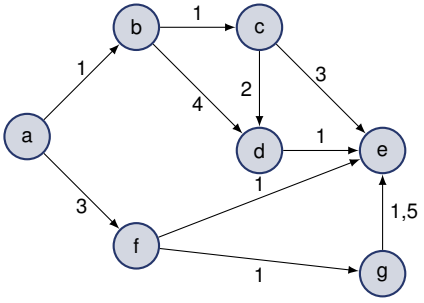
1. 从候选路径集合 K 中选择一个最短的路径 p_k，这个路径的长度比已找到的最短路径 p_{k-1} 要长。
2. 将新找到的路径 p_k 添加到路径列表 P 中。
3. 从候选路径集合 K 中移除路径 p_k。
4. 对路径 p_k 中的每个节点（除了终点），做以下操作：
   - 计算绕过已知最短路径 $\{p_1, p_2, \ldots, p_{k-1}\}$ 的替代路径 p'。
   - 如果这样的路径 p' 存在，将其添加到候选路径集合 K 中。

算法结束的条件：

1. 当已经找到了所需数量的最短路径，即 k 等于所希望找到的最短路径的数量。
2. 当没有更多的候选路径可用时，即候选路径集合 K 为空时。

Algorithmus von Yen – Beispiel

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative

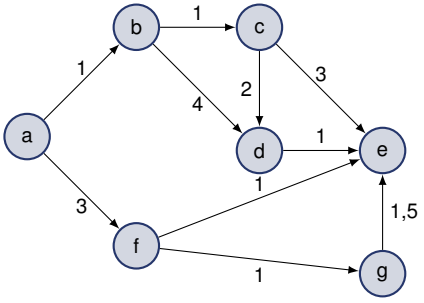


Kandidatenliste

Weg	Länge
afe	4

Algorithmus von Yen – Beispiel

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative
2	afe (4)			

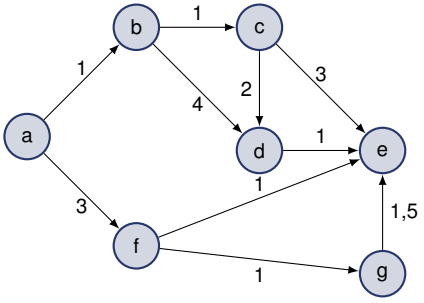


Kandidatenliste

Weg	Länge
~~afe~~	~~4~~

Algorithmus von Yen – Beispiel

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative
2	afe (4)	a		
		f		

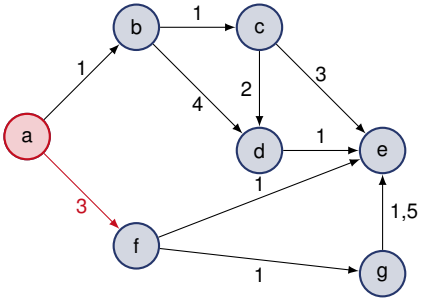


Kandidatenliste

Weg	Länge
~~afe~~	~~4~~

Algorithmus von Yen – Beispiel

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative
2	afe (4)	a f	af	abcde

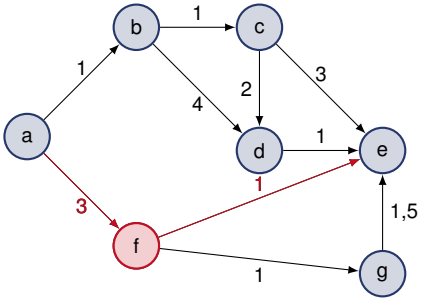


Kandidatenliste

Weg	Länge
~~afe~~	~~4~~

分支点和排除路径

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative
2	afe (4)	a	af	abcde
		f	afe	afge

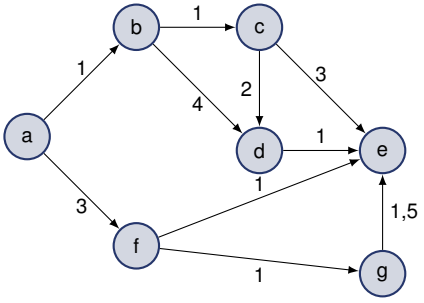


Kandidatenliste

Weg	Länge
~~afe~~	~~4~~

Algorithmus von Yen – Beispiel

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative
2	afe (4)	a	af	abcde
		f	afe	afge

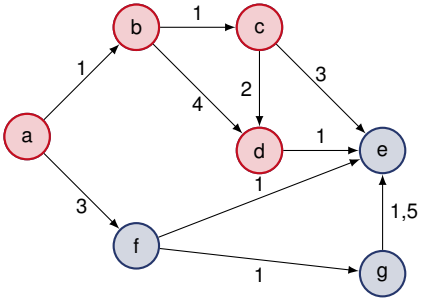


Kandidatenliste

Weg	Länge
~~afe~~	~~4~~
abcde	5
afge	5,5

Algorithmus von Yen – Beispiel

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative
2	afe (4)	a	af	abcde
		f	afe	afge
3	abcde (5)	a	af,ab	
		b	abc	abde
		c	abcd	abce
		d	abcde	

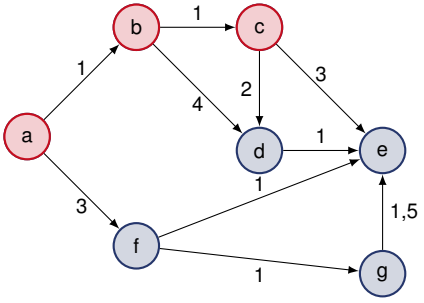


Kandidatenliste

Weg	Länge
~~afe~~	~~4~~
~~abcde~~	~~5~~
afge	5,5
abde	6
abce	5

Algorithmus von Yen – Beispiel

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative
2	afe (4)	a	af	abcde
		f	afe	afge
3	abcde (5)	a	af,ab	
		b	abc	abde
		c	abcd	abce
		d	abcde	
4	abce (5)	a	af,ab	
		b	abc	(abde)
		c	abcd,abce	

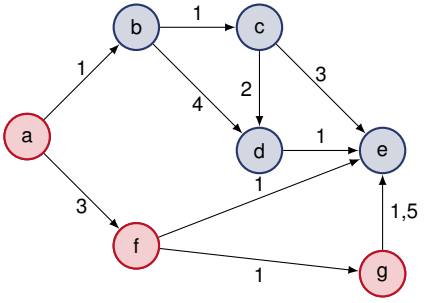


Kandidatenliste

Weg	Länge
~~afe~~	~~4~~
~~abcde~~	~~5~~
afge	5,5
abde	6
~~abce~~	~~5~~

Algorithmus von Yen – Beispiel

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative
2	afe (4)	a	af	abcde
		f	afe	afge
3	abcde (5)	a	af,ab	
		b	abc	abde
		c	abcd	abce
		d	abcde	
4	abce (5)	a	af,ab	
		b	abc	(abde)
		c	abcd,abce	
5	afge (5,5)	a/f/g	...	–

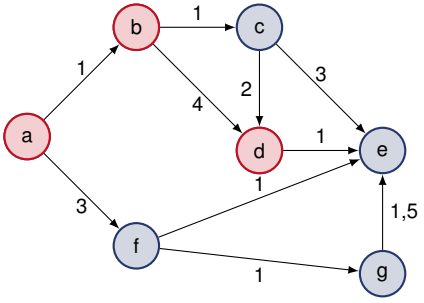


Kandidatenliste

Weg	Länge
~~afe~~	~~4~~
~~abcde~~	~~5~~
~~afge~~	~~5,5~~
abde	6
~~abce~~	~~5~~

Algorithmus von Yen – Beispiel

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative
2	afe (4)	a	af	abcde
		f	afe	afge
3	abcde (5)	a	af,ab	
		b	abc	abde
		c	abcd	abce
		d	abcde	
4	abce (5)	a	af,ab	
		b	abc	(abde)
		c	abcd,abce	
5	afge (5,5)	a/f/g	...	–
6	abde (6)	a/b/d	...	–

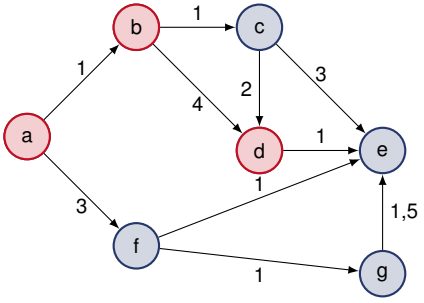


Kandidatenliste

Weg	Länge
~~afe~~	~~4~~
~~abcde~~	~~5~~
~~afge~~	~~5,5~~
~~abde~~	~~6~~
~~abce~~	~~5~~

Algorithmus von Yen – Beispiel

k	p_{k-1}	Abzweig	Ausschluss	Minimale Wegalternative
2	afe (4)	a	af	abcde
		f	afe	afge
3	abcde (5)	a	af,ab	
		b	abc	abde
		c	abcd	abce
		d	abcde	
4	abce (5)	a	af,ab	
		b	abc	(abde)
		c	abcd,abce	
5	afge (5,5)	a/f/g	...	–
6	abde (6)	a/b/d	...	–



Kandidatenliste

Weg	Länge
~~afe~~	~~4~~
~~abcde~~	~~5~~
~~afge~~	~~5,5~~
~~abde~~	~~6~~
~~abce~~	~~5~~