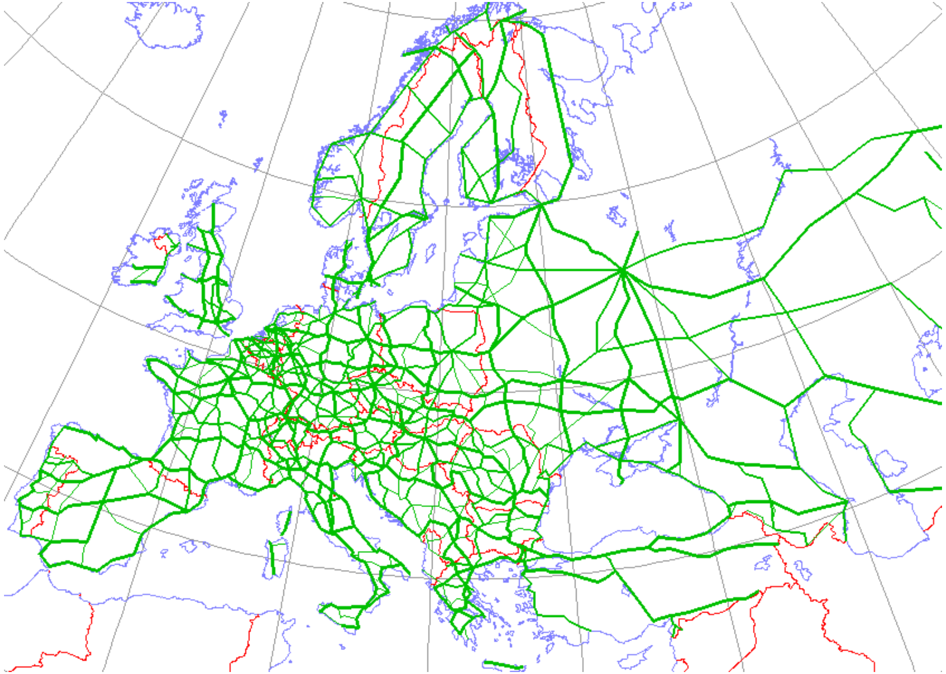
Technische Universität Braunschweig

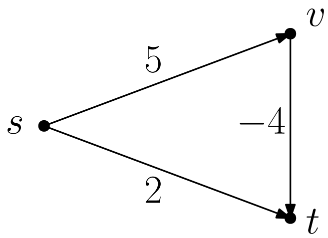
Netzwerkalgorithmen – Vorlesung #6

Arne Schmidt

Wiederholung

Wiederholung





Algorithmus 3.9
Eingabe:
Digraph $D = (V, A)$, konservative Kostenfunktion $c: A \to \mathbb{R}$, Startknoten $s \in V$
Ausgabe:
Länge $\ell(v)$ eines kürzesten sv-Pfades, Vorgänger $p(v)$ von v auf diesem Pfad

1. Function MBF(D, c, s)
2. $\quad \ell(s) \coloneqq 0$
3. $\quad \ell(v) \coloneqq \infty,\ p(v) \coloneqq nil \quad \forall v \in V \setminus \{s\}$
4. $\quad$ For $i = 1$ to $n - 1$ do
5. $\quad\quad$ For all $(v, w) \in A$ do
6. $\quad\quad\quad$ If $\ell(w) > \ell(v) + c\big((v, w)\big)$ then
7. $\quad\quad\quad\quad \ell(w) \coloneqq \ell(v) + c\big((v, w)\big)$
8. $\quad\quad\quad\quad p(w) \coloneqq v$

Satz 3.10
Der Algorithmus von Moore, Bellman und Ford löst Problem 3.4 (Kürzeste Wege) korrekt für konservative Kantenkosten in $O(nm)$ Zeit.

3.3 All Pairs Shortest Paths

All Pairs Shortest Paths

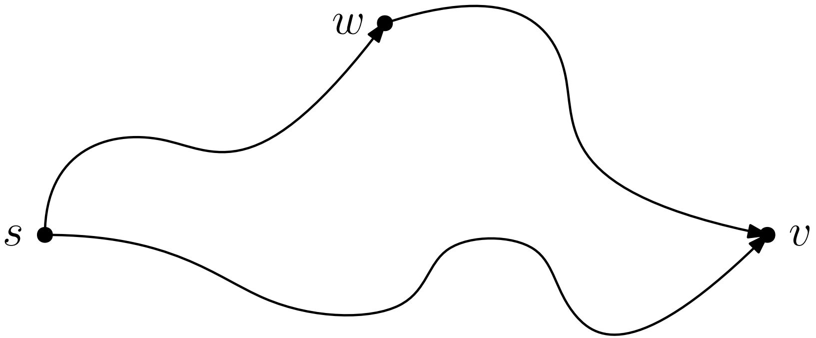
Problem 3.13: Paarweise kürzeste Wege Problem (All Pairs Shortest Path, APSP)
Gegeben:
Digraph $D = (V, A)$
Konservative Kostenfunktion $c\colon A \to \mathbb{R}$
Gesucht:
Für jedes Knotenpaar $\left(v_i, v_j\right) \in V^2$ einen kürzesten $v_i v_j$-Weg

Idee 1:
n Mal Dijkstra mit Laufzeit $O(mn + n^2 \log n)$
Problem: Nur positive Kantengewichte möglich!

Idee 2:
n Mal MBF mit Laufzeit $O(mn^2)$
Problem: Im Worst-Case ist die Laufzeit $O(n^4)$!

Idee

Betrachte nicht nur direkten Vorgänger, sondern einen Knoten auf einem sv-Pfad:



Annahme: Wir kennen zu jedem Knotenpaar den kürzesten Pfad, wenn Knoten $v_1, \dots, v_{k-1}$ auf diesem Pfad liegen dürfen.

Frage: Gelange ich schneller von v_i nach v_j, wenn erst Knoten v_k besucht wird?

Rekursionsgleichung

Sei $\ell(i,j,k)$ der kürzeste $v_i v_j$-Pfad, wenn Knoten $v_1, \dots, v_k$ benutzt werden dürfen.

Initialisierung: Weg über anliegende Kante (falls ex.).
∞ falls Kante nicht exisitert.

$$\ell(i,j,k) := \begin{cases} c\left(\left(v_i, v_j\right)\right), & \text{falls } k = 0 \\ \min\left(\ell(i,j,k-1), \ell(i,k,k-1) + \ell(k,j,k-1)\right), & \text{falls } k > 0 \end{cases}$$

Der neue Knoten bringt keine Verbesserung

Weg über neuen Knoten ist besser.

Algorithmus von Floyd-Warshall

Algorithmus 3.14

Eingabe:

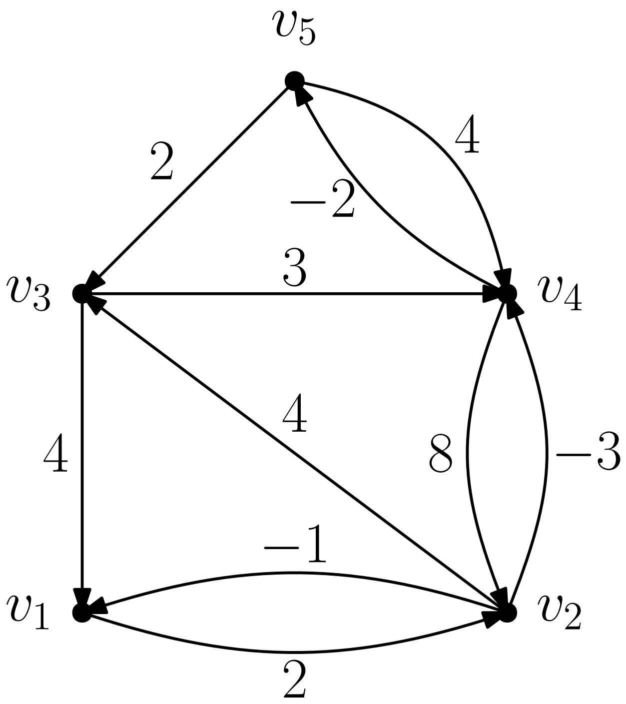
Digraph $D = (V, A)$, Kostenfunktion $c\colon A \to \mathbb{R}$

Ausgabe:

Für jedes Paar $(v_i, v_j) \in V^2$ die Länge $\ell_i(j)$ des kürzesten $v_i v_j$-Pfades und den Vorgänger $p_i(j)$ von v_j auf diesem Pfad.

Function $\textsc{FloydWarshall}(D, c)$
 $\ell_i(j) = \infty,\ p_i(j) = \text{nil},\ \forall (v_i, v_j) \in V^2$
 $\ell_i(j) = c((v_i, vj\,)),\ p_i(j) = v_i,\ \forall (v_i, v_j) \in A$
 $\ell_i(i) = 0,\ \forall v_i \in V$
 For $k = 1$ to n do
  For $i = 1$ to n do
   For $j = 1$ to n do
    If $\ell_i(j) > \ell_i(k) + \ell_k(j)$ then
     $\ell_i(j) \coloneqq \ell_i(k) + \ell_k(j)$
     $p_i(j) \coloneqq p_k(j)$

Beispiel



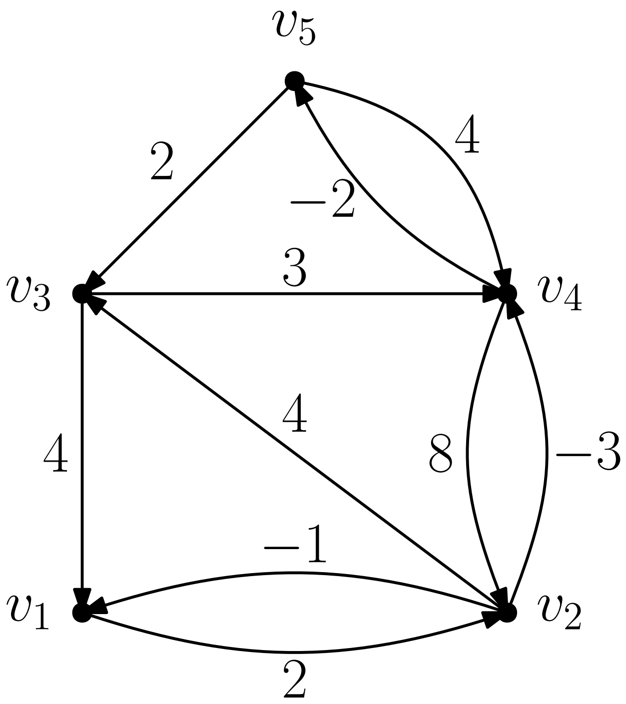
$k = 0$

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 2 & \infty & \infty & \infty \\ -1 & 0 & 4 & -3 & \infty \\ 4 & \infty & 0 & 3 & \infty \\ \infty & 8 & \infty & 0 & -2 \\ \infty & \infty & 2 & 4 & 0 \end{pmatrix} \end{array}$$

$k = 1$

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 2 & \infty & \infty & \infty \\ -1 & 0 & 4 & -3 & \infty \\ 4 & 6 & 0 & 3 & \infty \\ \infty & 8 & \infty & 0 & -2 \\ \infty & \infty & 2 & 4 & 0 \end{pmatrix} \end{array}$$

Beispiel



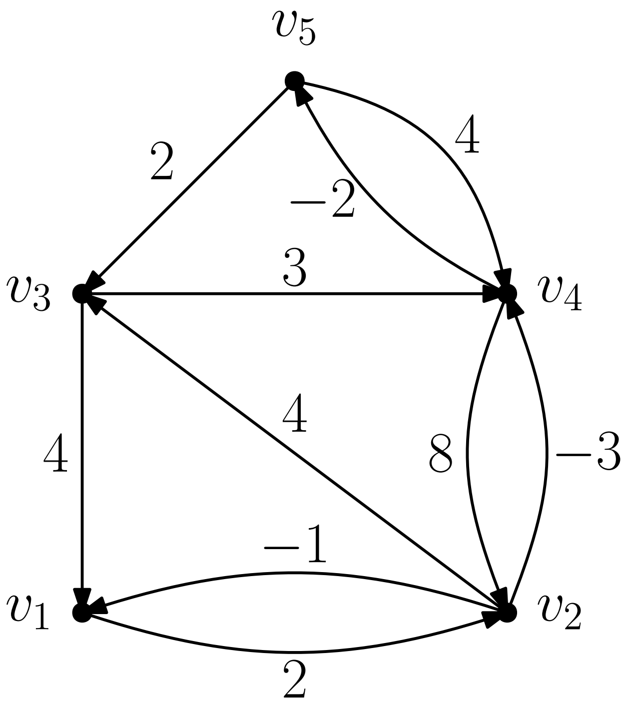
$k = 1$

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 2 & \infty & \infty & \infty \\ -1 & 0 & 4 & -3 & \infty \\ 4 & 6 & 0 & 3 & \infty \\ \infty & 8 & \infty & 0 & -2 \\ \infty & \infty & 2 & 4 & 0 \end{pmatrix} \end{array}$$

$k = 2$

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 2 & 6 & -1 & \infty \\ -1 & 0 & 4 & -3 & \infty \\ 4 & 6 & 0 & 3 & \infty \\ 7 & 8 & 12 & 0 & -2 \\ \infty & \infty & 2 & 4 & 0 \end{pmatrix} \end{array}$$

Beispiel



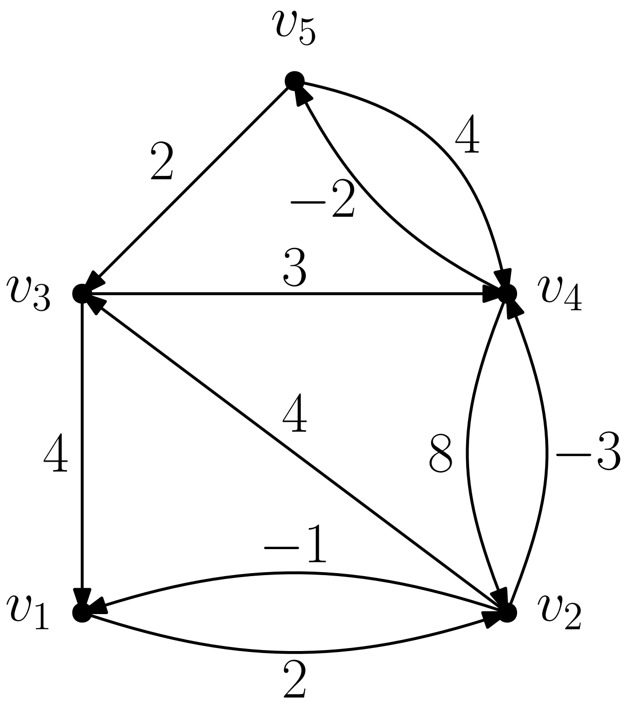
$k = 2$

$$
\begin{array}{c|ccccc}
 & v_1 & v_2 & v_3 & v_4 & v_5 \\
\hline
v_1 & 0 & 2 & 6 & -1 & \infty \\
v_2 & -1 & 0 & 4 & -3 & \infty \\
v_3 & 4 & 6 & 0 & 3 & \infty \\
v_4 & 7 & 8 & 12 & 0 & -2 \\
v_5 & \infty & \infty & 2 & 4 & 0
\end{array}
$$

$k = 3$

$$
\begin{array}{c|ccccc}
 & v_1 & v_2 & v_3 & v_4 & v_5 \\
\hline
v_1 & 0 & 2 & 6 & -1 & \infty \\
v_2 & -1 & 0 & 4 & -3 & \infty \\
v_3 & 4 & 6 & 0 & 3 & \infty \\
v_4 & 7 & 8 & 12 & 0 & -2 \\
v_5 & 6 & 8 & 2 & 4 & 0
\end{array}
$$

Beispiel



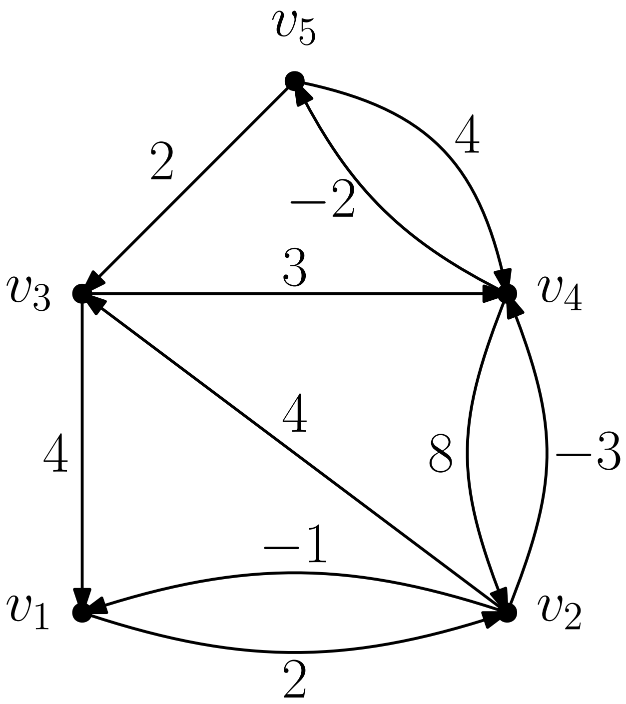
$k = 3$

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 2 & 6 & -1 & \infty \\ -1 & 0 & 4 & -3 & \infty \\ 4 & 6 & 0 & 3 & \infty \\ 7 & 8 & 12 & 0 & -2 \\ 6 & 8 & 2 & 4 & 0 \end{pmatrix} \end{array}$$

$k = 4$

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 2 & 6 & -1 & -3 \\ -1 & 0 & 4 & -3 & -5 \\ 4 & 6 & 0 & 3 & 1 \\ 7 & 8 & 12 & 0 & -2 \\ 6 & 8 & 2 & 4 & 0 \end{pmatrix} \end{array}$$

Beispiel



$k = 4$

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 2 & 6 & -1 & -3 \\ -1 & 0 & 4 & -3 & -5 \\ 4 & 6 & 0 & 3 & 1 \\ 7 & 8 & 12 & 0 & -2 \\ 6 & 8 & 2 & 4 & 0 \end{pmatrix} \end{array}$$

$k = 5$

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 2 & -1 & -1 & -3 \\ -1 & 0 & -3 & -3 & -5 \\ 4 & 6 & 0 & 3 & 1 \\ 4 & 6 & 0 & 0 & -2 \\ 6 & 8 & 2 & 4 & 0 \end{pmatrix} \end{array}$$

Algorithmus von Floyd-Warshall

Satz 3.15
Der Algorithmus von Floyd-Warshall löst Problem 3.13 (APSP) korrekt in Zeit $O(n^3)$.

Beweis: Selbst!

Lemma 3.16
Ein Digraph $D = (V, A)$ mit Kostenfunktion $c: A \to \mathbb{R}$ enthält genau dann einen negativen Kreis, wenn für (mind.) einen Knoten $v_i \in V$ nach Ausführung des Algorithmus von Floyd-Warshall $\ell_i(i) < 0$ gilt.

Beweis: Selbst!

Geht das schneller?

Können wir die Gewichte verändern?

Sind alle positiv, können wir Dijkstra nutzen…

Geht das schneller?

Reweighting – Idee

Suche Knotenlabel $h(v)$, sodass für jede Kante $e = (u, v)$ folgendes gilt

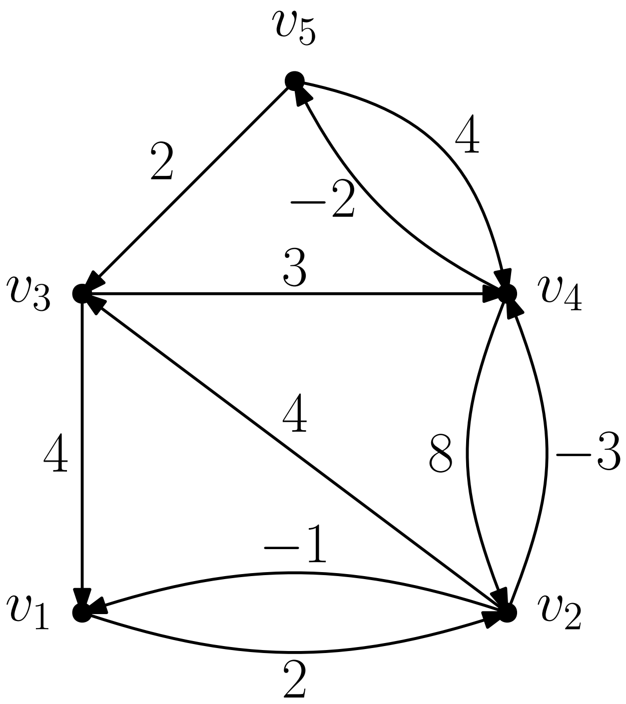
$$c'(e) \coloneqq c(e) + h(u) - h(v) \geq 0$$

Dann gilt für jeden $v_1 v_k$-Pfad $P \coloneqq (v_1, \dots, v_k)$

$$c'(P) = \sum_{i=1}^{k-1} c'\big((v_i, v_{i+1})\big) = \sum_{i=1}^{k-1} c\big((v_i, v_{i+1})\big) + \underbrace{h(v_i) - h(v_{i+1})}_{\text{Teleskopsumme}} = c(P) + h(v_1) - h(v_k)$$

Also ist $c(P)$ minimal genau dann, wenn $c'(P)$ minimal ist.

Gut erreichbare Knoten



Einerseits:

v_5 lässt sich leicht über v_4 erreichen.
v_4 lässt sich leicht über v_2 erreichen.

Andererseits:

v_3 lässt sich leicht über v_5 erreichen, wenn vorher v_2 und v_4 besucht wurde.

Algorithmus von Johnson(-Dijkstra)

Algorithmus 3.17

Eingabe:

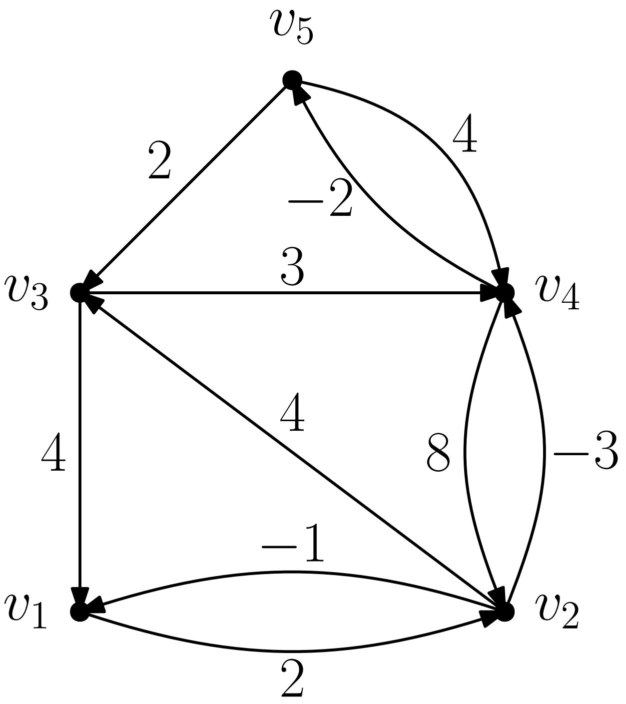
Digraph $D = (V, A)$, Kostenfunktion $c: A \to \mathbb{R}$

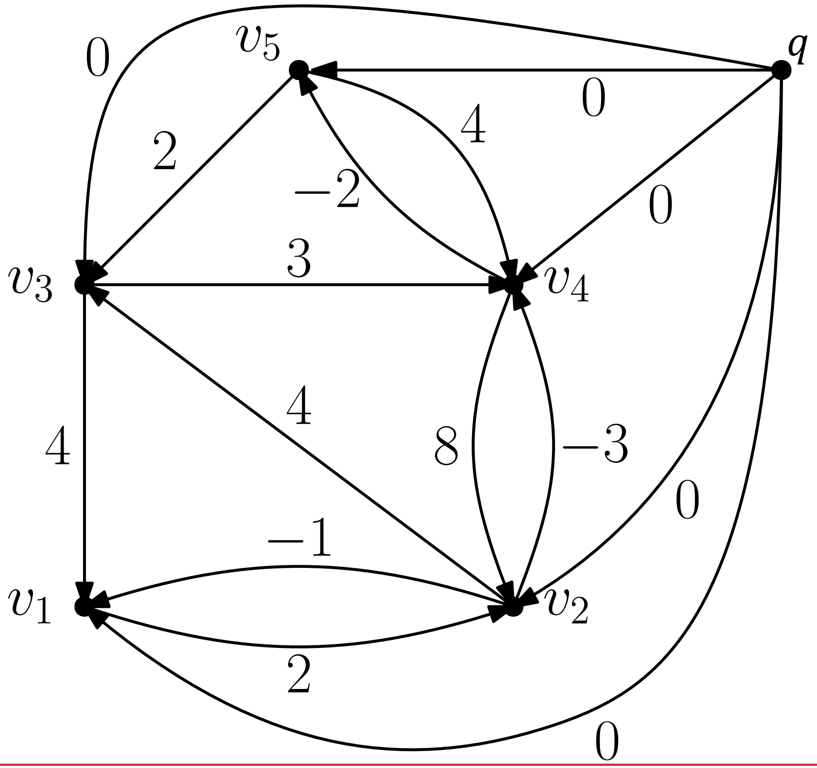
Ausgabe:

Für jedes Paar $\left(v_i, v_j\right) \in V^2$ die Länge $\ell_i(j)$ des kürzesten $\mathrm{v}_i v_j$-Pfades und den Vorgänger $p_i(j)$ von v_j auf diesem Pfad.

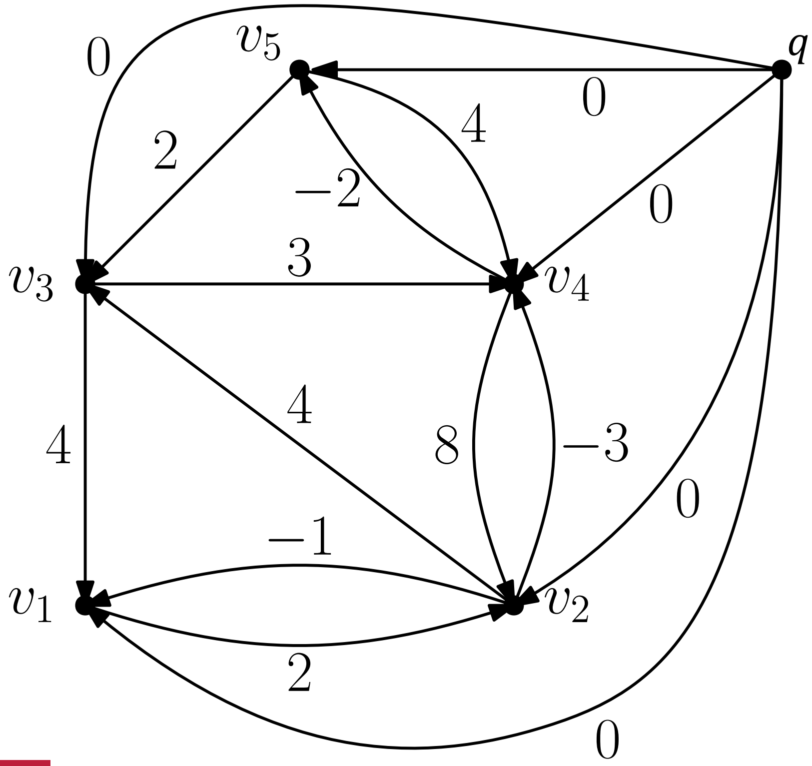
1. Function $\textsc{Johnson}(D, c)$
2. Füge Knoten q hinzu
3. Füge Kanten (q, v) mit $c\big((q, v)\big) := 0$ für alle $v \in V$ hinzu.
4. Führe $\mathrm{MBF}(D, c, q)$ aus.
   Das ergibt Label $\ell(v)$ für alle Knoten $v \in V$
5. Entferne q
6. Setze $c'\big((u, v)\big) := c\big((u, v)\big) + \ell(u) - \ell(v)$
7. Führe einmal Dijkstra von jedem Knoten v_i aus,
   um Label $\ell_i(j)$ zu erhalten.
8. Subtrahiere von jedem Label $\ell_i(j)$ den Wert $\ell(v_i) - \ell\big(v_j\big)$.

Beispiel





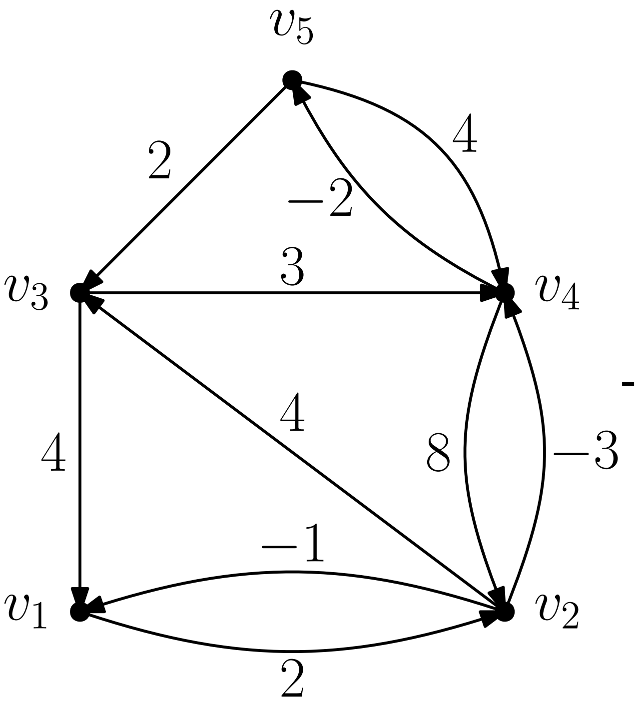
Beispiel



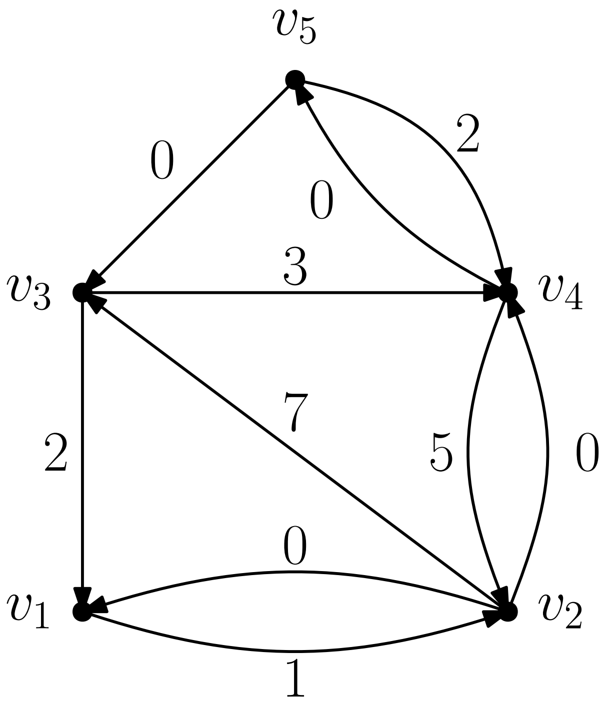
Label nach MBF:

$\ell(v_1) = -1$
$\ell(v_2) = 0$
$\ell(v_3) = -3$
$\ell(v_4) = -3$
$\ell(v_5) = -5$

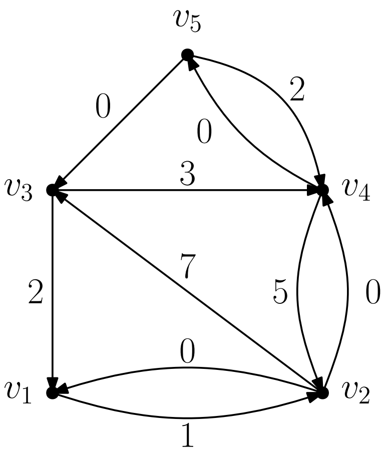
Beispiel



Label nach MBF:
$\ell(v_1) = -1$
$\ell(v_2) = 0$
$\ell(v_3) = -3$
$\ell(v_4) = -3$
$\ell(v_5) = -5$

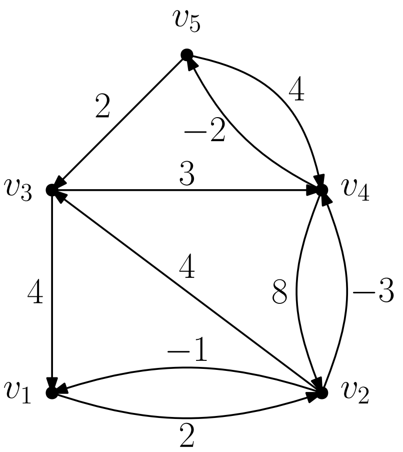


Beispiel



$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}
\begin{array}{c}
\begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\
\begin{pmatrix}
0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 \\
2 & 3 & 0 & 3 & 3 \\
2 & 3 & 0 & 0 & 0 \\
2 & 3 & 0 & 2 & 0
\end{pmatrix}
\end{array}
$$

Label nach MBF:
$\ell(v_1) = -1$
$\ell(v_2) = 0$
$\ell(v_3) = -3$
$\ell(v_4) = -3$
$\ell(v_5) = -5$



$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}
\begin{array}{c}
\begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\
\begin{pmatrix}
0 & 2 & -1 & -1 & -3 \\
-1 & 0 & -3 & -3 & -5 \\
4 & 6 & 0 & 3 & 1 \\
4 & 6 & 0 & 0 & -2 \\
6 & 8 & 2 & 4 & 0
\end{pmatrix}
\end{array}
$$

Algorithmus von Johnson

Satz 3.18
Der Algorithmus von Johnson löst Problem 3.13 (APSP) korrekt in Zeit $O(mn + n^2 \log n)$.

Beweisidee für **Korrektheit**:
Zeige, dass $c'(e) \geq 0$ für alle $e \in A$ gilt.

Laufzeit:
Im Wesentlichen wird einmal MBF und n-Mal Dijsktra durchgeführt.
Also ist die Laufzeit

$$O(mn + n \cdot (m + n \log n)) = O(mn + n^2 \log n)$$

3.4 Überblick

Algorithmen

Algorithmus	Kostenfunktionen	Problem	Laufzeit	Paradigma
Dijkstra	$\mathbb{R}_{\geq 0}$	SSSP	$O(m + n \log n)$	Greedy, Dyn. Programming
Moore, Bellman, Ford	Konservativ	SSSP	$O(mn)$	Dyn. Programming
Floyd-Warshall	Konservativ	APSP	$O(n^3)$	Dyn. Programming
Johnson	Konservativ	APSP	$O(mn + n^2 \log n)$	Reweighting, Dyn. Programming

Graphenklassen – SSSP

Graphenklasse	Kostenfunktionen	Laufzeit	Algorithmus
Dicht $m \in \Omega(n \log n)$	$\mathbb{R}_{\geq 0}$	$O(m)$	Dijkstra
Dünn $m \in O(n \log n)$	$\mathbb{R}_{\geq 0}$	$O(n \log n)$	Dijkstra
Beliebig	$\mathbb{N}, c(e) \leq L$	$O(Lm + n)$	BFS
Dicht $m \in \Omega(n \log n)$	Konservativ	$O(mn)$	MBF
Dünn $m \in O(n \log n)$	Konservativ	$O(n^2 \log n)$	MBF

Graphenklassen – APSP

Graphenklasse	Kostenfunktionen	Laufzeit	Algorithmus
Dicht $m \in \Omega(n \log n)$	$\mathbb{R}_{\geq 0}$	$O(nm)$	n Mal Dijkstra
Dünn $m \in O(n \log n)$	$\mathbb{R}_{\geq 0}$	$O(n^2 \log n)$	n Mal Dijkstra
Sehr dicht $m \in \Omega(n^2)$	Konservativ	$O(n^3)$	Floyd-Warshall
Dicht $m \in \Omega(n \log n)$	Konservativ	$O(mn)$	Johnson
Dünn $m \in O(n \log n)$	Konservativ	$O(n^2 \log n)$	Johnson

Quiz!

Mentimeter