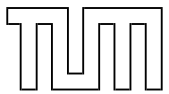
Fortgeschrittene Netzwerk- und Graph-Algorithmen

Prof. Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Wintersemester 2010/11



Übersicht

Kürzeste Wege für beliebige Graphen mit beliebigen Gewichten

Gegeben:

- beliebiger Graph mit beliebigen Kantengewichten

$\Rightarrow$ Anhängen einer Kante an einen Weg kann zur Verkürzung des Weges (Kantengewichtssumme) führen (wenn Kante negatives Gewicht hat)

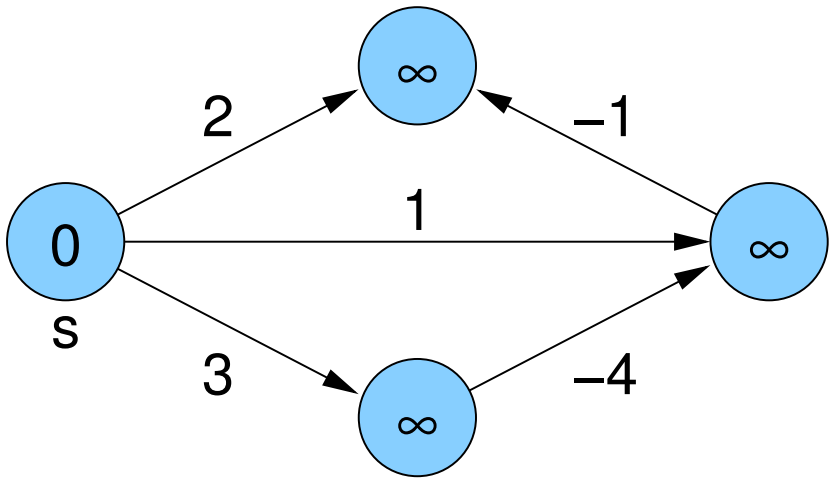
$\Rightarrow$ es kann negative Kreise und Knoten mit Distanz $-\infty$ geben

Problem:

- besuche Knoten eines kürzesten Weges in der richtigen Reihenfolge
- Dijkstra kann nicht mehr verwendet werden, weil Knoten nicht unbedingt in der Reihenfolge der kürzesten Distanz zum Startknoten s besucht werden

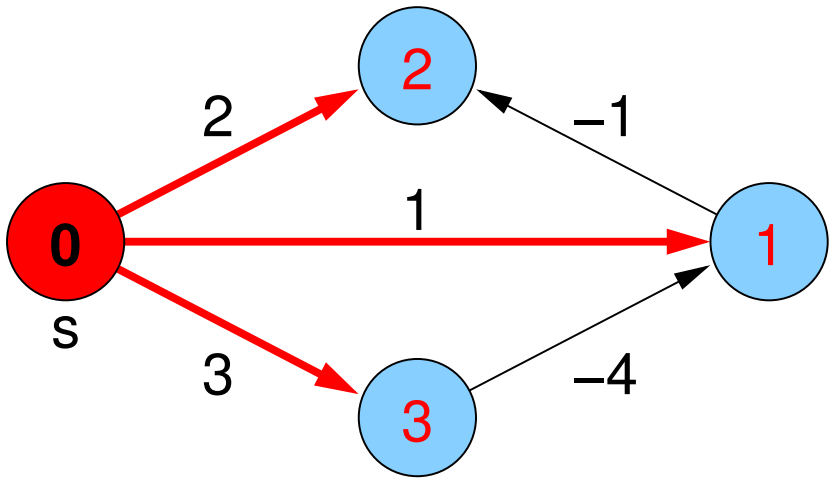
Kürzeste Wege für beliebige Graphen mit beliebigen Gewichten

Gegenbeispiel für Dijkstra-Algorithmus:



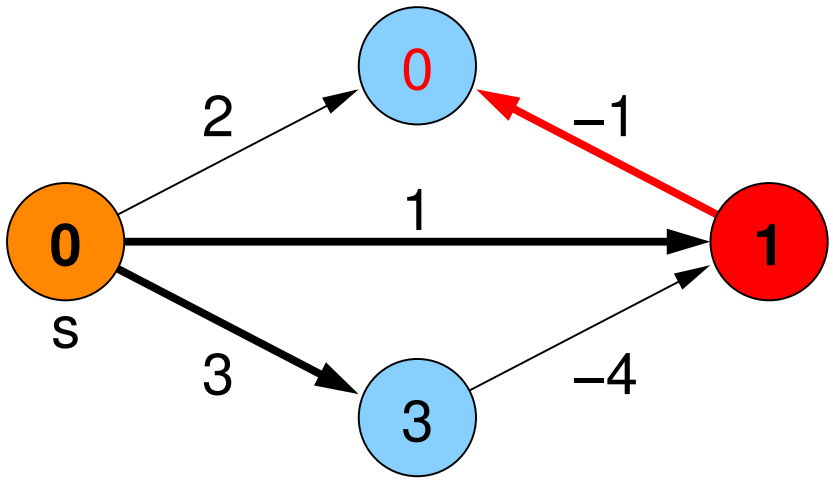
Kürzeste Wege für beliebige Graphen mit beliebigen Gewichten

Gegenbeispiel für Dijkstra-Algorithmus:



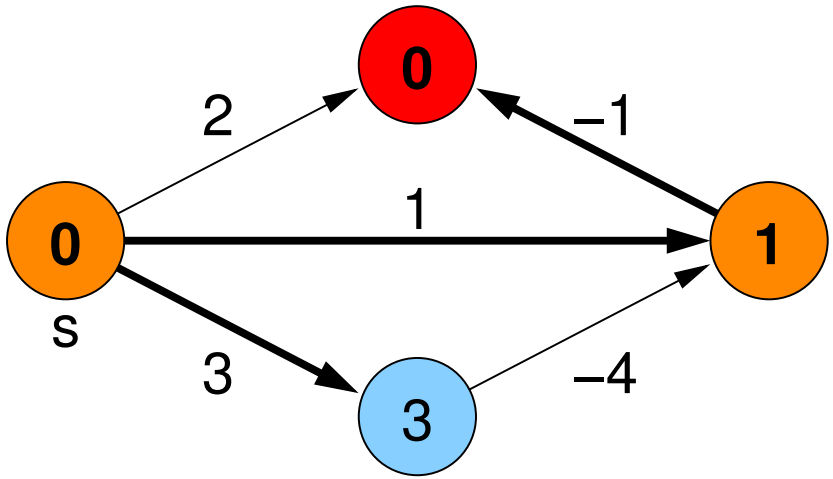
Kürzeste Wege für beliebige Graphen mit beliebigen Gewichten

Gegenbeispiel für Dijkstra-Algorithmus:



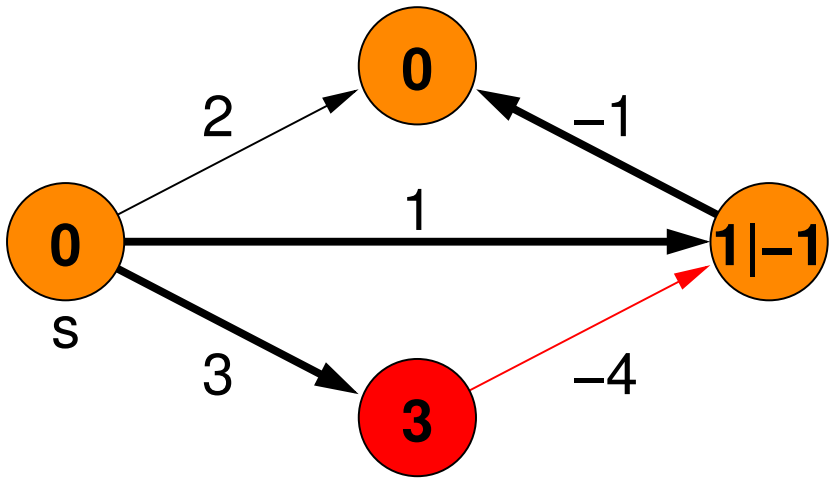
Kürzeste Wege für beliebige Graphen mit beliebigen Gewichten

Gegenbeispiel für Dijkstra-Algorithmus:



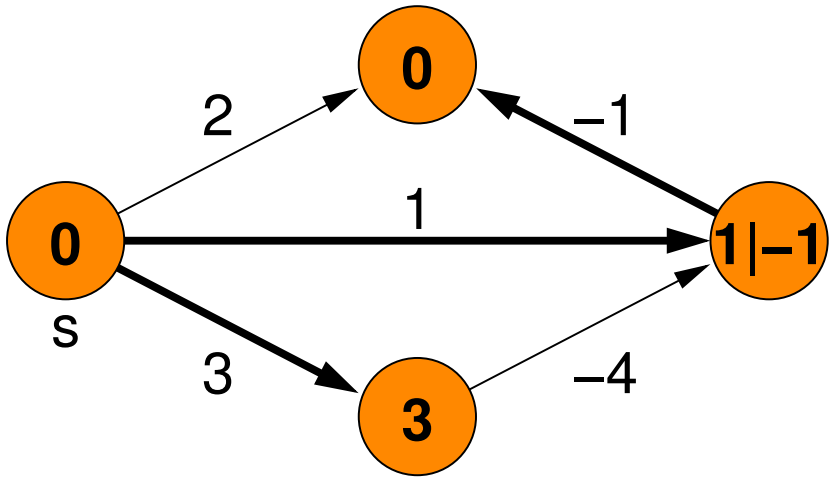
Kürzeste Wege für beliebige Graphen mit beliebigen Gewichten

Gegenbeispiel für Dijkstra-Algorithmus:



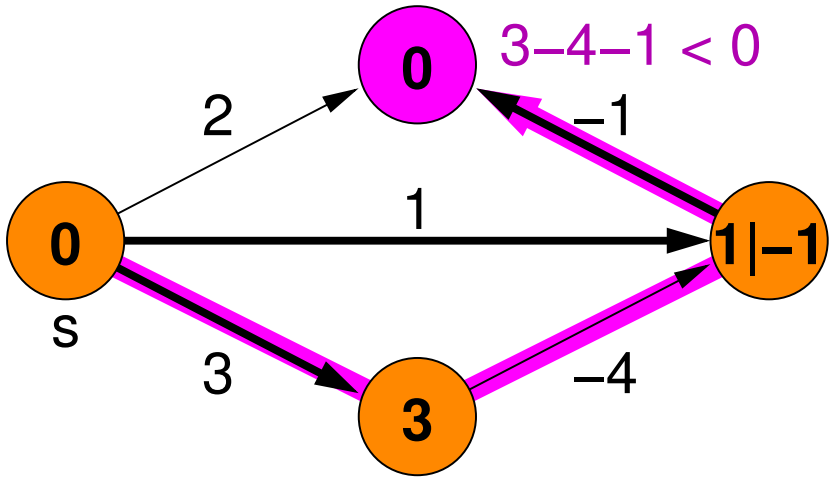
Kürzeste Wege für beliebige Graphen mit beliebigen Gewichten

Gegenbeispiel für Dijkstra-Algorithmus:



Kürzeste Wege für beliebige Graphen mit beliebigen Gewichten

Gegenbeispiel für Dijkstra-Algorithmus:



Kürzeste Wege für beliebige Graphen mit beliebigen Gewichten

Lemma

Für jeden Knoten v mit $d(s,v) > -\infty$ gibt es einen einfachen Pfad (ohne Kreis) von s nach v der Länge $d(s,v)$.

Beweis.

- Weg mit Kreis mit Kantengewichtssumme ≥ 0:
  Entfernen des Kreises erhöht nicht die Kosten
- Weg mit Kreis mit Kantengewichtssumme < 0:
  Distanz von s ist $-\infty$

Bellman-Ford-Algorithmus

Folgerung

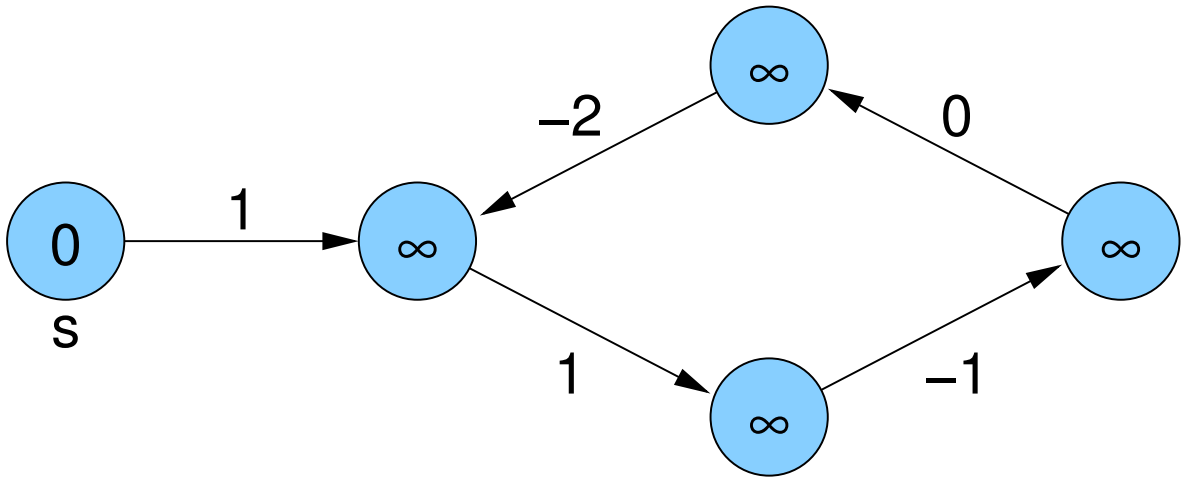
In einem Graph mit n Knoten gibt es für jeden erreichbaren Knoten v mit $d(s,v) > -\infty$ einen kürzesten Weg bestehend aus $< n$ Kanten zwischen s und v.

Strategie:

- anstatt kürzeste Pfade in Reihenfolge wachsender Gewichtssumme zu berechnen, betrachte sie in Reihenfolge steigender Kantenanzahl
- durchlaufe $(n-1)$-mal alle Kanten im Graph
  und aktualisiere die Distanz
- dann alle kürzesten Wege berücksichtigt

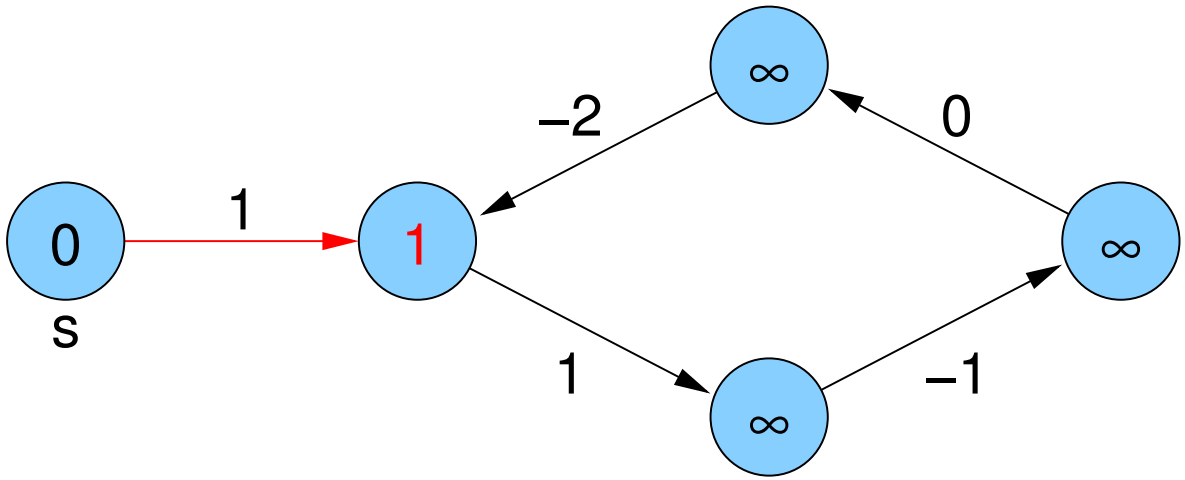
Bellman-Ford-Algorithmus

Problem: Erkennung negativer Kreise



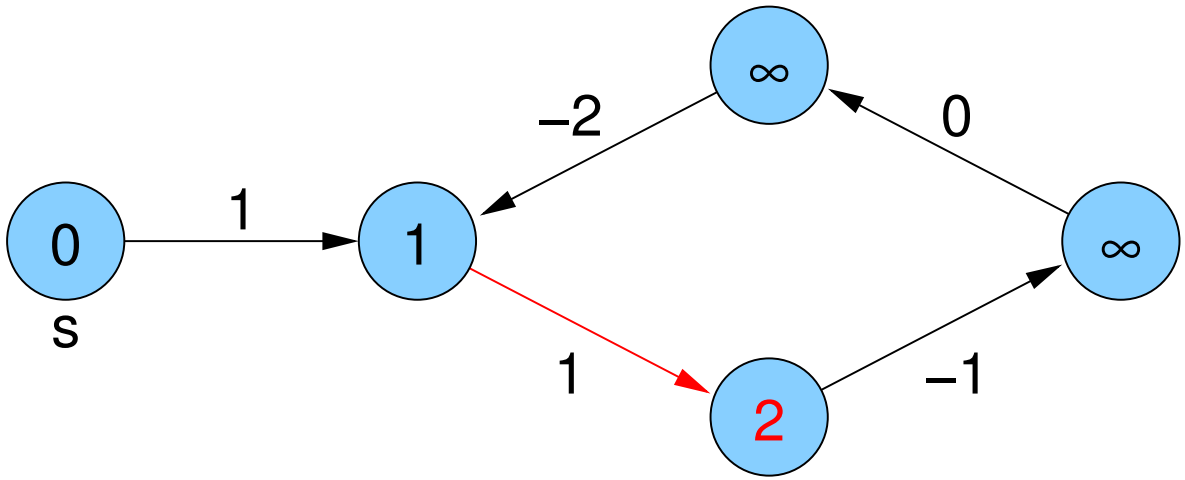
Bellman-Ford-Algorithmus

Problem: Erkennung negativer Kreise



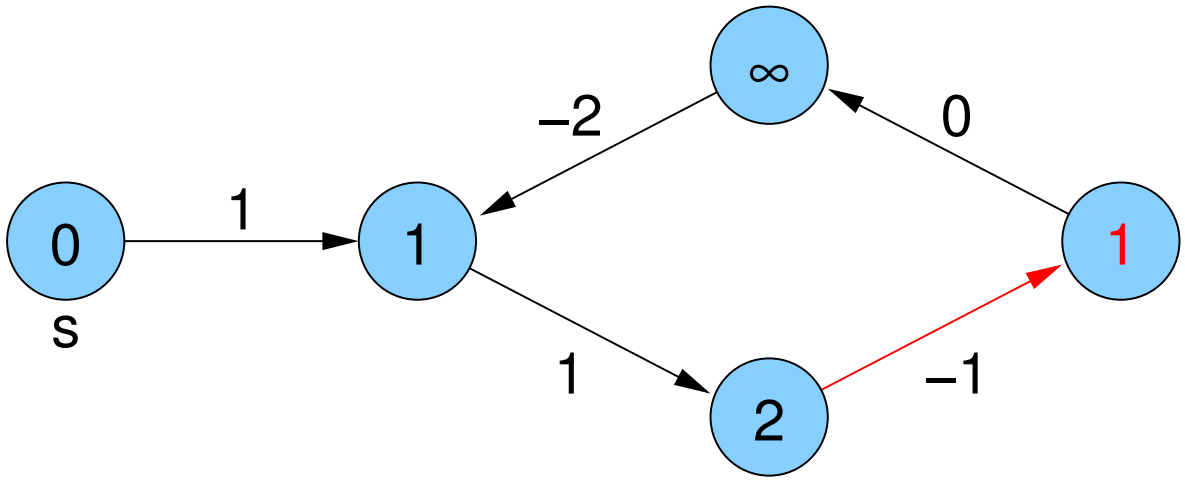
Bellman-Ford-Algorithmus

Problem: Erkennung negativer Kreise



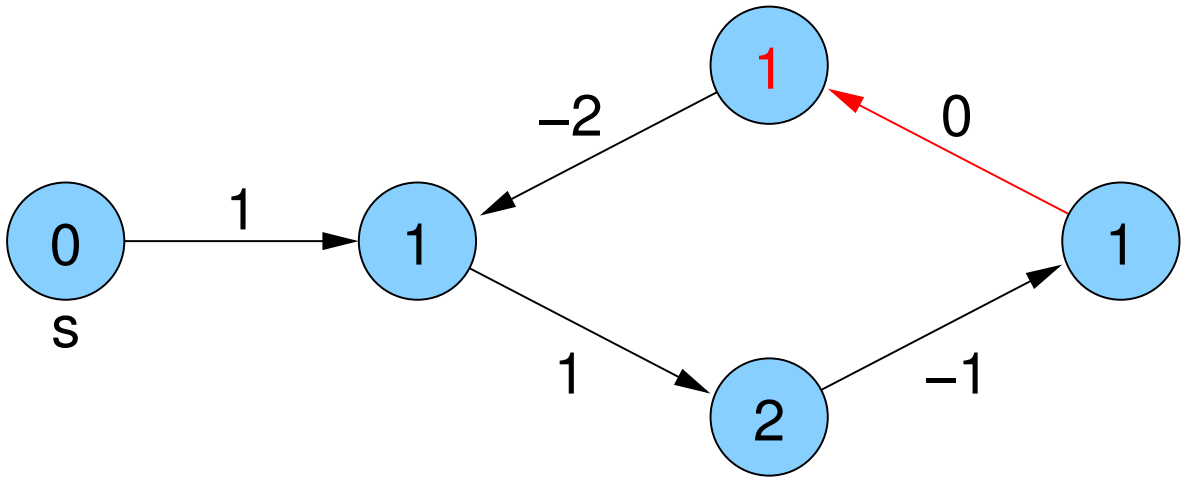
Bellman-Ford-Algorithmus

Problem: Erkennung negativer Kreise



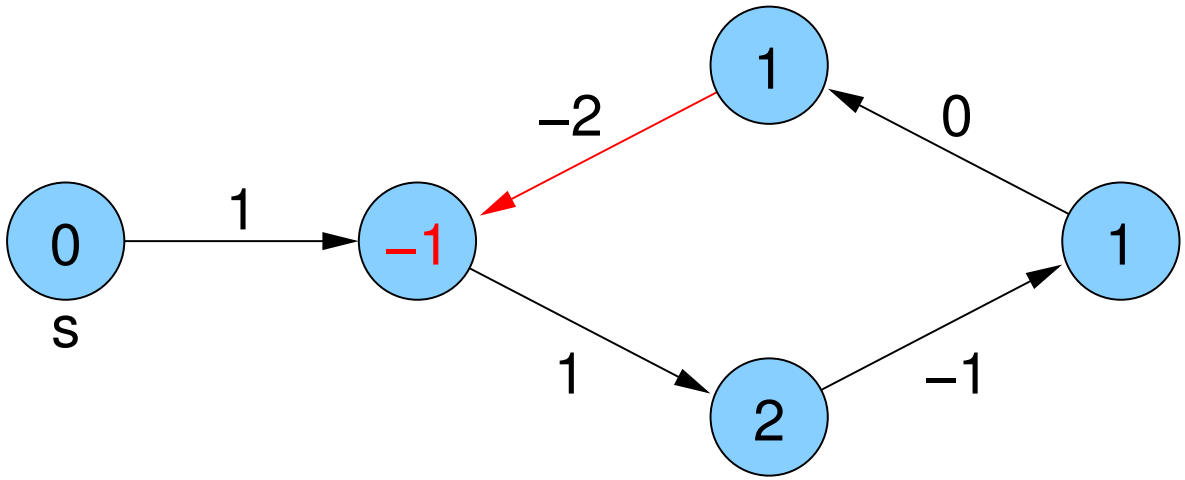
Bellman-Ford-Algorithmus

Problem: Erkennung negativer Kreise



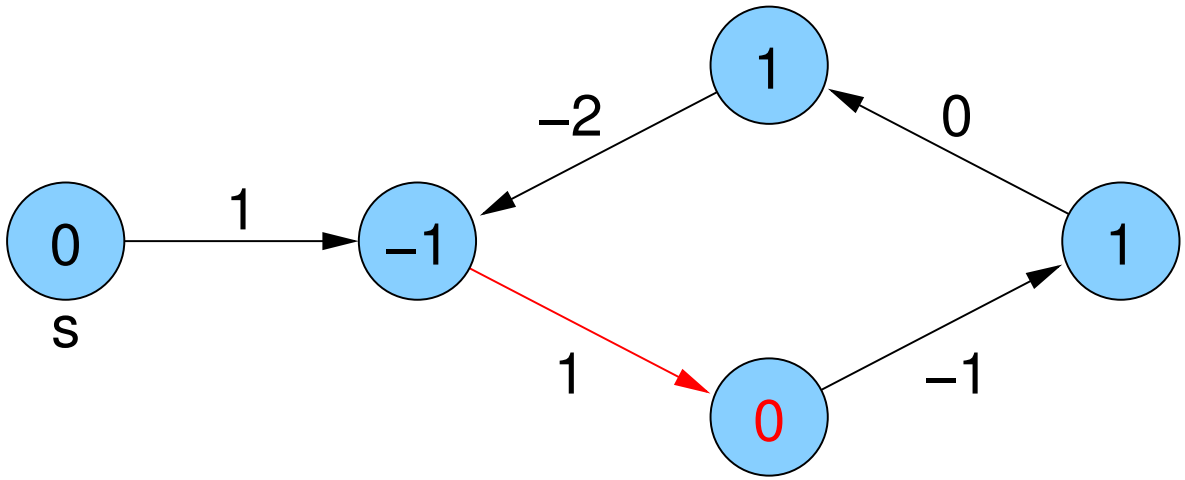
Bellman-Ford-Algorithmus

Problem: Erkennung negativer Kreise



Bellman-Ford-Algorithmus

Problem: Erkennung negativer Kreise



Bellman-Ford-Algorithmus

Keine Distanzerniedrigung möglich:

- Annahme: zu einem Zeitpunkt gilt für alle Kanten (v, w) $d[v] + c(v, w) \geq d[w]$
- dann gilt (per Induktion) für jeden Weg p von s nach w, dass $d[s] + c(p) \geq d[w]$ für alle Knoten w
- falls sichergestellt, dass zu jedem Zeitpunkt für kürzesten Weg p von s nach w gilt $d[w] \geq c(p)$, dann ist $d[w]$ zum Schluss genau die Länge eines kürzesten Pfades von s nach w (also korrekte Distanz)

Bellman-Ford-Algorithmus

Zusammenfassung:

- keine Distanzerniedrigung mehr möglich
  ($d[v] + c(v, w) \geq d[w]$ für alle w):
  fertig, alle $d[w]$ korrekt für alle w

- Distanzerniedrigung möglich selbst noch in n-ter Runde
  ($d[v] + c(v, w) < d[w]$ für ein w):
  Es gibt einen negativen Kreis, also Knoten w mit Distanz $-\infty$.

Bellman-Ford-Algorithmus

```
void BellmanFord(Node s) {
  d[s] = 0;   parent[s] = s;
  for (int i = 0; i < n − 1; i++) {   // n − 1 Runden
    foreach (e = (v, w) ∈ E)
      if (d[v] + c(e) < d[w]) {   // kürzerer Weg?
        d[w] = d[v] + c(e);
        parent[w] = v;
      }
  }
  foreach (e = (v, w) ∈ E)
    if (d[v] + c(e) < d[w]) {   // kürzerer Weg in n-ter Runde?
      infect(w);
}
```

Bellman-Ford-Algorithmus

```
void infect(Node v) {   // −∞-Knoten
  if (d[v] > −∞) {
    d[v] = −∞;
    foreach (e = (v, w) ∈ E)
      infect(w);
  }
}
```

Gesamtlaufzeit: $\mathcal{O}(m \cdot n)$

Bellman-Ford-Algorithmus

Bestimmung der Knoten mit Distanz $-\infty$:

- betrachte alle Knoten, die in der n-ten Phase noch Distanzverbesserung erfahren
- aus jedem Kreis mit negativem Gesamtgewicht muss mindestens ein Knoten dabei sein
- jeder von diesen Knoten aus erreichbare Knoten muss Distanz $-\infty$ bekommen
- das erledigt hier die infect-Funktion
- wenn ein Knoten zweimal auftritt (d.h. der Wert ist schon $-\infty$), wird die Rekursion abgebrochen

Bellman-Ford-Algorithmus

Bestimmung eines negativen Zyklus:

- bei den oben genannten Knoten sind vielleicht auch Knoten, die nur an negativen Kreisen über ausgehende Kanten angeschlossen sind, die selbst aber nicht Teil eines negativen Kreises sind
- Rückwärtsverfolgung der parent-Werte, bis sich ein Knoten wiederholt
- Kanten vom ersten bis zum zweiten Auftreten bilden einen negativen Zyklus

Bellman-Ford-Algorithmus

Ursprüngliche Idee der Updates vorläufiger Distanzwerte stammt von Lester R. Ford Jr.

Verbesserung (Richard E. Bellman / Edward F. Moore):

- verwalte eine FIFO-Queue von Knoten, zu denen ein kürzerer Pfad gefunden wurde und deren Nachbarn am anderen Ende ausgehender Kanten noch auf kürzere Wege geprüft werden müssen
- wiederhole: nimm ersten Knoten aus der Queue und prüfe für jede ausgehende Kante die Distanz des Nachbarn

  falls kürzerer Weg gefunden, aktualisiere Distanzwert des Nachbarn und hänge ihn an Queue an (falls nicht schon enthalten)
- Phase besteht immer aus Bearbeitung der Knoten, die am Anfang des Algorithmus (bzw. der Phase) in der Queue sind
  (dabei kommen während der Phase schon neue Knoten ans Ende der Queue)

Kürzeste einfache Pfade bei beliebigen Kantengewichten

Achtung!

Fakt

Die Suche nach kürzesten einfachen Pfaden
(also ohne Knotenwiederholungen / Kreise)
in Graphen mit beliebigen Kantengewichten
(also möglichen negativen Kreisen)
ist ein $\mathcal{NP}$-vollständiges Problem.

(Man könnte Hamilton-Pfad-Suche damit lösen.)

All Pairs Shortest Paths

gegeben:

- Graph mit beliebigen Kantengewichten,
  der aber keine negativen Kreise enthält

gesucht:

- Distanzen / kürzeste Pfade zwischen allen Knotenpaaren

Naive Strategie:

- n-mal Bellman-Ford-Algorithmus
  (jeder Knoten einmal als Startknoten)

$\Rightarrow \mathcal{O}(n^2 \cdot m)$

All Pairs Shortest Paths

Bessere Strategie:

- reduziere n Aufrufe des Bellman-Ford-Algorithmus auf n Aufrufe des Dijkstra-Algorithmus

Problem:

- Dijkstra-Algorithmus funktioniert nur für nichtnegative Kantengewichte

Lösung:

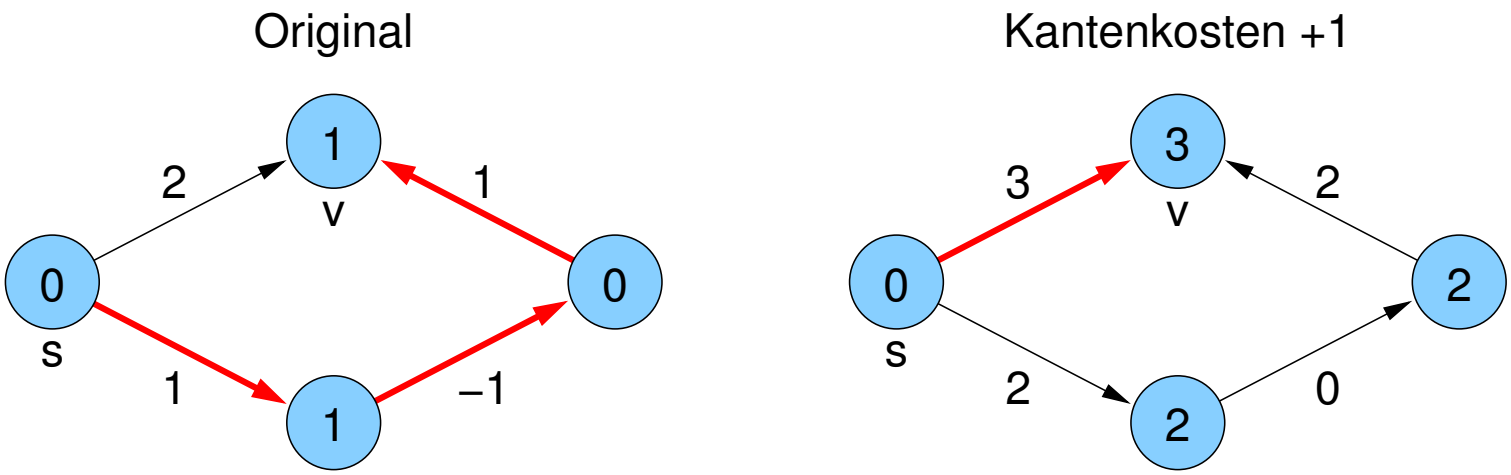
- Umwandlung in nichtnegative Kantenkosten ohne Verfälschung der kürzesten Wege

All Pairs Shortest Paths

Naive Idee:

- negative Kantengewichte eliminieren, indem auf jedes Kantengewicht der gleiche Wert c addiert wird

$\Rightarrow$ verfälscht kürzeste Pfade



All Pairs Shortest Paths

Sei $\Phi : V \mapsto \mathbb{R}$ eine Funktion, die jedem Knoten ein Potential zuordnet.

Modifizierte Kantenkosten von $e = (v, w)$:

$$\bar{c}(e) = \Phi(v) + c(e) - \Phi(w)$$

Lemma

Seien p und q Wege von v nach w in G.

$c(p)$ und $c(q)$ bzw. $\bar{c}(p)$ und $\bar{c}(q)$ seien die aufsummierten Kosten bzw. modifizierten Kosten der Kanten des jeweiligen Pfads.

Dann gilt für jedes Potential Φ:

$$\bar{c}(p) < \bar{c}(q) \quad \Leftrightarrow \quad c(p) < c(q)$$

All Pairs Shortest Paths

Beweis.

Sei $p = (v_1, \ldots, v_k)$ beliebiger Weg und $\forall i: \; e_i = (v_i, v_{i+1}) \in E$

Es gilt:

$$\begin{aligned} \bar{c}(p) &= \sum_{i=1}^{k-1} \bar{c}(e_i) \\ &= \sum_{i=1}^{k-1} (\Phi(v_i) + c(e_i) - \Phi(v_{i+1})) \\ &= \Phi(v_1) + c(p) - \Phi(v_k) \end{aligned}$$

d.h. modifizierte Kosten eines Pfads hängen nur von ursprünglichen Pfadkosten und vom Potential des Anfangs- und Endknotens ab.
(Im Lemma ist $v_1 = v$ und $v_k = w$) □

All Pairs Shortest Paths

Lemma

Annahme:

- *Graph hat keine negativen Kreise*
- *alle Knoten von s aus erreichbar*

Sei für alle Knoten v das Potential $\Phi(v) = d(s, v)$.

Dann gilt für alle Kanten e: $\bar{c}(e) \geq 0$

Beweis.

- für alle Knoten v gilt nach Annahme: $d(s, v) \in \mathbb{R}$ (also $\neq \pm\infty$)
- für jede Kante $e = (v, w)$ ist

$$\begin{aligned} d(s, v) + c(e) &\geq d(s, w) \\ d(s, v) + c(e) - d(s, w) &\geq 0 \end{aligned}$$

□

All Pairs Shortest Paths / Johnson-Algorithmus

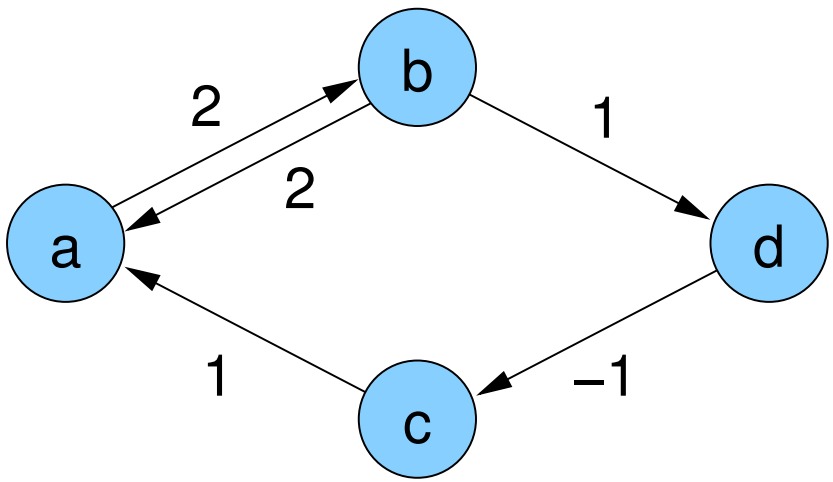
- füge neuen Knoten s und Kanten (s, v) für alle v hinzu mit $c(s, v) = 0$

$\Rightarrow$ alle Knoten erreichbar

- berechne $d(s, v)$ mit Bellman-Ford-Algorithmus
- setze $\Phi(v) = d(s, v)$ für alle v
- berechne modifizierte Kosten $\bar{c}(e)$
- berechne für alle Knoten v die Distanzen $\bar{d}(v, w)$ mittels Dijkstra-Algorithmus mit modifizierten Kantenkosten auf dem Graph ohne Knoten s
- berechne korrekte Distanzen $d(v, w) = \bar{d}(v, w) + \Phi(w) - \Phi(v)$

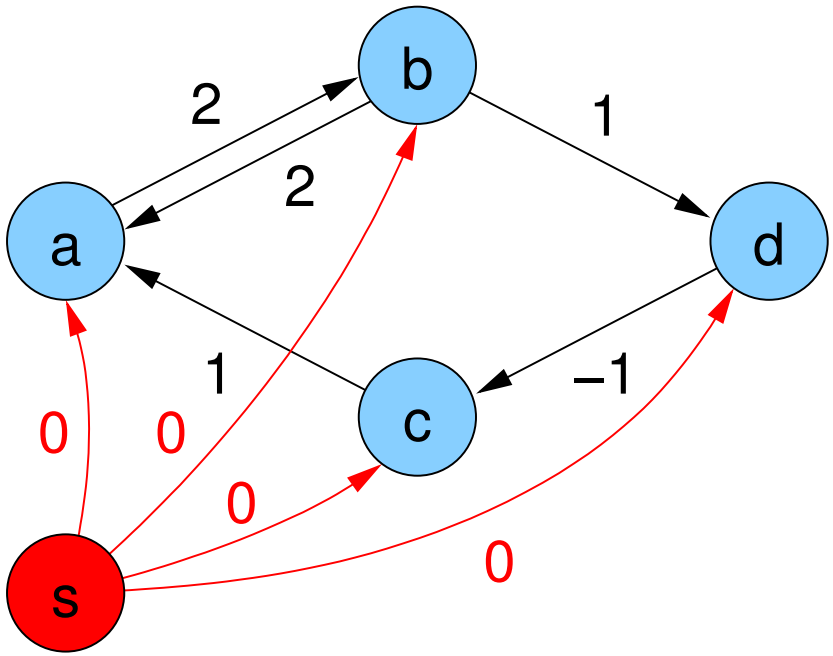
All Pairs Shortest Paths / Johnson-Algorithmus

Beispiel:



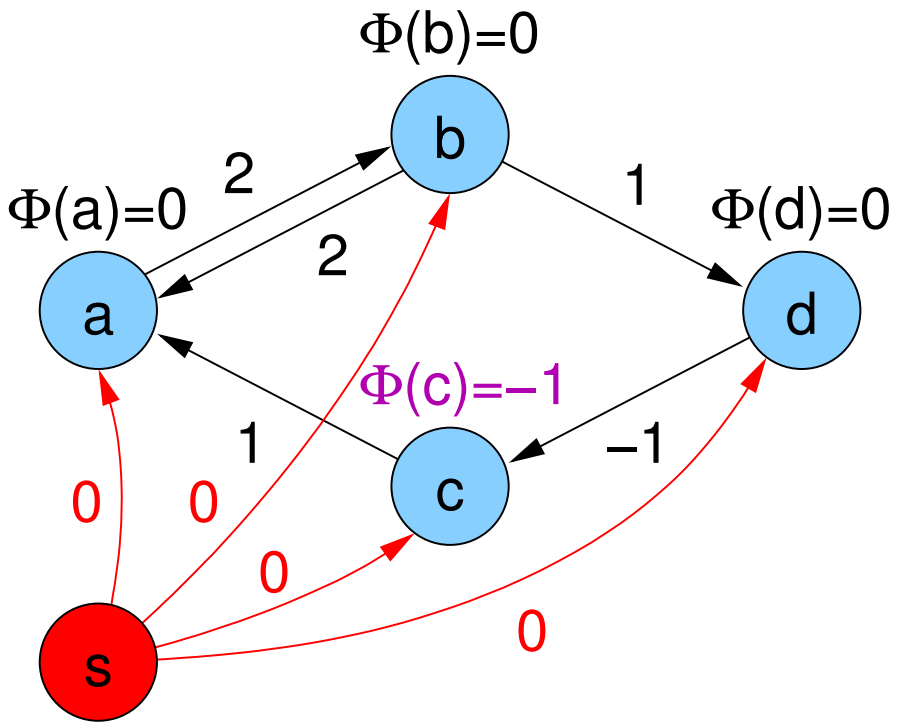
All Pairs Shortest Paths / Johnson-Algorithmus

1. künstliche Quelle s:



All Pairs Shortest Paths / Johnson-Algorithmus

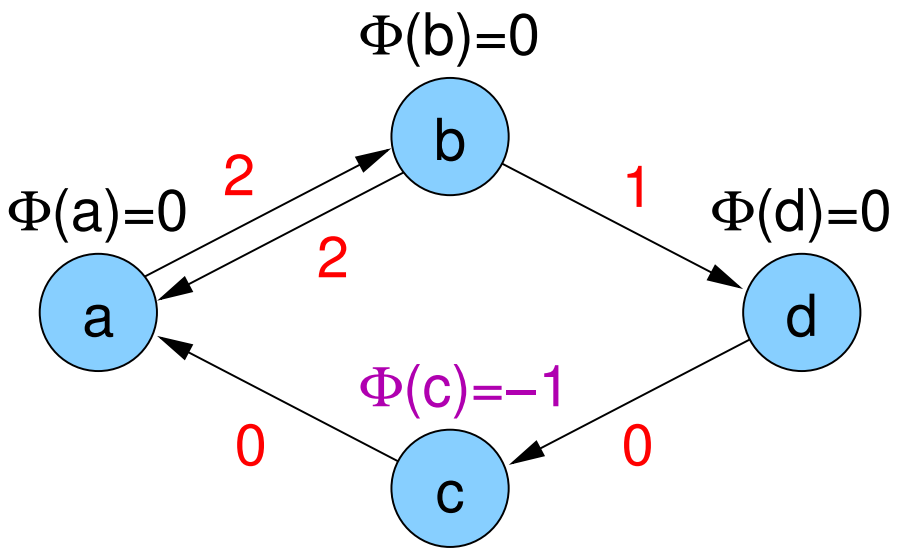
2. Bellman-Ford-Algorithmus auf s:



All Pairs Shortest Paths / Johnson-Algorithmus

3. $\bar{c}(e)$-Werte für alle $e = (v, w)$ berechnen:

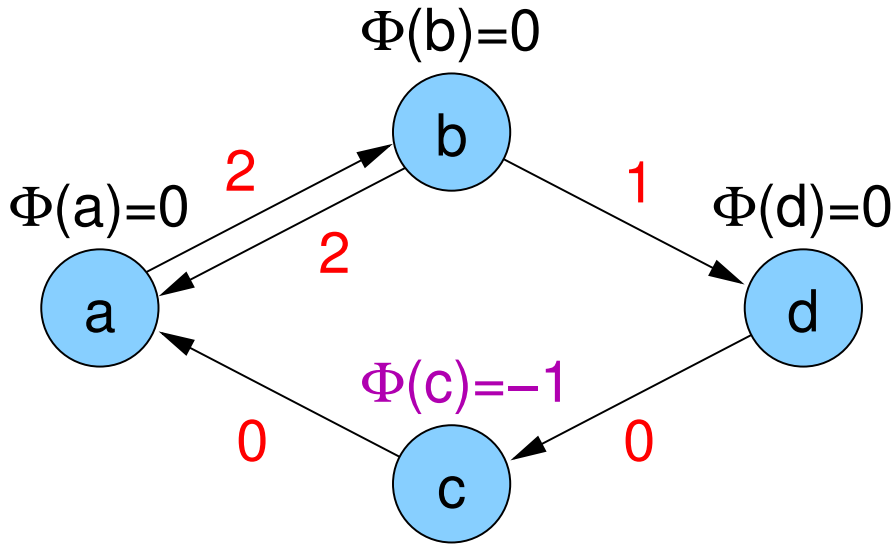
$$\bar{c}(e) = \Phi(v) + c(e) - \Phi(w)$$



All Pairs Shortest Paths / Johnson-Algorithmus

4. Distanzen $\bar{d}$ mit modifizierten Kantengewichten via Dijkstra:

$\bar{d}$	a	b	c	d
a	0	2	3	3
b	1	0	1	1
c	0	2	0	3
d	0	2	0	0

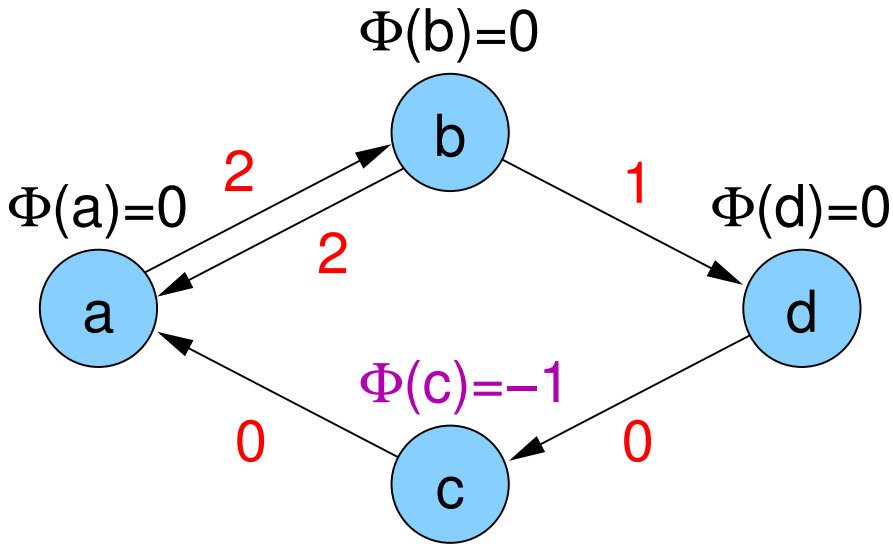


All Pairs Shortest Paths / Johnson-Algorithmus

5. korrekte Distanzen berechnen mit Formel

$$d(v,w) = \bar{d}(v,w) + \Phi(w) - \Phi(v)$$

d	a	b	c	d
a	0	2	2	3
b	1	0	0	1
c	1	3	0	4
d	0	2	−1	0



All Pairs Shortest Paths / Johnson-Algorithmus

Laufzeit:

$$\begin{aligned} T(APSP) &= \mathcal{O}(T_{\text{Bellman-Ford}}(n+1, m+n) + n \cdot T_{\text{Dijkstra}}(n, m)) \\ &= \mathcal{O}((m+n) \cdot (n+1) + n(n \log n + m)) \\ &= \mathcal{O}(m \cdot n + n^2 \log n) \end{aligned}$$

(bei Verwendung von Fibonacci Heaps)

APSP / Floyd-Warshall-Algorithmus

Grundlage:

- geht der kürzeste Weg von u nach w über v, dann sind auch die beiden Teile von u nach v und von v nach w kürzeste Pfade zwischen diesen Knoten
- Annahme: alle kürzeste Wege bekannt, die nur über Zwischenknoten mit Index kleiner als k gehen

$\Rightarrow$ kürzeste Wege über Zwischenknoten mit Indizes bis einschließlich k können leicht berechnet werden:

- entweder der schon bekannte Weg über Knoten mit Indizes kleiner als k
- oder über den Knoten mit Index k (hier im Algorithmus der Knoten v)

APSP / Floyd-Warshall-Algorithmus

Algorithmus 3 : Floyd-Warshall APSP Algorithmus

Input : Graph $G = (V, E)$, $c : E \to R$

Output : Distanzen $d(u, v)$ zwischen allen $u, v \in V$

for $u, v \in V$ **do**
  $d(u, v) = \infty$; $\text{pred}(u, v) = 0$;
for $v \in V$ **do** $d(v, v) = 0$;
for $\{u, v\} \in E$ **do**
  $d(u, v) = c(u, v)$; $\text{pred}(u, v) = u$;
for $v \in V$ **do**
  for $\{u, w\} \in V \times V$ **do**
    if $d(u, w) > d(u, v) + d(v, w)$ **then**
      $d(u, w) = d(u, v) + d(v, w)$;
      $\text{pred}(u, w) = \text{pred}(v, w)$;

APSP / Floyd-Warshall-Algorithmus

- Komplexität: $\mathcal{O}(n^3)$
- funktioniert auch, wenn Kanten mit negativem Gewicht existieren
- Kreise negativer Länge werden nicht direkt erkannt und verfälschen das Ergebnis, sind aber indirekt am Ende an negativen Diagonaleinträgen der Distanzmatrix erkennbar