Algorithmen und Datenstrukturen

SSSP für Graphen mit neg. Kantenkosten
All-Pairs-Shortest-Paths-Algorithmen

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Felix Kuhr (Übungen)

sowie viele Tutoren

Danksagung

Die nachfolgenden Präsentationen wurden mit ausdrücklicher Erlaubnis des Autors übernommen aus:

- „Effiziente Algorithmen und Datenstrukturen“ (Kapitel 7,8,9) gehalten von Christian Scheideler an der TUM http://www14.in.tum.de/lehre/2008WS/ea/index.html.de

Es wurden Veränderungen vorgenommen, und Fehler sind uns anzulasten

Anwendung: Ereignisplanung

- Ereignis 2 mind. 2 Tage nach Ereignis 1 $x_2 - x_1 > 2$
- Ereignis 3 mind. 6 Tage nach Ereignis 1 $x_3 - x_1 > 6$
- Ereignis 4 nicht mehr als 5 Tage vor Ereignis 3
- Ereignis 2 nicht mehr als 4 Tage vor Ereignis 3
- Ereignis 4 nicht mehr als 1 Tag vor Ereignis 2
- Ereignis 1 nicht mehr als 1 Tag vor Ereignis 4

- Tag von Ereignis i = x_i
- Lösung des Ungleichungssystems ergibt Ereignisplanung

$$x_1 - x_2 <= -2$$
$$x_1 - x_3 <= -6$$
$$x_3 - x_4 <= 5$$
$$x_3 - x_2 <= 4$$
$$x_2 - x_4 <= 1$$
$$x_4 - x_1 <= 1$$

Umwandlung in gerichteten Graphen

$x_1 - x_2 <= -2$

$x_1 - x_3 <= -6$

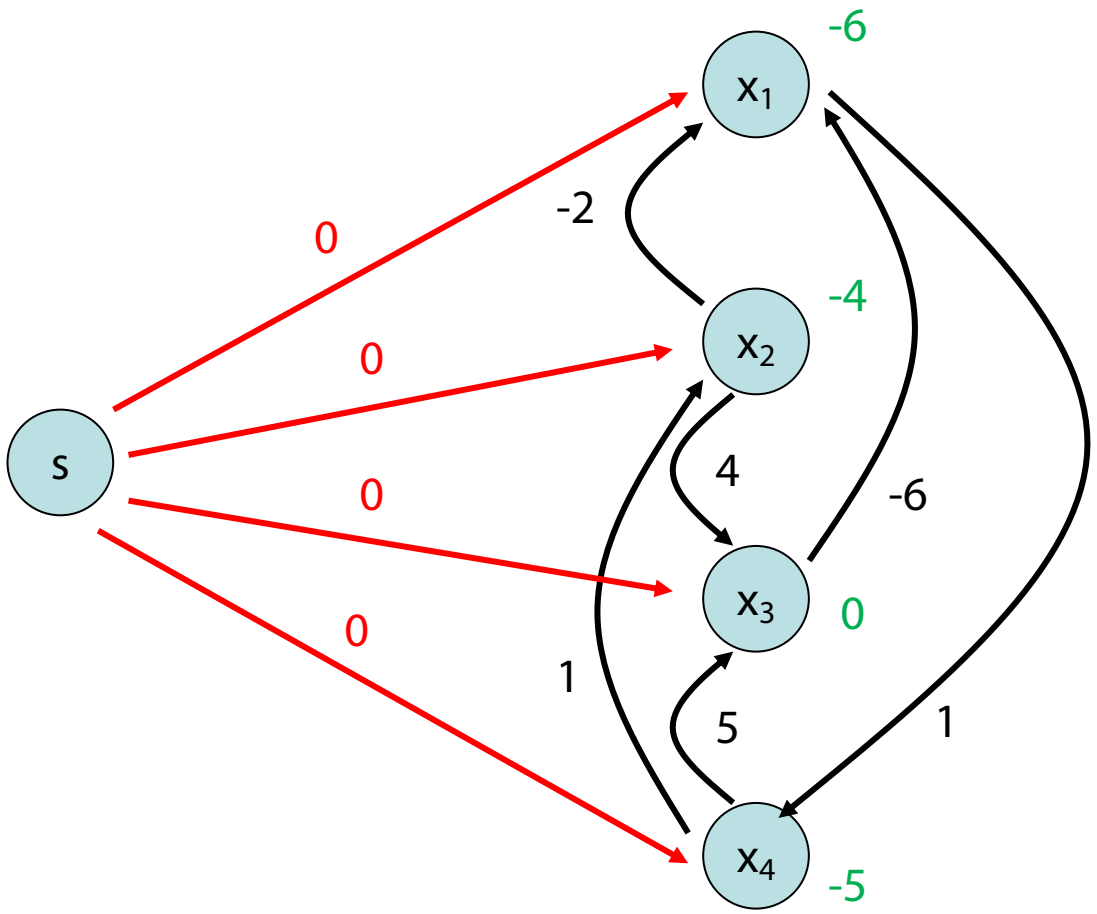
$x_3 - x_4 <= 5$

$x_3 - x_2 <= 4$

$x_2 - x_4 <= 1$

$x_4 - x_1 <= 1$

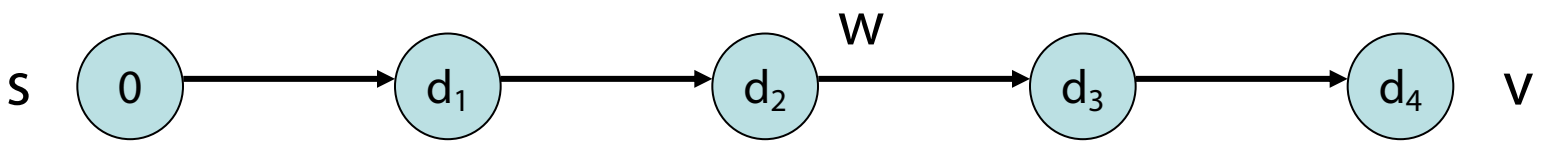
Lösung: $\mu(s,x_i)$



Bellman-Ford Algorithmus

Kürzeste Wege für beliebige Graphen mit beliebigen Kantenkosten (aber noch SSSP).

Problem: besuche Knoten eines kürzesten Weges in richtiger Reihenfolge

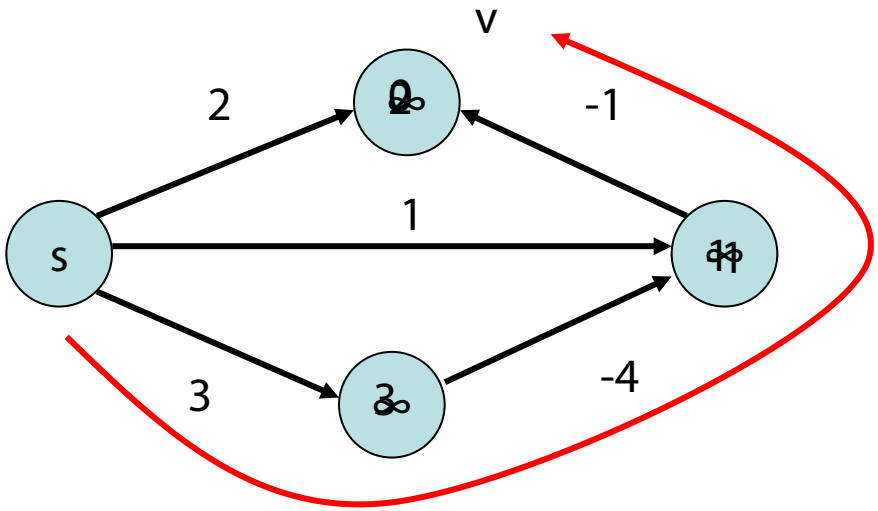


- Dijkstra Algo kann nicht verwendet werden, da im Allgemeinen nicht mehr die Knoten in der Reihenfolge ihrer Distanz zu s besucht werden

R. E. Bellman: On a Routing Problem. In: Quarterly of Applied Mathematics. 16(1). Brown University, S. 87-90, **1958**

Bellman-Ford Algorithmus

Beispiel für Problem mit Dijkstra Algo:



Knoten v hat falschen Distanzwert!

Bellman-Ford Algorithmus

Beh: Für jeden Knoten v mit $\mu(s,v) > -\infty$ zu s gibt es einfachen Weg (ohne Kreis!) von s nach v mit Kosten $\mu(s,v)$.

Beweis:

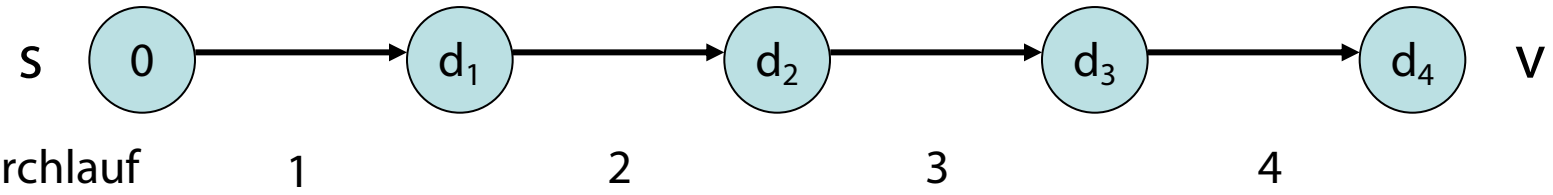
- Weg mit Kreis mit Kantenkosten ≥ 0: Entfernen des Kreises erhöht nicht die Kosten
- Weg mit Kreis mit Kantenkosten < 0: Distanz zu s ist $-\infty$!

Bellman-Ford Algorithmus

Folgerung: (für Graph mit n Knoten)

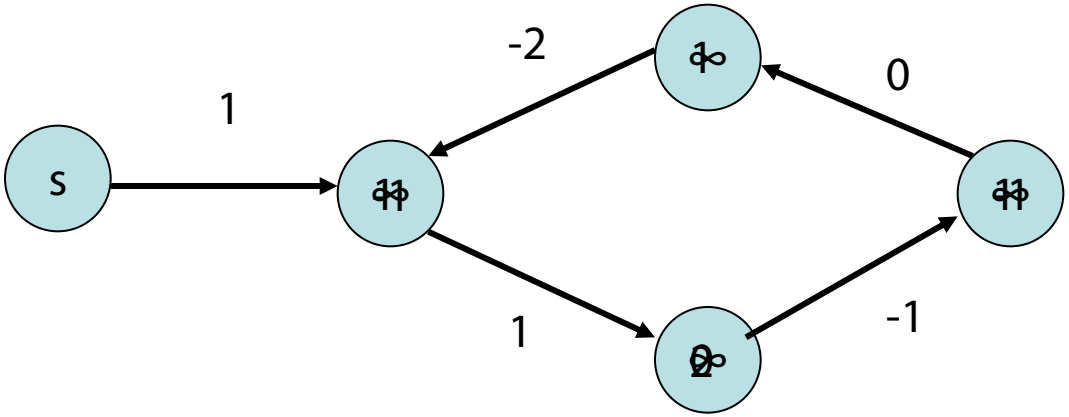
Für jeden Knoten v mit $\mu(s,v) > -\infty$ gibt es kürzesten Weg der Länge (Anzahl Kanten!) <n zu v

Strategie: Durchlaufe (n-1)-mal sämtliche Kanten in Graph und aktualisiere Distanz. Dann alle kürzesten Wege berücksichtigt.



Bellman-Ford Algorithmus

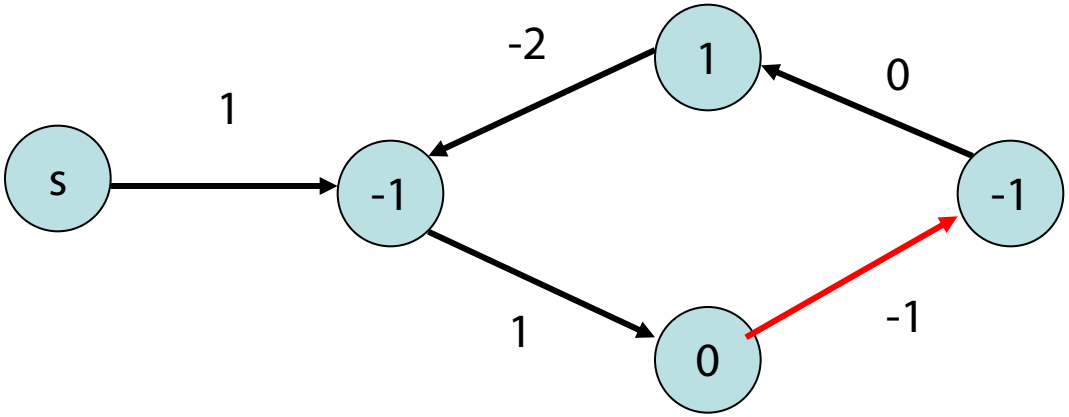
Problem: Erkennung negativer Kreise



Einsicht: in negativem Kreis erniedrigt sich Distanz in jeder Runde bei mindestens einem Knoten

Bellman-Ford Algorithmus

Problem: Erkennung negativer Kreise



Zeitpunkt: kontinuierliche Distanzerniedrigung startet spätestens in n-ter Runde (dann Kreis mindestens einmal durchlaufen)

Bellman-Ford Algorithmus

Keine Distanzerniedrigung möglich:

- Angenommen, wir erreichen Zeitpunkt mit $d[v]+c(v,w) \geq d[w]$ für alle Knoten w.
- Dann gilt (über Induktion) für jeden Weg p, dass $d[s]+c(p) \geq d[w]$ für alle Knoten w.
- Falls sichergestellt ist, dass für den kürzesten Weg p nach w, $d[w] \geq c(p)$ zu jedem Zeitpunkt ist, dann gilt am Ende $d[w]=\mu(s,w)$.

Bellman-Ford Algorithmus

Zusammenfassung:

- Keine Distanzerniedrigung mehr möglich ($d[v]+c(v,w) \geq d[w]$ für alle w):
  Fertig, $d[w]=\mu(s,w)$ für alle w

- Distanzerniedrigung möglich selbst noch in n-ter Runde ($d[v]+c(v,w)<d[w]$ für ein w):
  Dann gibt es negative Kreise, also Knoten w mit Distanz $\mu(s,w)=-\infty$. Ist das wahr für ein w, dann für alle von w erreichbaren Knoten (Infektion).

Bellman-Ford Algorithmus

```
procedure BellmanFord(s: NodeId)
    d=<∞,…,∞>: Array of ℝ U {-∞,∞}
    parent=<⊥,…,⊥>: Array of NodeId
    d[s]:=0; parent[s]:=s
    for i:=1 to n-1 do  // aktualisiere Kosten für n-1 Runden
       for e=(v,w) ∈ E do
          if d[w] > d[v]+c(e) then  // bessere Distanz möglich?
             d[w]:=d[v]+c(e); parent[w]:=v
    for e=(v,w) ∈ E do       // in n-ter Runde noch besser?
       if d[w] > d[v]+c(e) then infect(w)


procedure infect(v)  // propagiere -∞-Kosten von v aus
    if d[v]>-∞ then     // noch nicht markiert?
       d[v]:=-∞
       for (v,w) ∈ E do infect(w)
```

Bellman-Ford Algorithmus

Laufzeit: $O(n \cdot m)$

Verbesserungsmöglichkeiten:

- Überprüfe in jeder Aktualisierungsrunde, ob noch irgendwo $d[v]+c[v,w]<d[w]$ ist.
  Nein: fertig!
  (Hauptschleife kann frühzeitig verlassen werden)
- Besuche in jeder Runde nur die Knoten w, für die Test $d[v]+c[v,w]<d[w]$ sinnvoll (d.h. $d[v]$ hat sich in letzter Runde geändert).

All Pairs Shortest Paths

Annahme: Graph mit beliebigen Kantenkosten, aber keine negativen Kreise

Naive Strategie für Graph mit n Knoten: lass n-mal Bellman-Ford Algorithmus (einmal für jeden Knoten) laufen

Laufzeit: $O(n^2 \cdot m)$

All Pairs Shortest Paths

Bessere Strategie: Reduziere n Bellman-Ford Anwendungen auf n Dijkstra Anwendungen (auch Johnson-Dijkstra-Algorithmus genannt)

Problem: wir brauchen dazu nichtnegative Kantenkosten

Lösung: Umwandlungsstrategie in nichtnegative Kantenkosten, ohne kürzeste Wege zu verfälschen (nicht so einfach!)

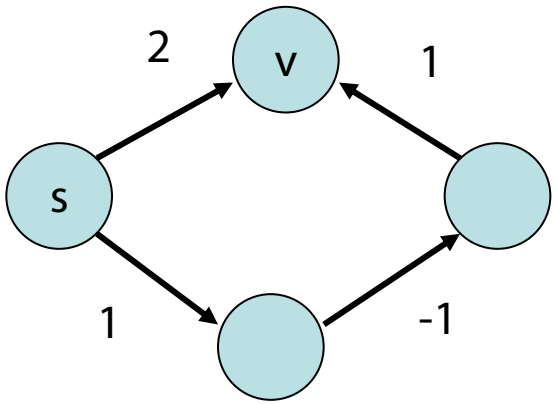
Dijkstra erfordert, dass jeder Knoten über kürzesten Weg erreichbar

Johnson, Donald B., Efficient algorithms for shortest paths in sparse networks, Journal of the ACM 24 (1): pp. 1–13, **1977**

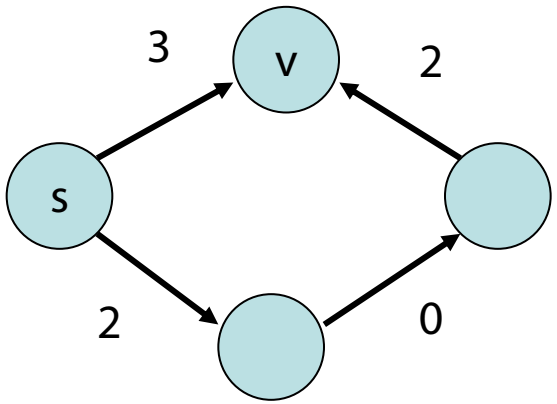
Naive Erhöhung?

- Naive Umwandlung negativer Kantenkosten durch Addition von $c := - \min (\{ 0 \} \cup \{ c(e) \mid e \in E, c(e) < 0 \})$ geht im Allgemeinen schief
- Gegenbeispiel zur Erhöhung um Wert c:

Vorher



Kosten +1 überall



: kürzester Weg

Neuer Ansatz

- Sei ϕ:V$\longrightarrow \mathbb{R}$ eine Funktion, die jedem Knoten ein Potenzial zuweist.
- Die reduzierten Kosten von e=(v,w) sind:
  $r(e) := \phi(v) + c(e) - \phi(w)$
- Kantenkosten r und c in offensichtlicher Weise auf Wegekosten erweiterbar

Beh: Seien p und q Wege in G von v_1 bis v_k. Dann gilt für jedes Potenzial ϕ : r(p)<r(q) genau dann wenn c(p)<c(q).

All Pairs Shortest Paths

Beh : Seien p und q Wege in G von v_1 bis v_k. Dann gilt für jedes Potenzial ϕ : $r(p)<r(q)$ genau dann, wenn $c(p)<c(q)$.

Beweis: Sei $p=(v_1,\ldots,v_k)$ ein beliebiger Weg und $e_i=(v_i,v_{i+1})$ für alle i. Es gilt:

$$
\begin{aligned}
r(p) &= \Sigma_i\, r(e_i) \\
&= \Sigma_i\, (\phi(v_i) + c(e_i) - \phi(v_{i+1})) \\
&= \phi(v_1) + c(p) - \phi(v_k)
\end{aligned}
$$

All Pairs Shortest Paths

Beh: Angenommen, G habe keine negativen Kreise und alle Knoten können von s erreicht werden. Sei $\phi(v)=\mu(s,v)$ für alle $v \in V$.
Mit diesem ϕ ist $r(e)=\phi(v) + c(e) - \phi(w) \geq 0$
für alle $e = (v, w) \in E$.

Beweis:

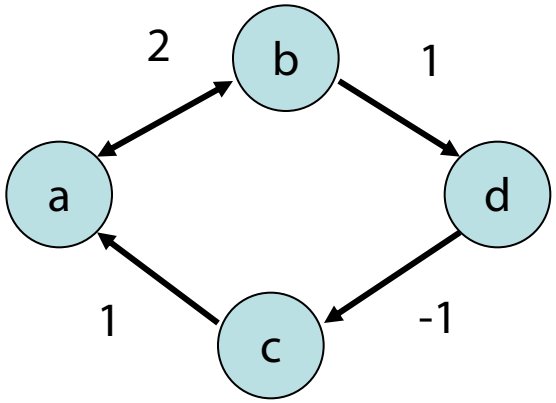
- Nach Annahme ist $\mu(s,v) \in \mathbb{R}$ für alle v
- Wir wissen: Für jede Kante $e=(v,w)$ ist
  $\mu(s,v)+c(e)\geq\mu(s,w)$ (wg. Minimalität von μ)
- Also ist $r(e) = \mu(s,v) + c(e) - \mu(s,w) \geq 0$

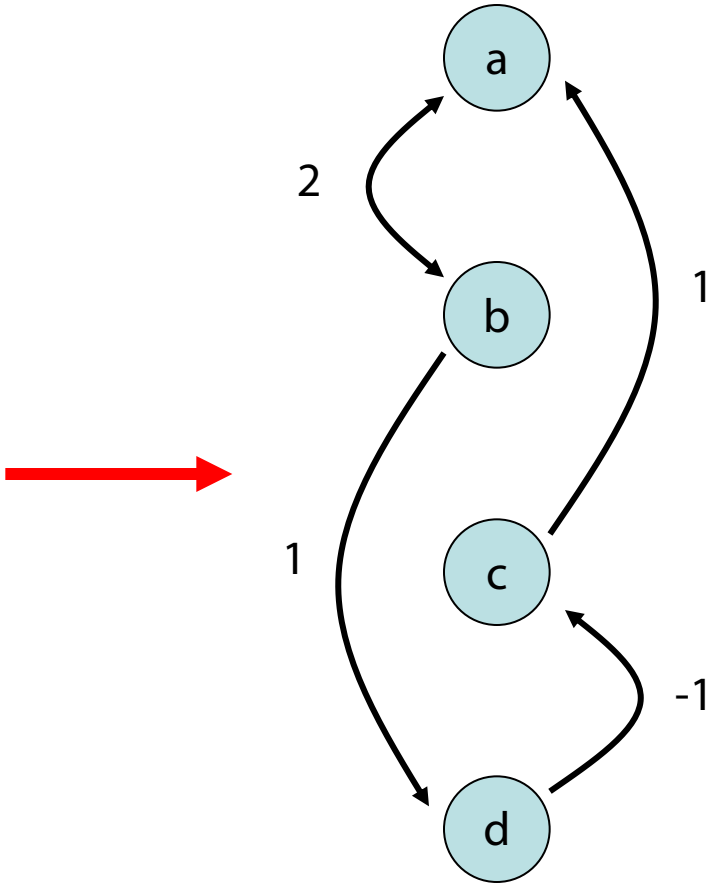
All Pairs Shortest Paths

1. Füge neuen Knoten s und Kanten (s,v) für alle v hinzu mit c(s,v)=0 (alle erreichbar!)
2. Berechne $\mu(s,v)$ nach Bellman-Ford und setze $\phi(v):=\mu(s,v)$ für alle v
3. Berechne die reduzierten Kosten für e = (v, w)
   $r(e) := \phi(v) + c(e) - \phi(w)$
4. Berechne für alle Knoten v die Distanzen $\bar{\mu}(v,w)$ mittels Dijkstra Algorithmus mit reduzierten Kosten auf Graph ohne Knoten s
5. Berechne korrekte Distanzen $\mu(v,w)$ durch $\mu(v,w)=\bar{\mu}(v,w)+ \phi(w)- \phi(v)$

All Pairs Shortest Paths

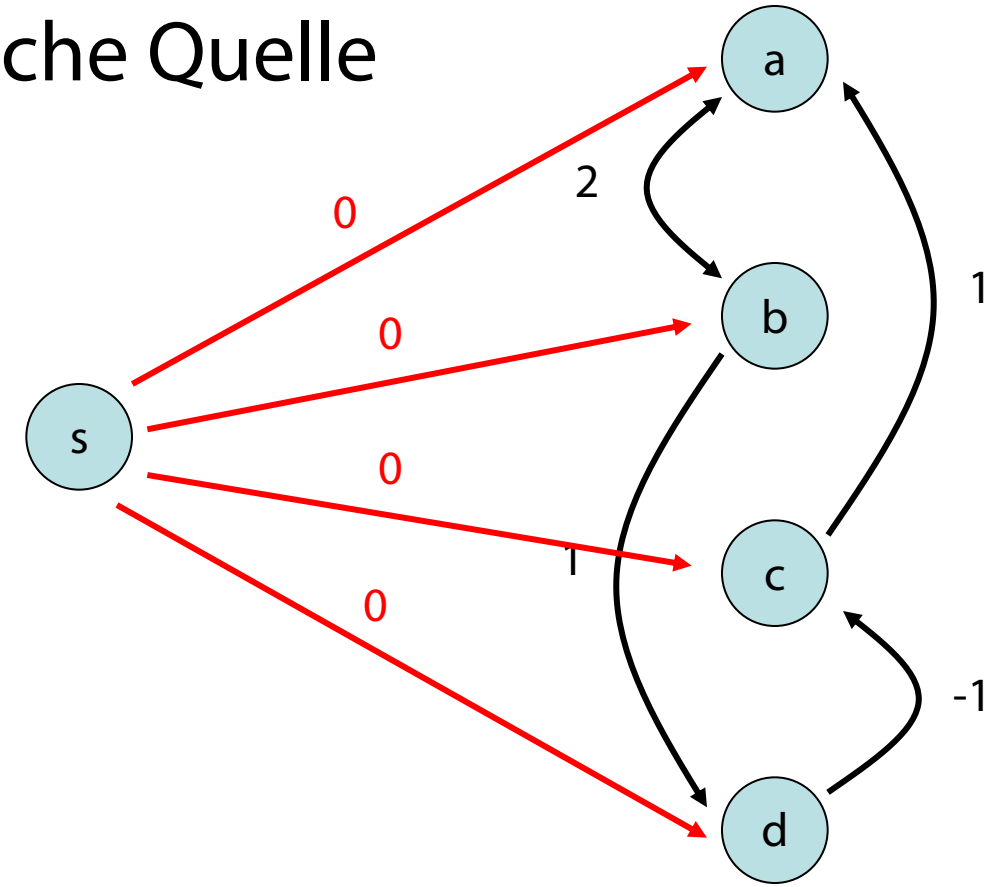
Beispiel:





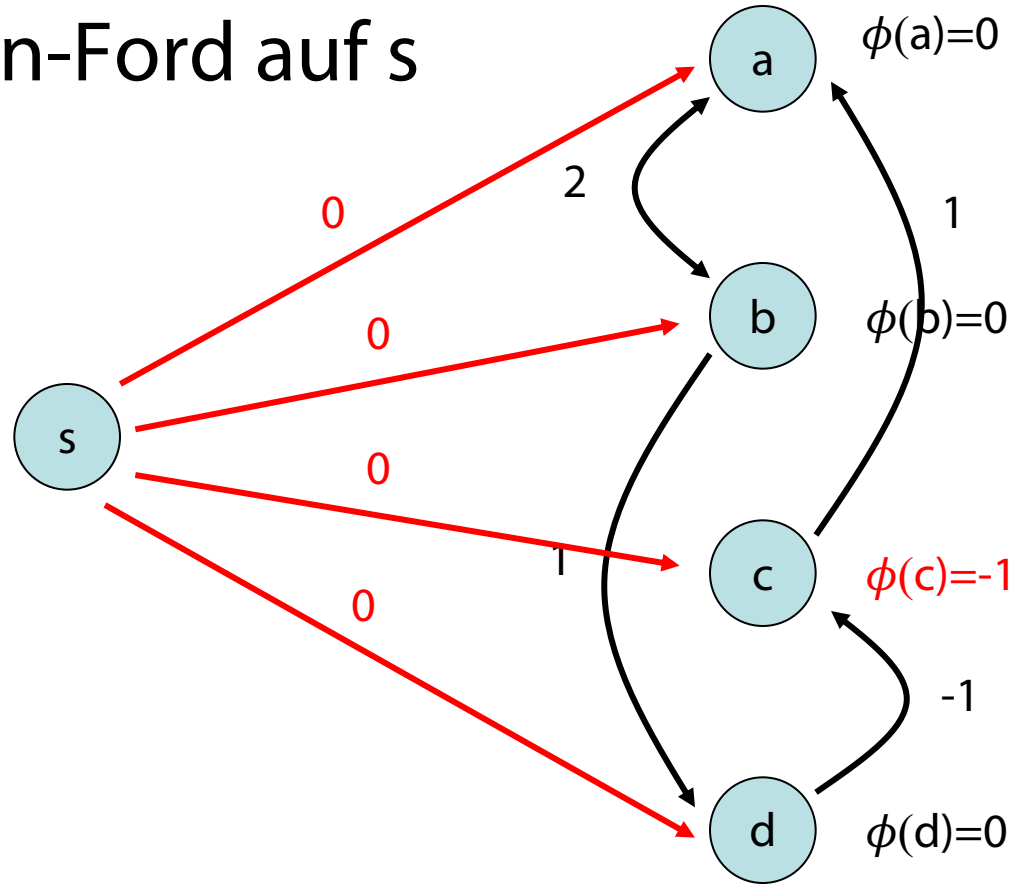
All Pairs Shortest Paths

Schritt 1: Künstliche Quelle



All Pairs Shortest Paths

Schritt 2: Bellman-Ford auf s

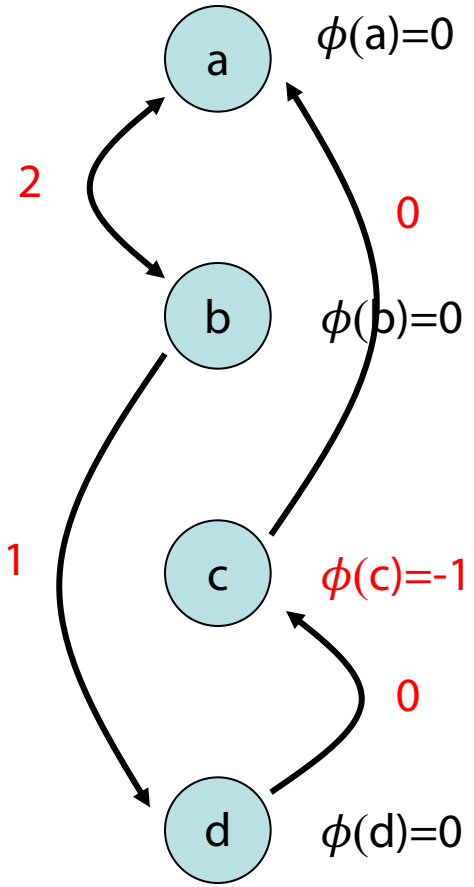


All Pairs Shortest Paths

Schritt 3: r(e)-Werte berechnen

Die reduzierten Kosten von e=(v,w) sind:

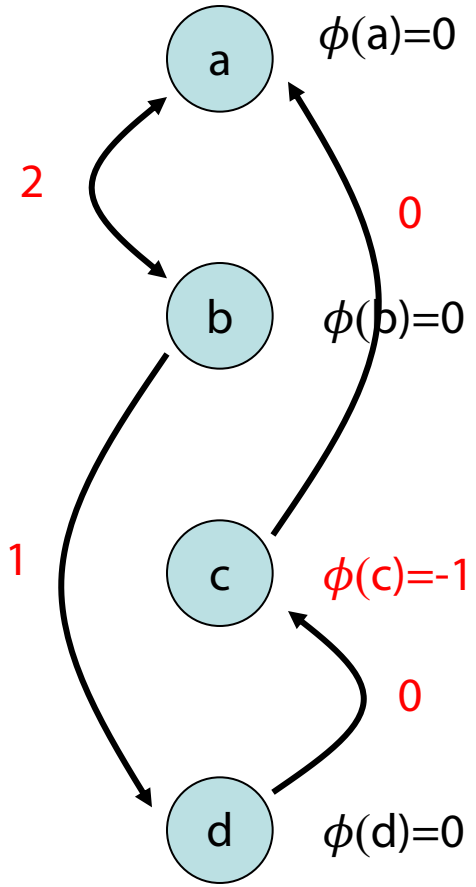
$$r(e) := \phi(v) + c(e) - \phi(w)$$



All Pairs Shortest Paths

Schritt 4: Berechne alle Distanzen $\bar{\mu}(v,w)$ via Dijkstra

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

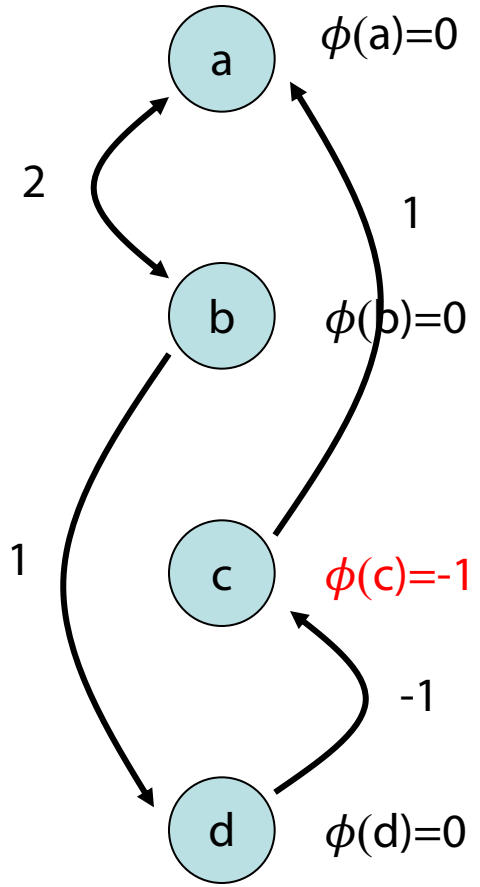


All Pairs Shortest Paths

Schritt 5: Berechne korrekte Distanzen durch die Formel

$\mu(v,w) = \bar{\mu}(v,w) + \phi(w) - \phi(v)$

μ	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

Laufzeit des APSP-Algorithmus (Johnson-Dijkstra):

$$O(n+ T_{\text{Bellman-Ford}}(n,m) + m + n \cdot T_{\text{Dijkstra}}(n,m) + n^2)$$
$$= O(n + n \cdot m + m + n(n \log n + m) + n^2)$$
$$= O(n \cdot m + n^2 \log n)$$

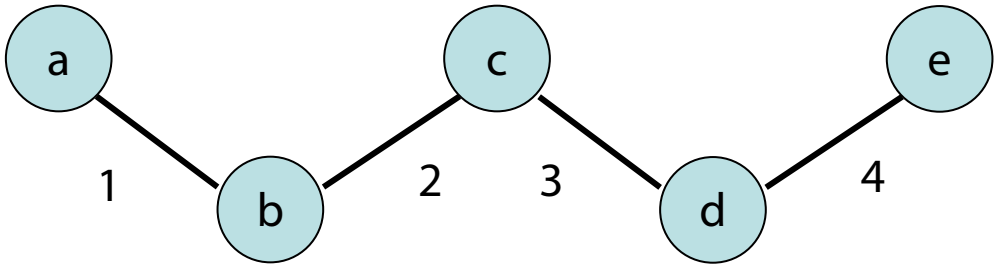
unter Verwendung von Fibonacci Heaps.

Da i. Allg. $m>n$, ist das sicher besser als

$n \times$ Bellman-Ford $\in O(n^2 \cdot m)$

Annahme: Ungerichteter Graph gegeben

Gegeben: Kantenkosten w[i, j]
Können wir APSP nicht besser hinkriegen?




```
(1) Für alle i,j  : d[i,j] = w[i,j]
(2) Für k = 1 bis n
(3)   Für alle Paare i,j
(4)     d[i,j] = min (d[i,j],d[i,k] + d[k,j])
```

Annahme: Ungerichteter Graph gegeben

```
(1) Für alle i,j  : d[i,j] = w[i,j]
(2) Für k = 1 bis n
(3)   Für alle Paare i,j
(4)     d[i,j] = min (d[i,j],d[i,k] + d[k,j])
```

Analyse: Zeit $O(n^3)$, Platz $O(n^2)$

- Wenn wir annehmen, dass $m>n$, ist das immer noch besser als $n \times$ Bellman-Ford $\in O(n^2 \cdot m)$
- Aber nicht besser als Johnson-Dijkstra: $O(n \cdot m + n^2 \log n)$

Robert W. Floyd: Algorithm 97 (SHORTEST PATH).
In: Communications of the ACM 5, 6, S. 345, **1962**
und schon früher wurden ähnliche Verfahren veröffentlicht

Zusammenfassung

- SSSP für Graphen mit neg. Kantenkosten
- All-Pairs-Shortest-Paths-Algorithmen