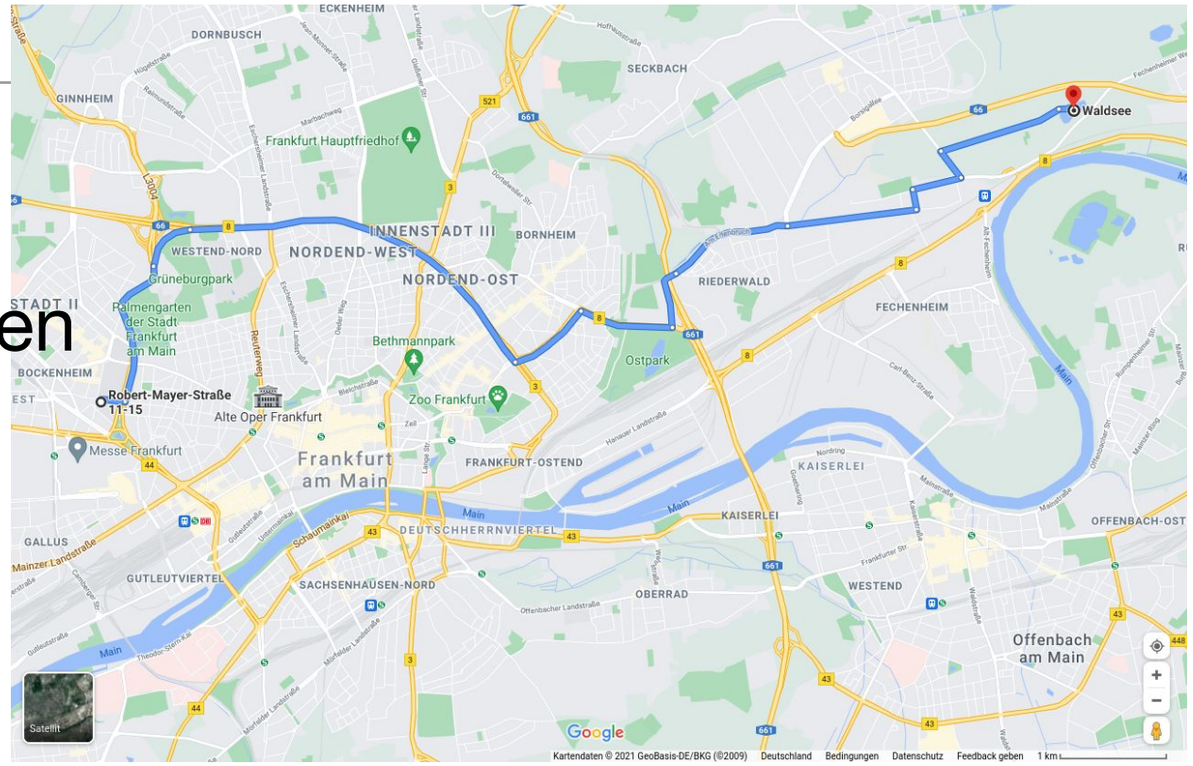
# Kürzeste Wege

- Kürzeste Wege
- Eigenschaften von kürzesten Wegen
- Dijkstras Algorithmus
- Kürzeste Wege in DAGs





Holger Dell

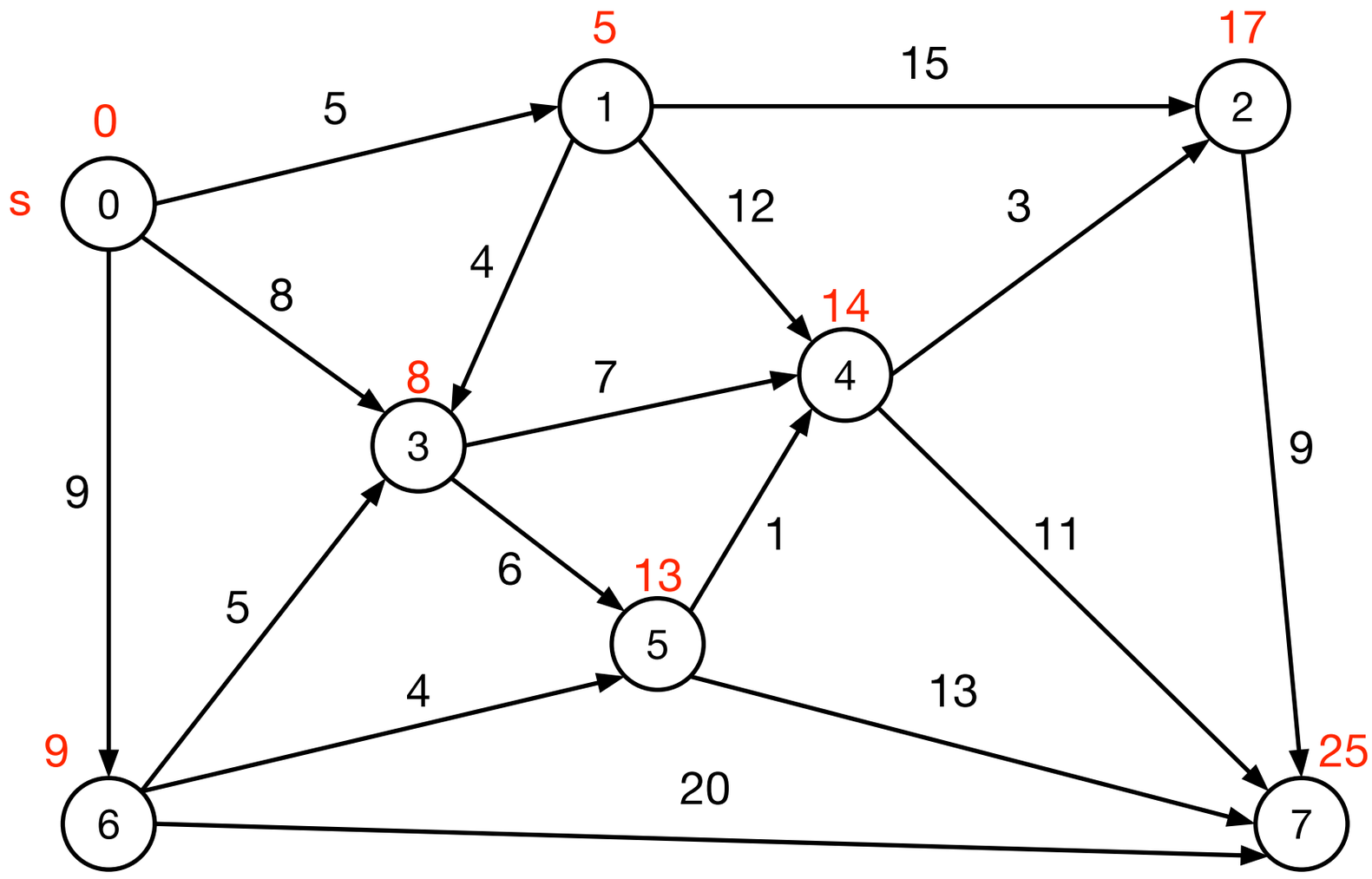
Folien adaptiert von Philip Bille

# Kürzeste Wege

- Kürzeste Wege
- Eigenschaften von kürzesten Wegen
- Dijkstras Algorithmus
- Kürzeste Wege in DAGs

# Kürzeste Wege

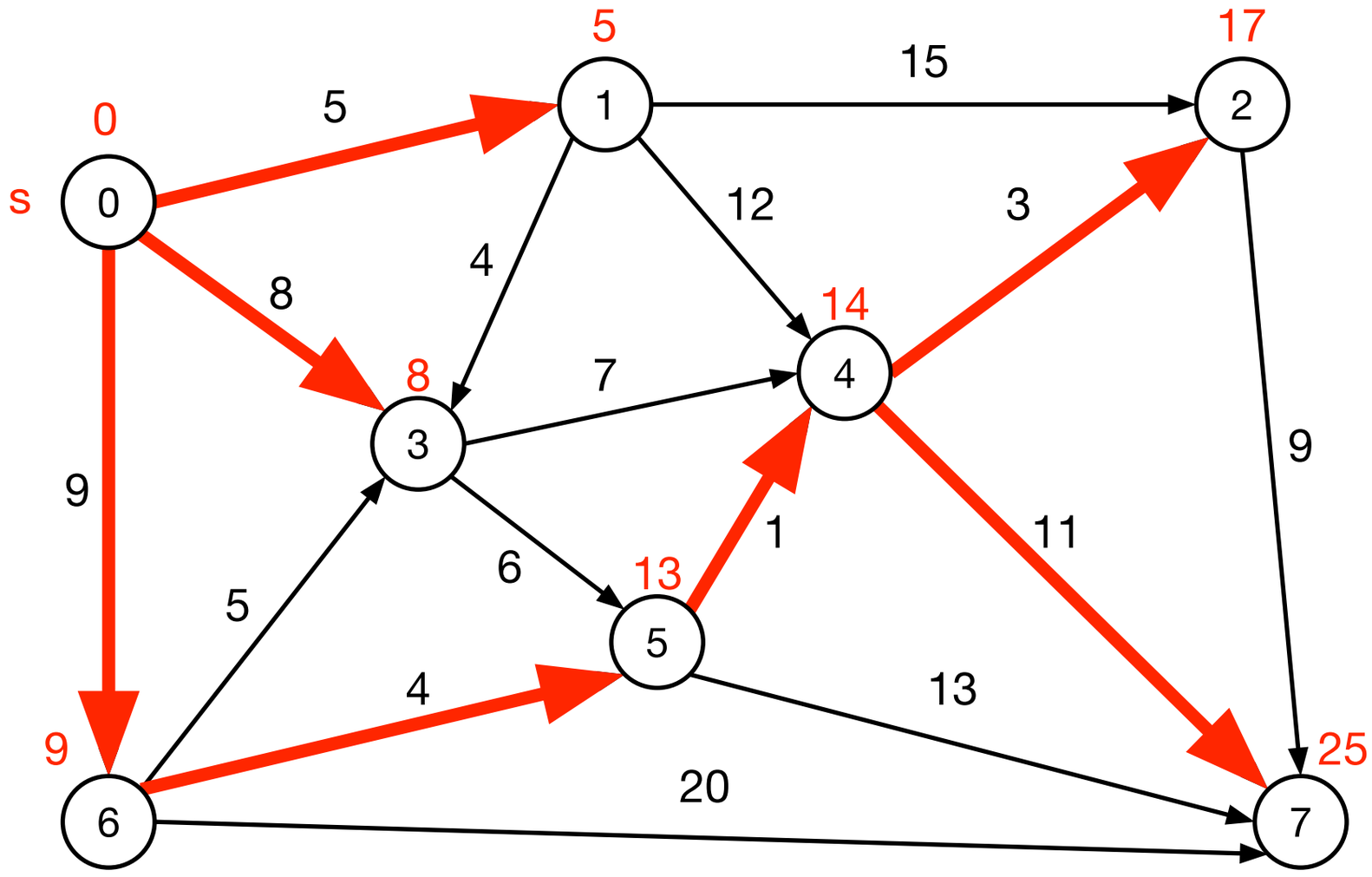
- Kürzeste Wege. Gegeben ein gerichteter, gewichteter Graph G und Knoten s, finde einen kürzesten Weg von s zu allen Knoten in G.



# Kürzeste Wege

- Kürzeste Wege. Gegeben ein gerichteter, gewichteter Graph G und Knoten s, finde einen kürzesten Weg von s zu allen Knoten in G.

- Baum kürzester Wege. Stellt kürzeste Wege in einem Baum gewurzelt an s dar.



# Anwendungen

- Routing, scheduling, pipelining, **...**

# Kürzeste Wege

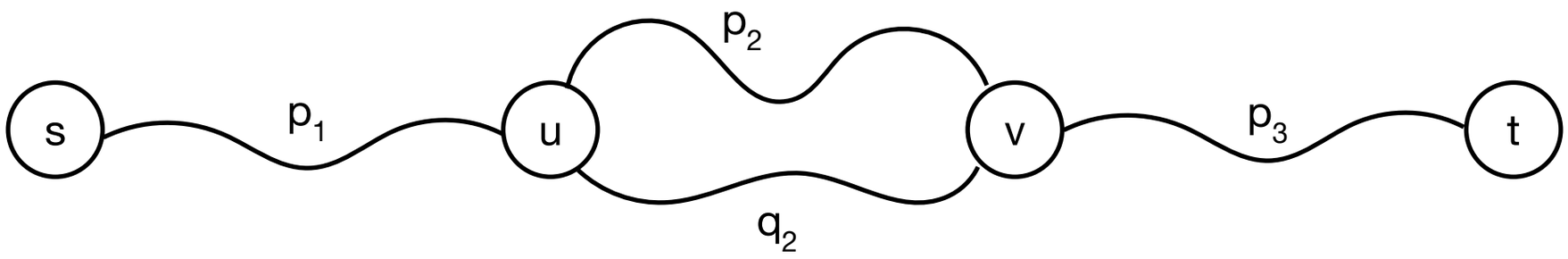
- Kürzeste Wege
- Eigenschaften von kürzesten Wegen
- Dijkstras Algorithmus
- Kürzeste Wege in DAGs

# Eigenschaften von kürzesten Wegen

- Vereinfachende Annahme:
  - Alle Knoten sind von s aus erreichbar.
- $\Longrightarrow$ ein (kürzester) Weg zu jedem Knoten existiert immer.

# Eigenschaften von kürzesten Wegen

- Teilweg-Eigenschaft. Jeder Teilweg eines kürzesten Wegs ist ein kürzester Weg.
- Beweis.
  - Betrachte kürzesten Weg von s nach t, zusammengesetzt aus $p_1$ $p_2$ $p_3$.




- Angenommen $q_2$ wäre kürzer als $p_2$.
- $\Longrightarrow$ Der Weg $p_1$ $q_2$ $p_3$ kürzer als p. □

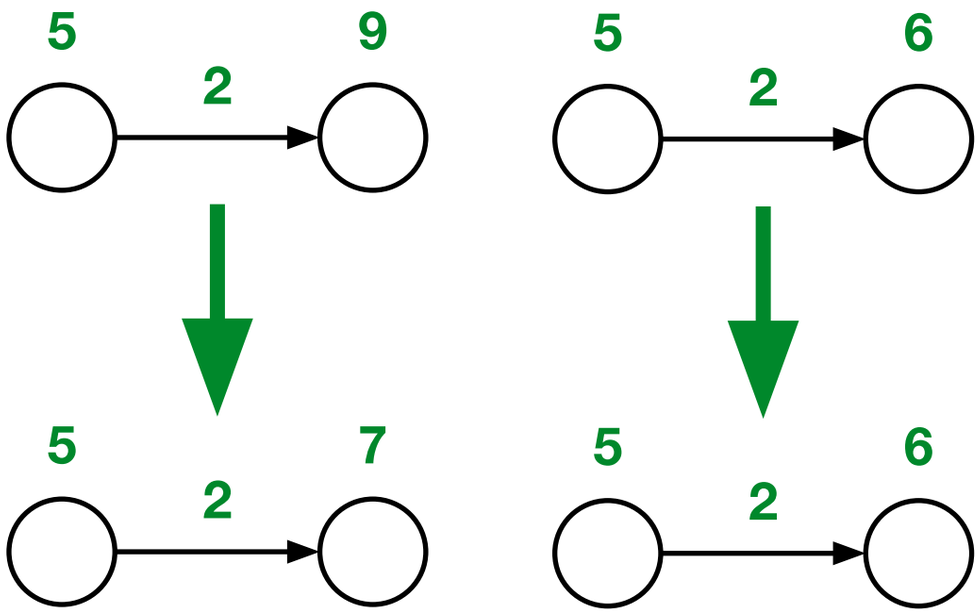
# Kürzeste Wege

- Kürzeste Wege
- Eigenschaften von kürzesten Wegen
- Dijkstras Algorithmus
- Kürzeste Wege in DAGs

# Dijkstras Algorithmus

- Ziel. Gegeben ein gerichteter, gewichteter Graph mit nicht-negativen Gewichten und einen Knoten s, berechne kürzeste Wege von s zu allen Knoten.

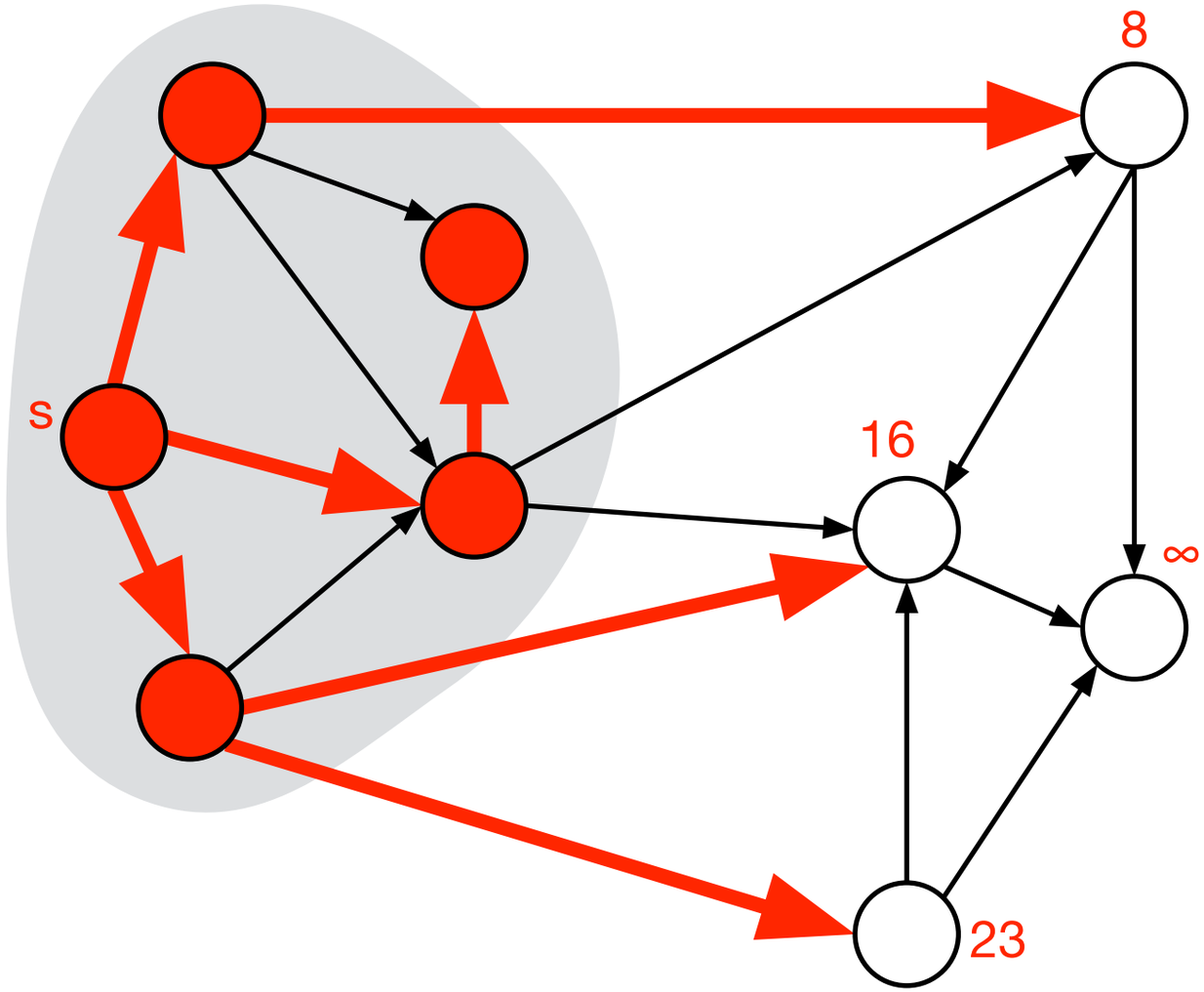
- Dijkstras Algorithmus.
  - Verwalte Abstandsschätzung v.d für jeden Knoten v = Länge des kürzesten bisher bekannten Wegs von s nach v.
  - Aktualisiere Abstandsschätzungen durch Relaxierung der Kanten.

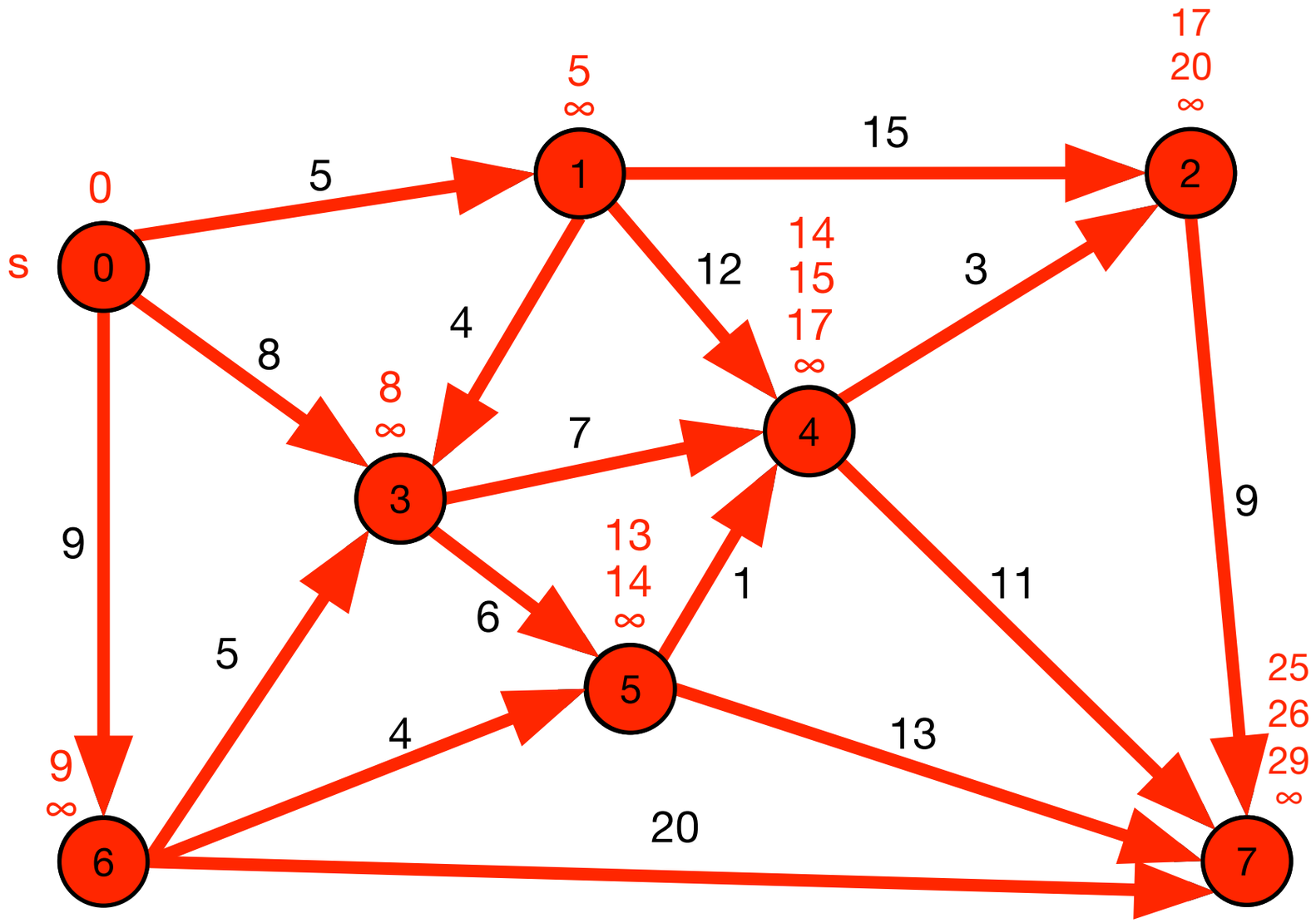


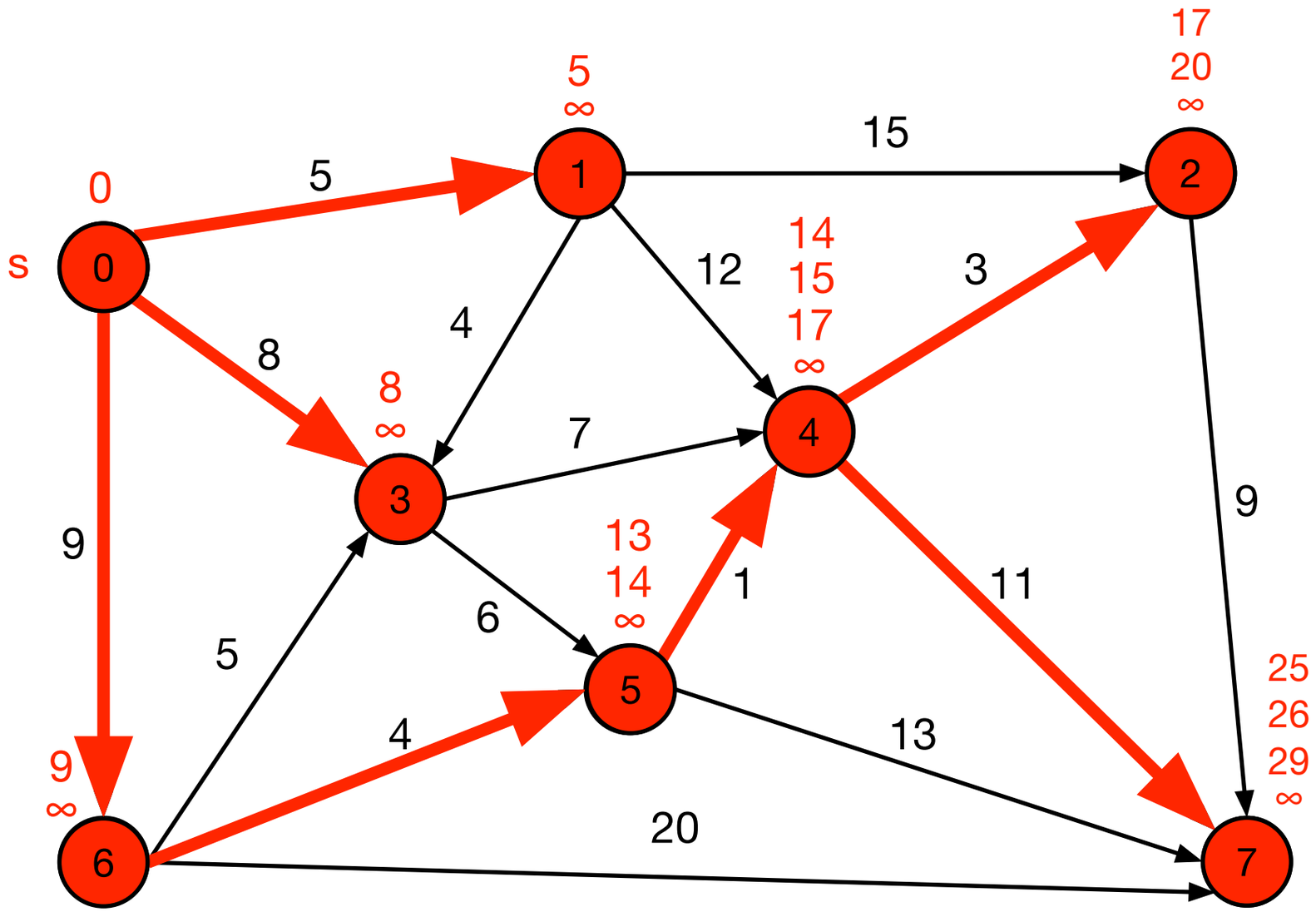

```
RELAX(u,v)
   if (v.d > u.d + w(u,v))
      v.d = u.d + w(u,v)
```

# Dijkstras Algorithmus

- Initialisiere s.d = 0 und v.d = ∞ für alle Knoten v ∈ V \ {s}.
- Wachse einen Baum T von s aus.
- In jedem Schritt, füge Knoten u ein mit kleinster Abstandsschätzung nach T.
- Relaxiere alle von u ausgehenden Kanten.

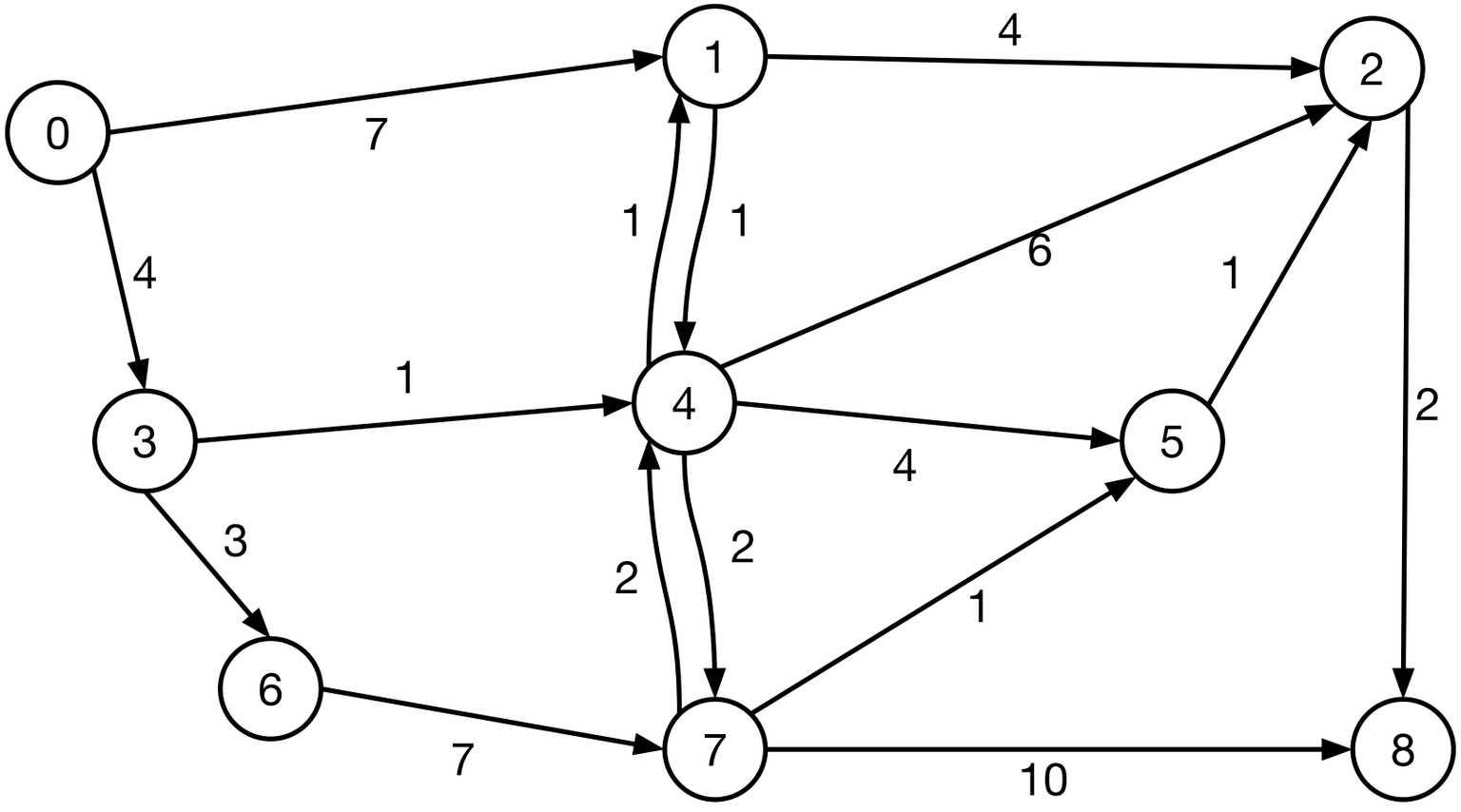



s
0
0
1
5
∞
2
17
20
∞
3
8
∞
4
14
15
17
∞
5
13
14
∞
6
9
∞
7
25
26
29
∞
5
8
4
12
15
3
9
7
9
6
5
1
11
4
13
20


s
0
0
1
5
∞
2
17
20
∞
3
8
∞
4
14
15
17
∞
5
13
14
∞
6
9
∞
7
25
26
29
∞
5
15
12
3
4
8
7
9
5
6
1
11
9
4
13
20

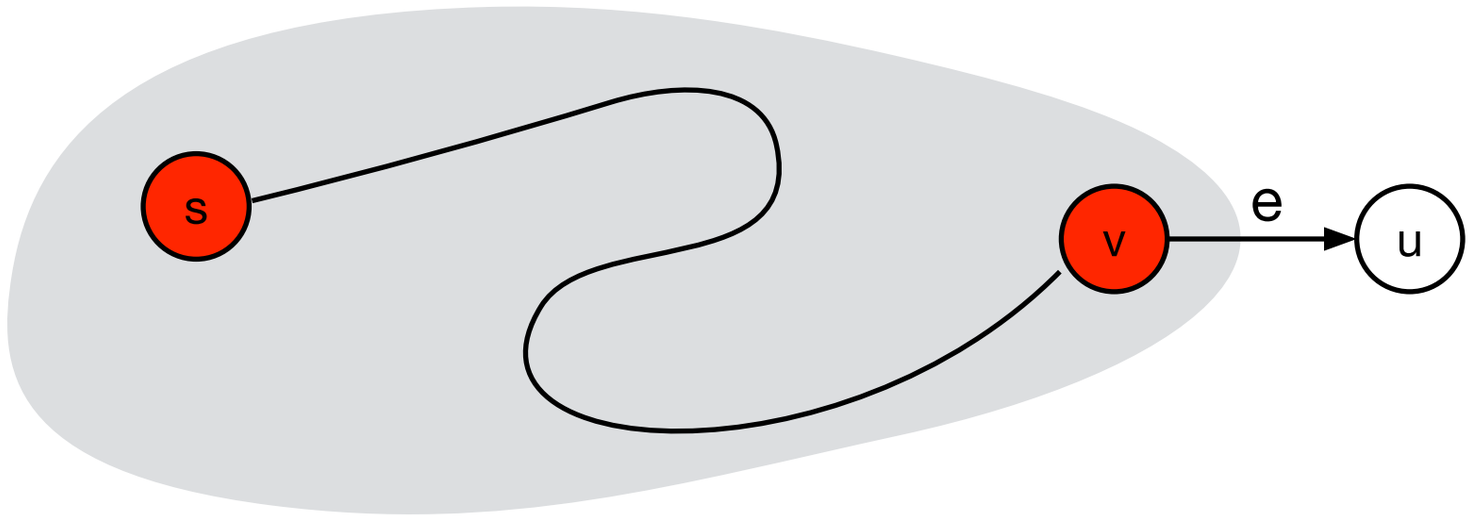
# Dijkstras Algorithmus

- Initialisiere s.d = 0 und v.d = ∞ für alle Knoten v ∈ V \ {s}.
- Wachse einen Baum T von s aus.
- In jedem Schritt, füge Knoten u ein mit kleinster Abstandsschätzung nach T.
- Relaxiere alle von u ausgehenden Kanten.
- Übung. Demonstriere die Ausführung von Dijkstras Algorithmus von Knoten 0.



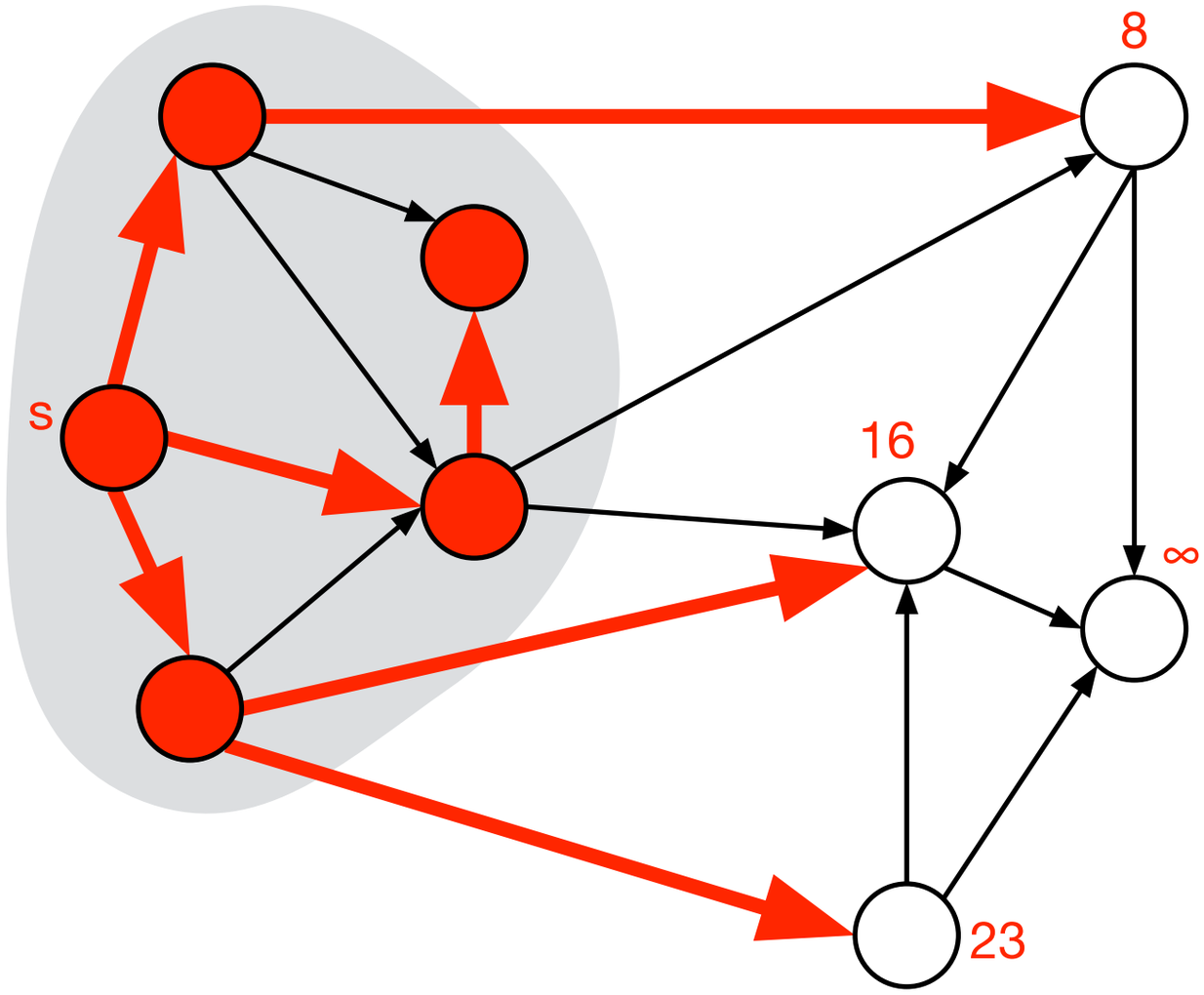
# Dijkstras Algorithmus

- Lemma. Dijkstras Algorithmus berechnet kürzeste Wege.
- Beweis.
  - Betrachte den aktuellen Baum T zu Beginn eines Schritts.
    Die Induktionsannahme ist, dass alle Distanzen in T korrekt sind.
  - Betrachte den an s nahesten Knoten u, der noch nicht in T ist.
  - Kürzester Weg von s nach u endet mit einer Kante e = (v,u).
  - v ist näher an s als u $\Longrightarrow$ v ist in T. (denn u ist nahster Knoten nicht in T)
  - $\Longrightarrow$ kürzester Weg von s nach v ist in T.
  - e wurde relaxiert $\Longrightarrow$ Abstandsschätzung von s nach u ist korrekt.
  - $\Longrightarrow$ Dijkstra fügt Kante e zu T hinzu
  - $\Longrightarrow$ Am Ende T = Baum kürzester Wege.



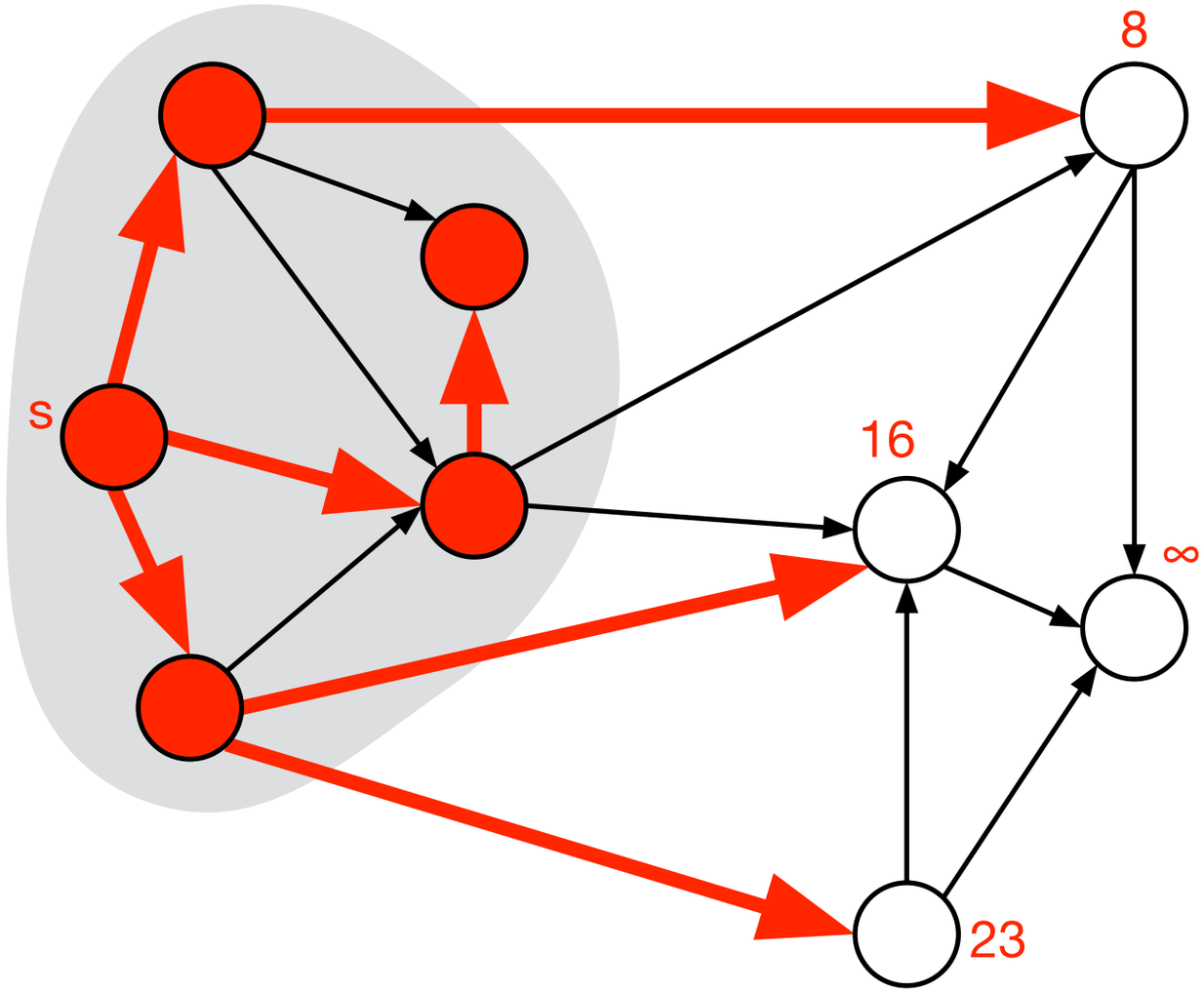
# Dijkstras Algorithmus

- Implementierung. Wie können wir Dijkstras Algorithmus implementieren?
- Schwierigkeit. Finde Knoten mit kleinster Abstandsschätzung.



# Dijkstras Algorithmus

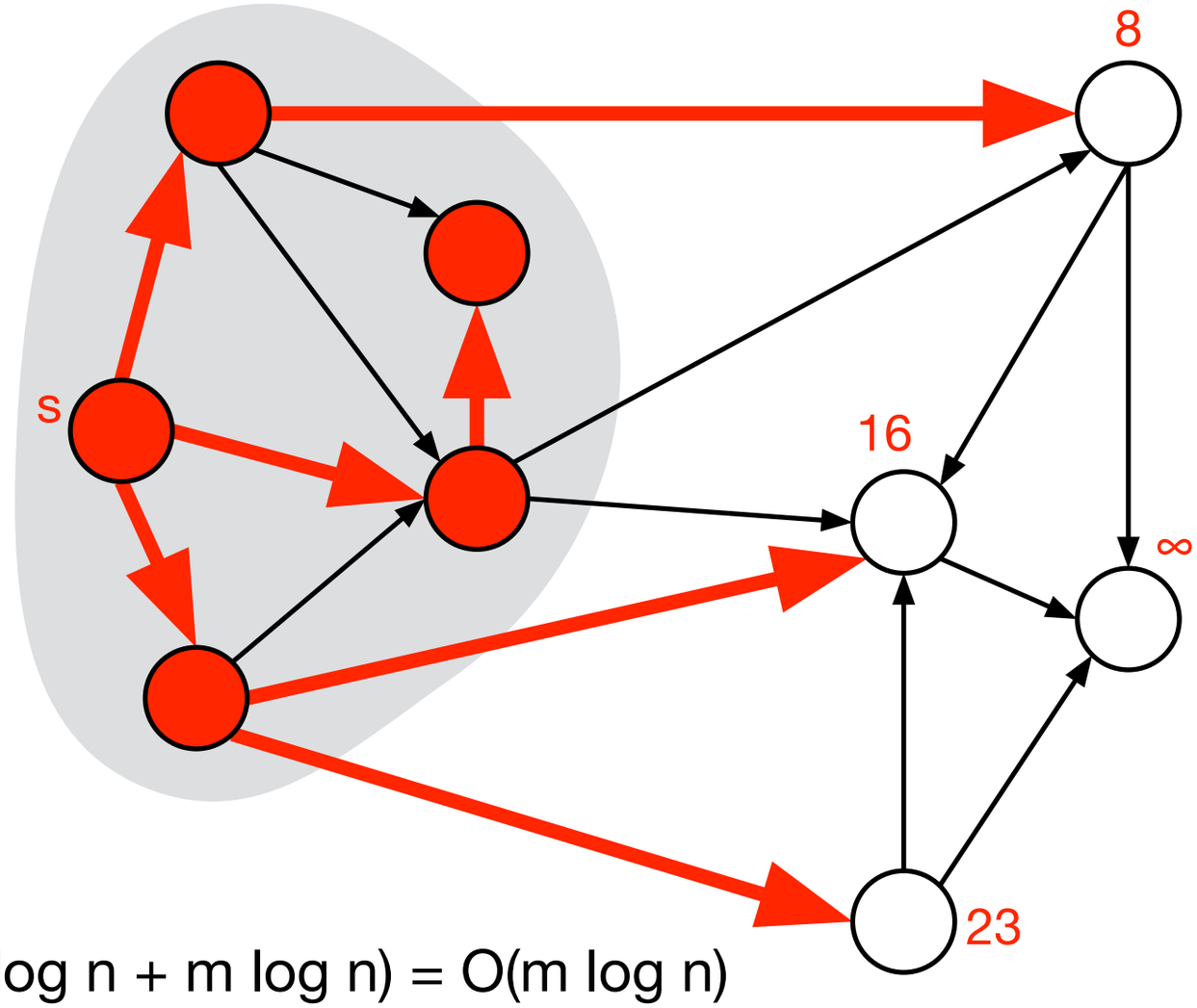
- Implementierung. Verwalte Knoten außerhalb von T in Prioritätswarteschlange.
  - Schlüssel von Knoten v = v.d.
  - In jedem Schritt:
    - Finde Knoten u mit kleinster Abstandsschätzung = EXTRACT-MIN
    - Relaxiere alle Kanten, zu denen u zeigt, mit DECREASE-KEY.



# Dijkstras Algorithmus

```
DIJKSTRA(G,s):
   für alle Knoten v∈V:
      v.d = ∞
      v.π = null
      INSERT(P,v)
   DECREASE-KEY(P,s,0)
   while (P ≠ ∅):
      u = EXTRACT-MIN(P)
      für alle Kanten (u,v):
         RELAX(u,v)
```

```
RELAX(u,v)
   if (v.d > u.d + w(u,v)):
      v.d = u.d + w(u,v)
      DECREASE-KEY(P,v,v.d)
      v.π = u
```




- Zeit.
  - n EXTRACT-MIN
  - n INSERT
  - < m DECREASE-KEY
- Gesamtzeit mit Min-Heap. O(n log n + n log n + m log n) = O(m log n)

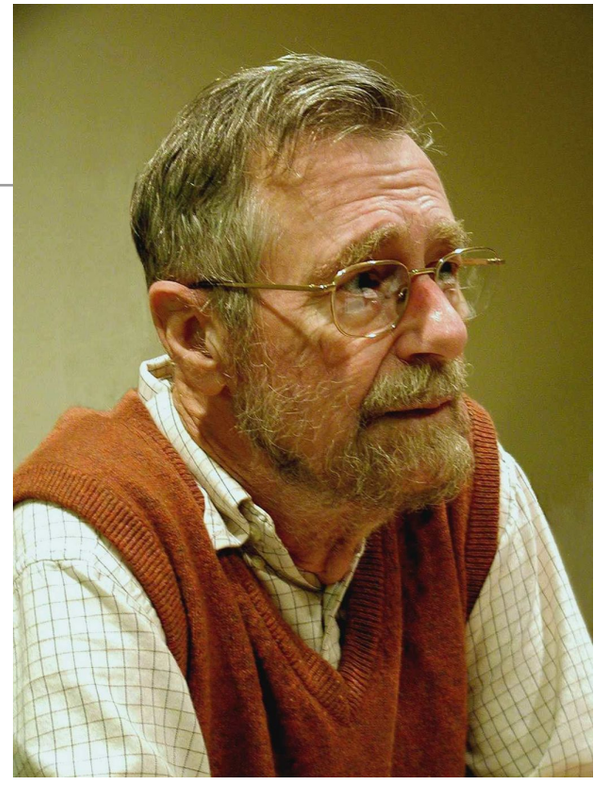
# Dijkstras Algorithmus

- Prioritätswarteschlange und Dijkstras Algorithmus. Komplexität von Dijkstras Algorithmus hängt von der Implementierung der Prioritätswarteschlange ab.
  - n INSERT
  - n EXTRACT-MIN
  - < m DECREASE-KEY

| Prioritäts-warteschlange | INSERT | EXTRACT-MIN | DECREASE-KEY | Total |
|---|---|---|---|---|
| Feld | O(1) | O(n) | O(1) | $O(n^2)$ |
| Binärer Heap | O(log n) | O(log n) | O(log n) | O(m log n) |
| Fibonacci Heap | $O(1)^\dagger$ | $O(\log n)^\dagger$ | $O(1)^\dagger$ | O(m + n log n) |

- Gier. Dijkstras Algorithmus ist ein gieriger Algorithmus.

# Edsger Wybe Dijkstra (1930-2002)



- Dijkstras Algorithmus.
  "*A note on two problems in connexion with graphs.*"
  Numerische Mathematik 1, 1959.
- Forschungsbeiträge.
  - Grundlagen der Programmierung,
  - Verteilte Systeme,
  - Programmverifikation,
  - usw.

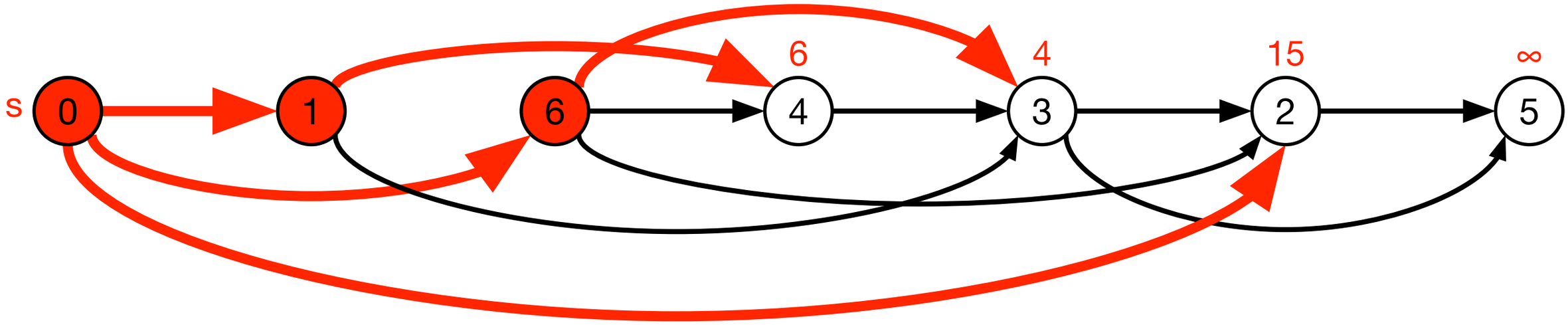
- Zitate. (Edsger Dijkstra Quotes)
  - *"Object-oriented programming is an exceptionally bad idea which could only have originated in California."*
  - *"The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence."*
  - *"The competent programmer is fully aware of the limited size of his own skull. He therefore approaches his task with full humility, and avoids clever tricks like the plague."*
  - *"Perfecting oneself is as much unlearning as it is learning."*

# Kürzeste Wege

- Kürzeste Wege
- Eigenschaften von kürzesten Wegen
- Dijkstras Algorithmus
- Kürzeste Wege in DAGs

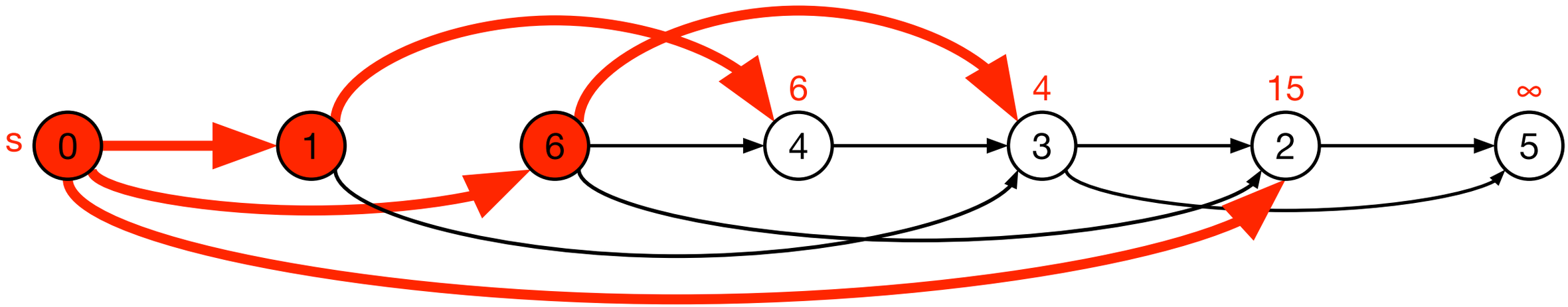
# Kürzeste Wege in DAGs

- Frage. Ist es komputational einfacher, kürzeste Wege in DAGs zu finden?
- Algorithmus für kürzeste Wege in DAGs.
  - Laufe die Knoten in topologischer Ordnung ab.
  - Für jeden Knoten v, relaxiere alle aus v ausgehenden Kanten.
- Funktioniert sogar, wenn die Kantengewichte negativ sind.



# Kürzeste Wege in DAGs

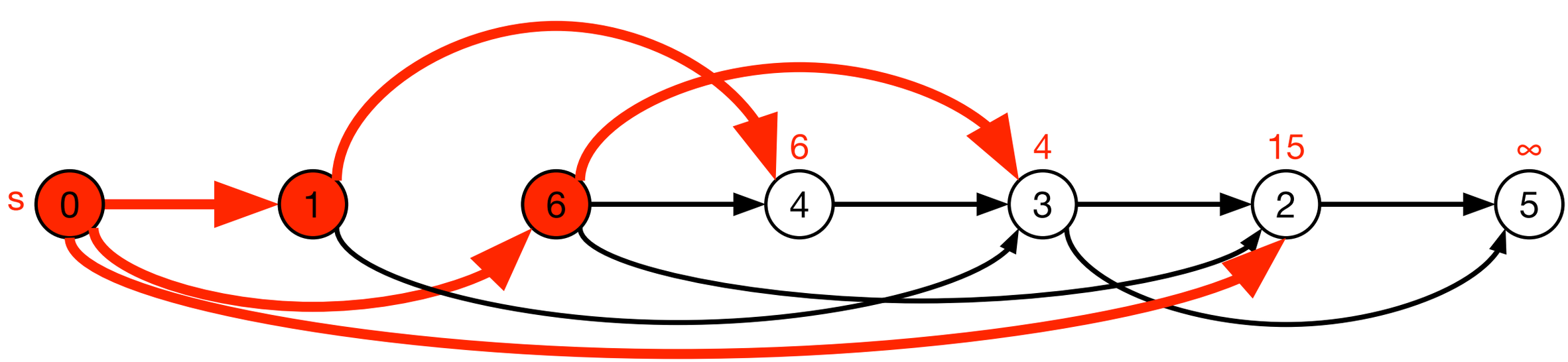
- Lemma. Algorithmus berechnet kürzeste Wege in DAGs.




- Beweis.
  - Betrachte den Beginn eines Schritts und nimm per Induktion an, dass der aktuelle Baum T die korrekten Abstände hat.
  - Betrachte den an s nahsten Knoten u, der noch nicht in T ist.
  - Jeder Weg nach u besteht aus Knoten in T & Kante e nach u.
  - Kante e wurde bereits relaxiert $\Longrightarrow$ Abstandsschätzung zu u ist korrekt.

# Kürzeste Wege in DAGs

- Implementierung.
  - Berechne eine topologische Ordnung der Knoten.
  - Relaxiere ausgehende Kanten von jedem Knoten.
- Gesamtzeit. O(m + n).



# Kürzeste Wege Varianten

- Knoten.
  - Einzelner Quellknoten.
  - Einzelner Quellknoten, einzelner Zielknoten.
  - Alle Paare. (*APSP = all pairs shortest paths)*
- Kantengewichte.
  - Nicht-negativ.
  - Beliebig.
  - Euklidischer Abstand.
- Kreise.
  - Keine Kreise.
  - Keine negativen Kreise.

# Kürzeste Wege

- Kürzeste Wege
- Eigenschaften von kürzesten Wegen
- Dijkstras Algorithmus
- Kürzeste Wege in DAGs