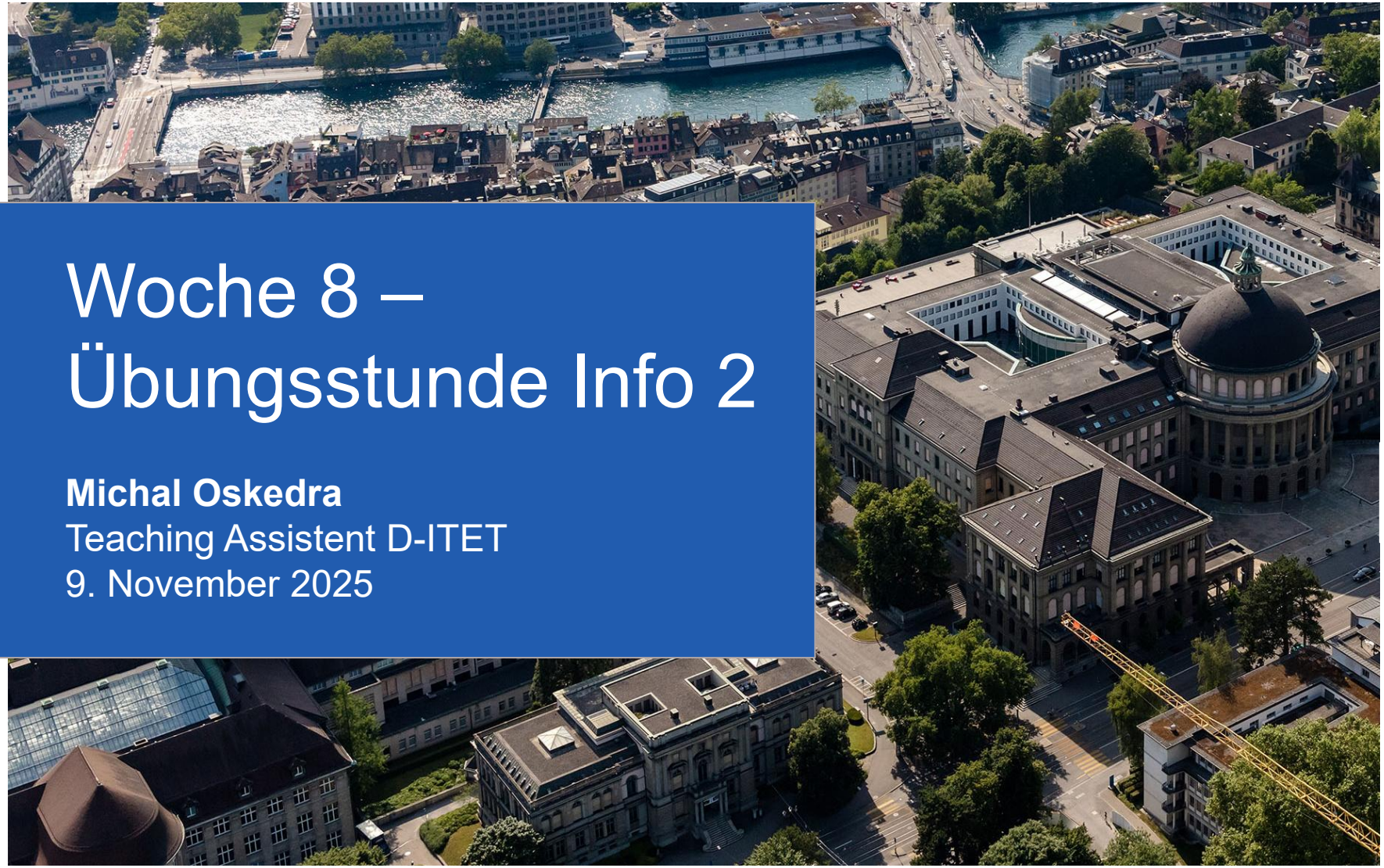
ETH zürich

Info2 - ML J 34.1 14-16

WhatsApp group



Woche 8 – Übungsstunde Info 2

Michal Oskedra
Teaching Assistent D-ITET
9. November 2025

Webseite:
moskedra-web.ethz.ch

01 – Theorie

01.1 Dijkstra

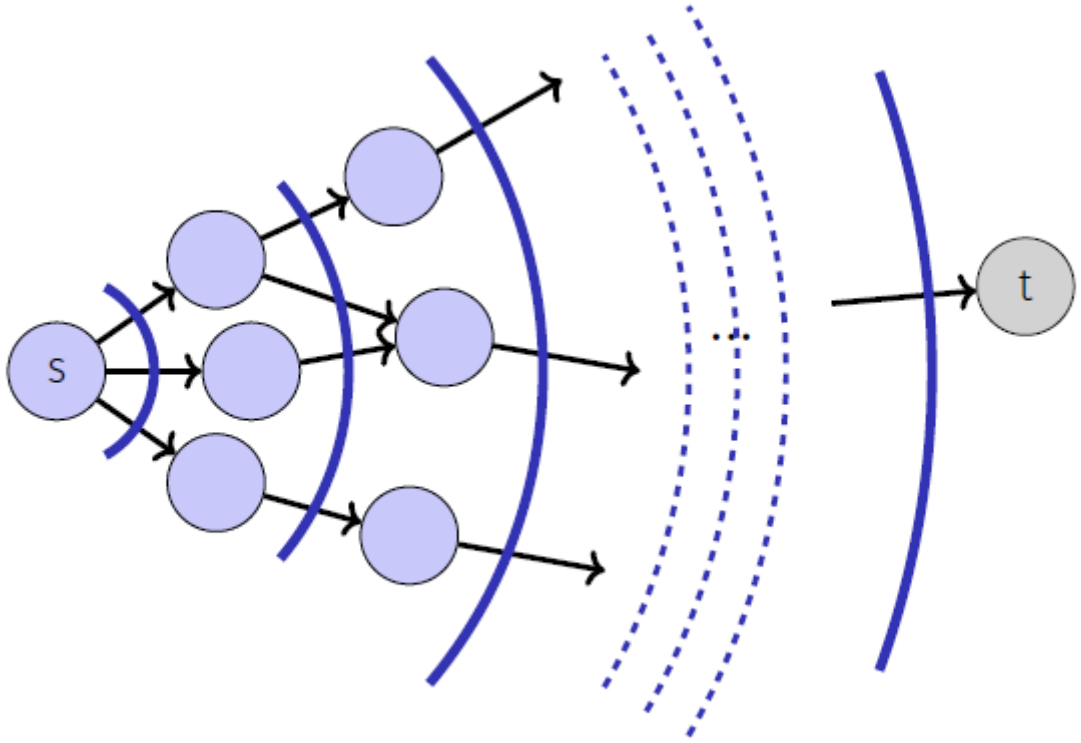
01.2 Bellman Ford

Wie komme ich an die ETH?



11:44	**Zürich, Albisriederplatz**	**Stop B**
	T 2	
	Direction Zürich Tiefenbrunnen, Bahnhof	
	NF	
11:50	Zürich, Stauffacher	
	Transfer	Show map ›
11:52	**Zürich, Stauffacher**	
	T 14	
	Direction Zürich, Seebach	
	NF	
11:55	Zürich, Löwenplatz	
	Transfer	Show map ›
11:57	**Zürich, Löwenplatz**	
	T 10	
	Direction Zürich Flughafen, Fracht	
	NF	
12:04	**Zürich, ETH/Universitätsspital**	

Konstante Kantenlänge

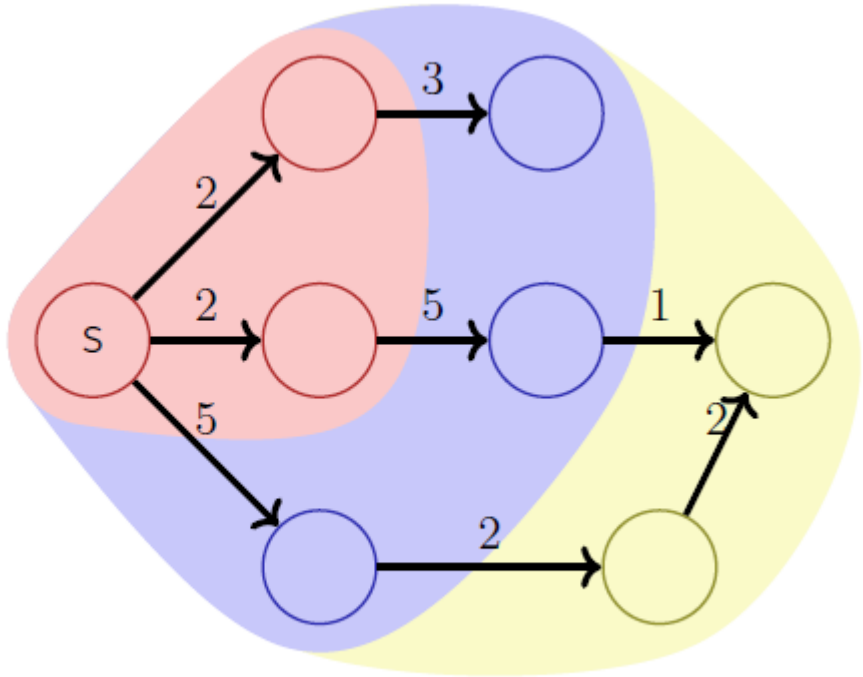


Lösung: Breitensuche!

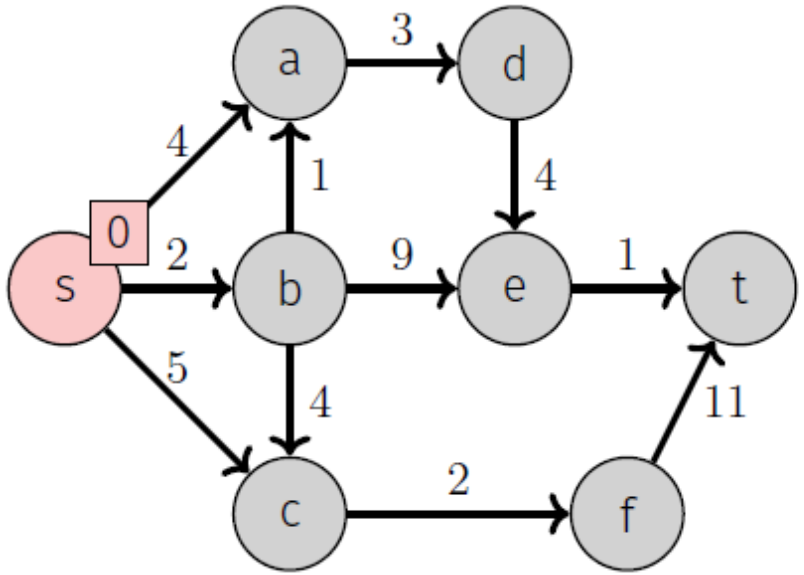
Dijkstra Algoritmus

Menge V aller Knoten wird unterteilt in

- die Menge S von Knoten, für die schon ein kürzester Weg von s bekannt ist
- die Menge $U = \bigcup_{v \in S} N^+(v) \setminus S$ von Knoten, für die kein kürzester Weg bekannt ist, die jedoch von S direkt erreichbar sind.
- die Menge $R = V \setminus (S \cup U)$ von Knoten, die noch nicht berücksichtigt wurden.



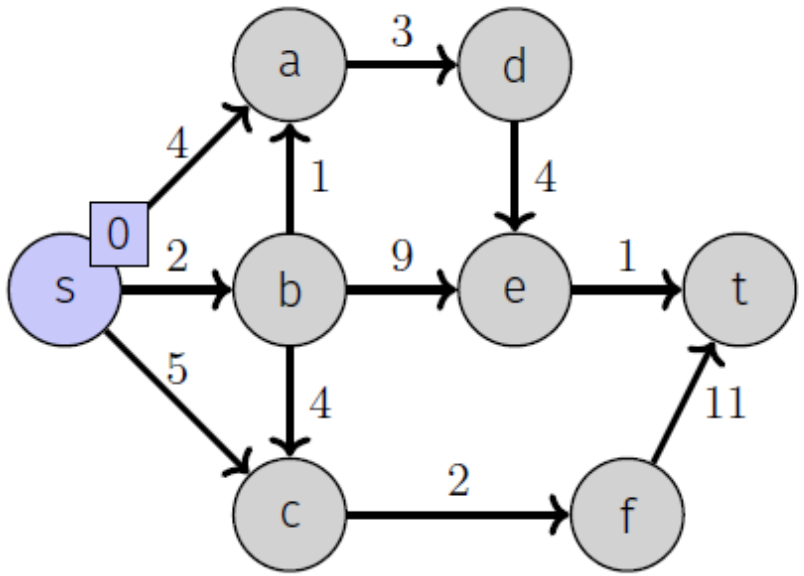
Dijkstra Algoritmus



1. Besuche den ausgesuchten Knoten → **S**
2. Aktualisiere die Wege zu den Nachfolgeknoten über den besuchten Knoten, wenn sie kürzer sind → **U**
3. Suche den Knoten mit kürzesten Weg aus.

$$\mathbf{S} = \{\}$$
$$\mathbf{U} = \{s\}$$
$$\mathbf{R} = \{a, b, c, d, e, f, t\}$$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

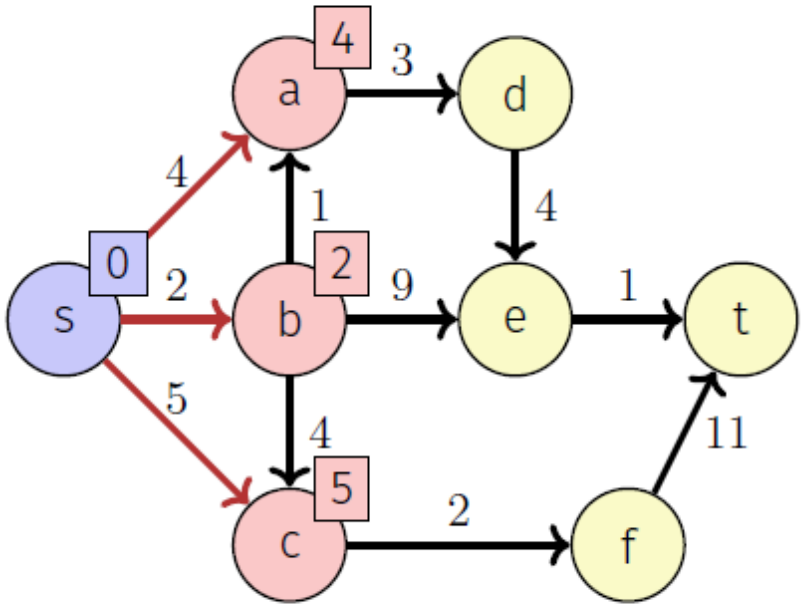
$s \rightsquigarrow s : 0$

$\mathbf{S} = \{s\}$

$\mathbf{U} = \{\}$

$\mathbf{R} = \{a, b, c, d, e, f, t\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s : 0$

Ausgehende Kanten:

$s \to a : 4$

$s \to b : 2$

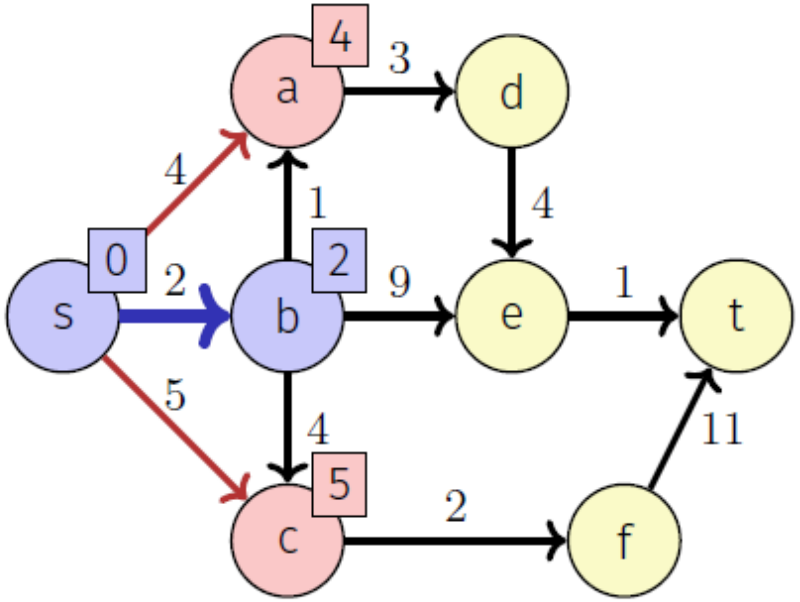
$s \to c : 5$

$\mathbf{S} = \{s\}$

$\mathbf{U} = \{a, b, c\}$

$\mathbf{R} = \{d, e, f, t\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s: 0$

$s \rightsquigarrow b: 2$

Ausgehende Kanten:

$s \to a: 4$

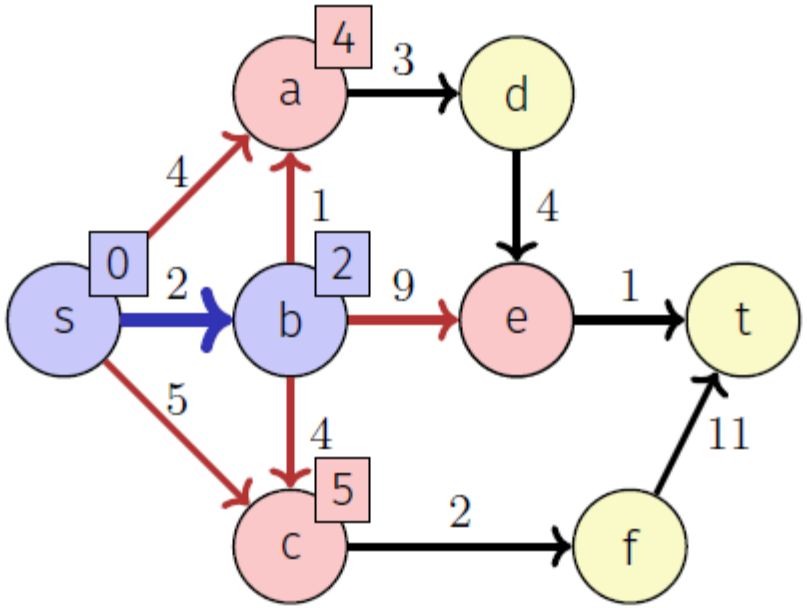
$s \to c: 5$

$\mathbf{S} = \{s, b\}$

$\mathbf{U} = \{a, c\}$

$\mathbf{R} = \{d, e, f, t\}$

Dijkstra Algoritmus



$\mathbf{S} = \{s, b\}$
$\mathbf{U} = \{a, c, e\}$
$\mathbf{R} = \{d, f, t\}$

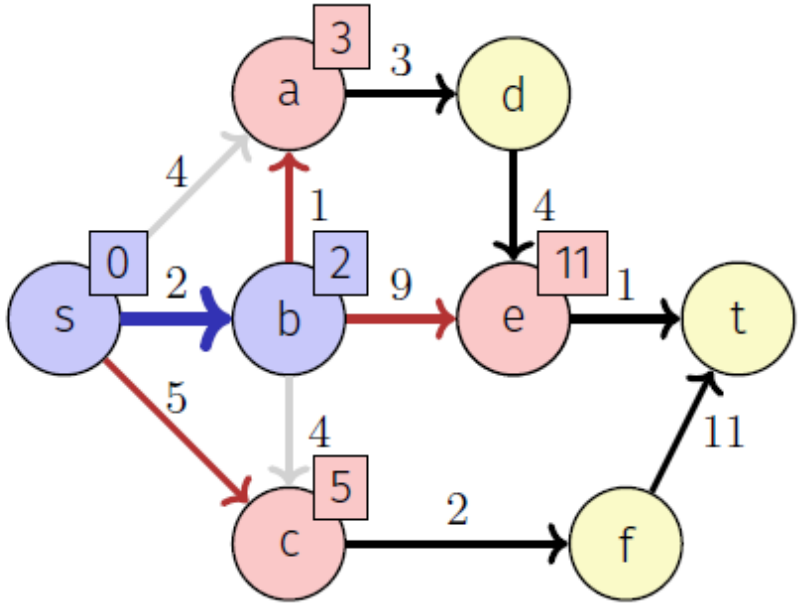
Bekannte kürzeste Wege von s:

$s \rightsquigarrow s$: 0
$s \rightsquigarrow b$: 2

Ausgehende Kanten:

$s \to a$: 4
$s \to c$: 5
$s \to b \to a$: 3
$s \to b \to e$: 11
$s \to b \to c$: 6

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s\colon 0$

$s \rightsquigarrow b\colon 2$

Ausgehende Kanten:

$s \to c\colon 5$

$s \to b \to a\colon 3$

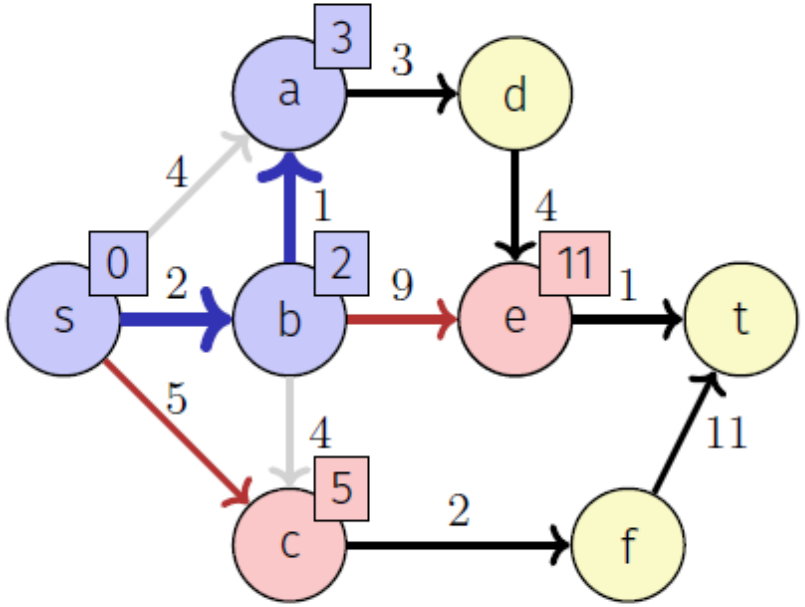
$s \to b \to e\colon 11$

$\mathbf{S} = \{s, b\}$

$\mathbf{U} = \{a, c, e\}$

$\mathbf{R} = \{d, f, t\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s\colon 0$

$s \rightsquigarrow b\colon 2$

$s \rightsquigarrow a\colon 3$

Ausgehende Kanten:

$s \to c\colon 5$

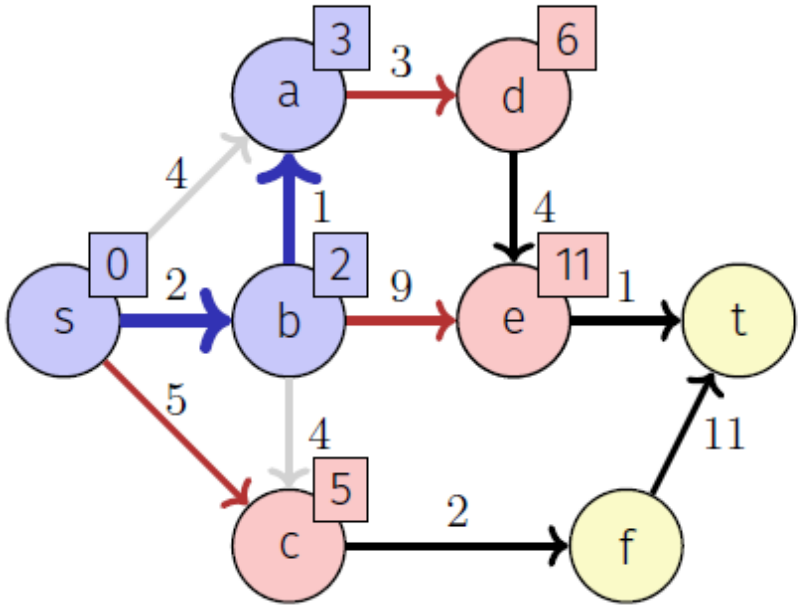
$s \to b \to e\colon 11$

$\mathbf{S} = \{s, b, a\}$

$\mathbf{U} = \{c, e\}$

$\mathbf{R} = \{d, f, t\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s\colon 0$

$s \rightsquigarrow b\colon 2$

$s \rightsquigarrow a\colon 3$

Ausgehende Kanten:

$s \to c\colon 5$

$s \to b \to a \to d\colon 6$

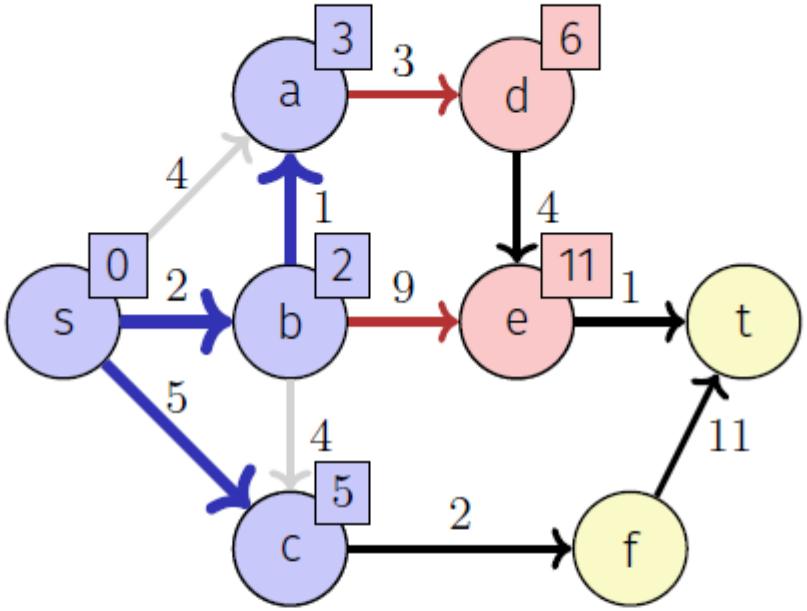
$s \to b \to e\colon 11$

$\mathbf{S} = \{s, b, a\}$

$\mathbf{U} = \{c, e, d\}$

$\mathbf{R} = \{f, t\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

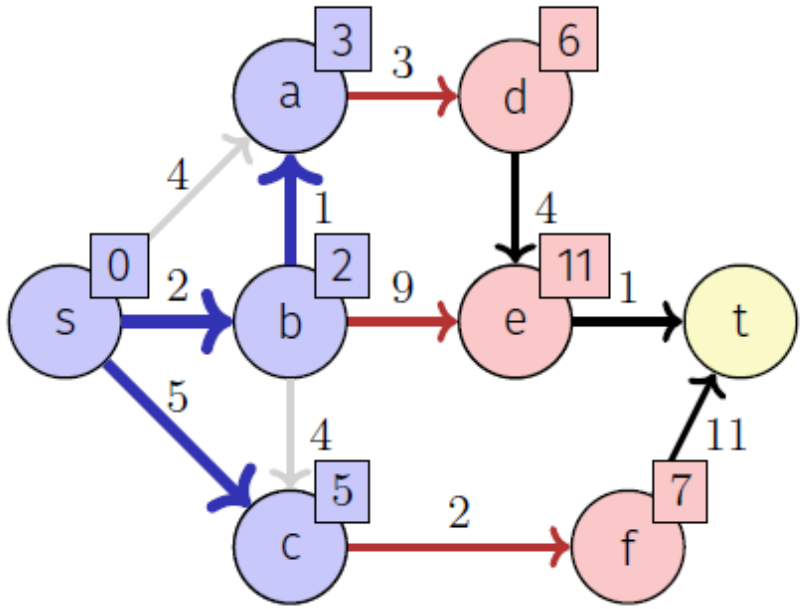
$s \rightsquigarrow s\colon 0$
$s \rightsquigarrow b\colon 2$
$s \rightsquigarrow a\colon 3$
$s \rightsquigarrow c\colon 5$

Ausgehende Kanten:

$s \to b \to a \to d\colon 6$
$s \to b \to e\colon 11$

$\mathbf{S} = \{s, b, a, c\}$
$\mathbf{U} = \{e, d\}$
$\mathbf{R} = \{f, t\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s: 0$

$s \rightsquigarrow b: 2$

$s \rightsquigarrow a: 3$

$s \rightsquigarrow c: 5$

Ausgehende Kanten:

$s \to b \to a \to d: 6$

$s \to b \to e: 11$

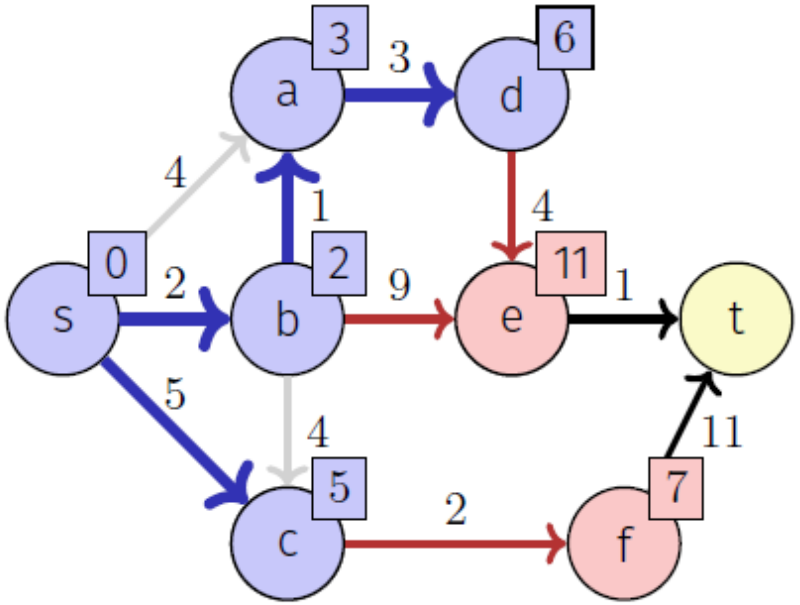
$s \to c \to f: 7$

$\mathbf{S} = \{s, b, a, c\}$

$\mathbf{U} = \{e, d, f\}$

$\mathbf{R} = \{t\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s: 0$ $\quad$ $s \rightsquigarrow d: 6$

$s \rightsquigarrow b: 2$

$s \rightsquigarrow a: 3$

$s \rightsquigarrow c: 5$

Ausgehende Kanten:

$s \to b \to a \to d \to e: 10$

$s \to b \to e: 11$

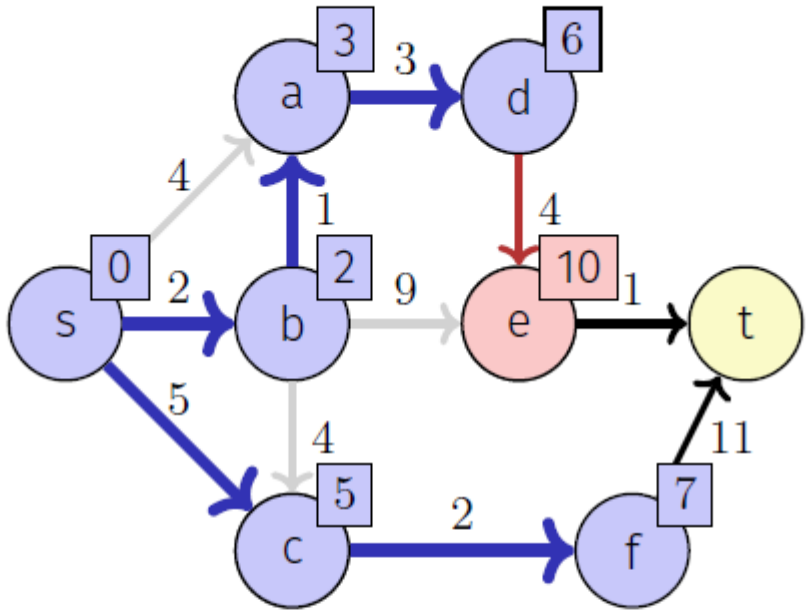
$s \to c \to f: 7$

$\mathbf{S} = \{s, b, a, c, d\}$

$\mathbf{U} = \{e, f\}$

$\mathbf{R} = \{t\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s: 0$ $\quad$ $s \rightsquigarrow d: 6$

$s \rightsquigarrow b: 2$ $\quad$ $s \rightsquigarrow f: 7$

$s \rightsquigarrow a: 3$

$s \rightsquigarrow c: 5$

Ausgehende Kanten:

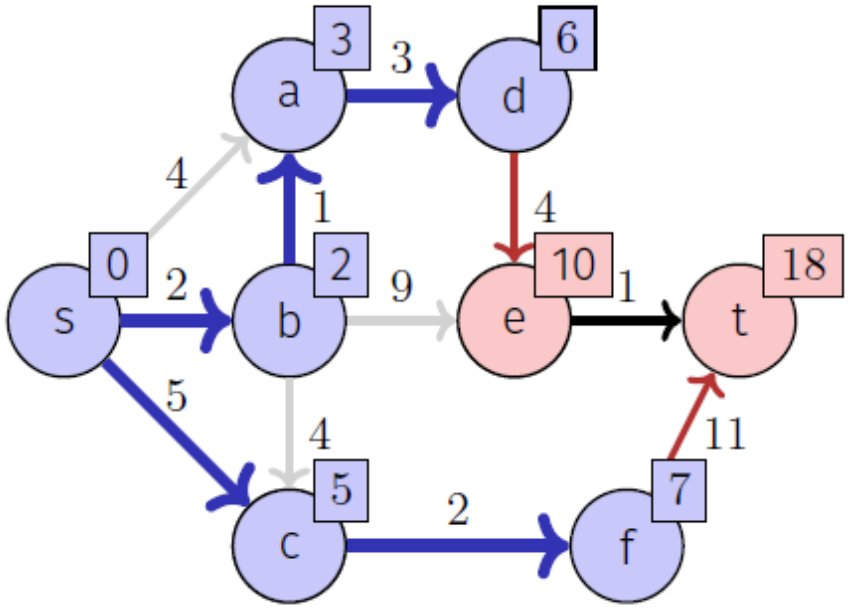
$s \to b \to a \to d \to e: 10$

$\mathbf{S} = \{s, b, a, c, d, f\}$

$\mathbf{U} = \{e\}$

$\mathbf{R} = \{t\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

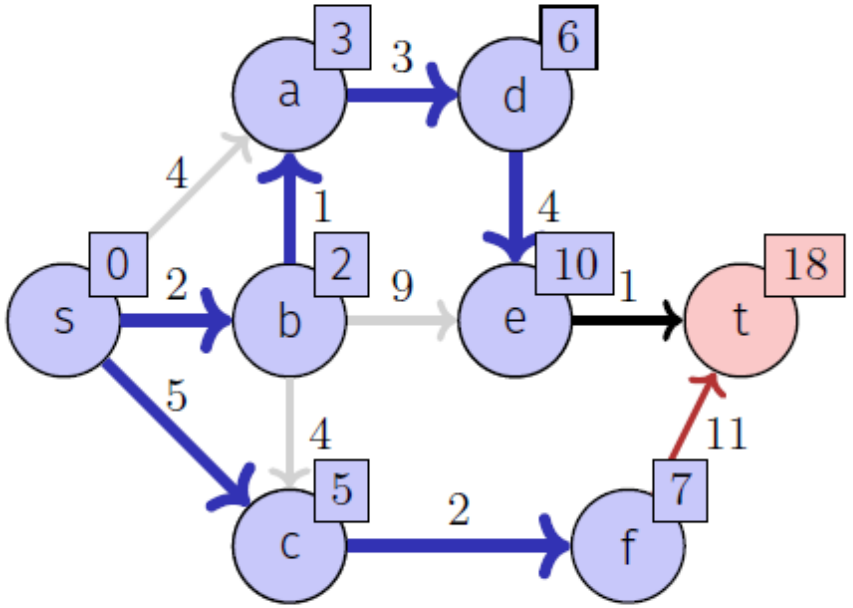
$s \rightsquigarrow s: 0$
$s \rightsquigarrow b: 2$
$s \rightsquigarrow a: 3$
$s \rightsquigarrow c: 5$
$s \rightsquigarrow d: 6$
$s \rightsquigarrow f: 7$

Ausgehende Kanten:

$s \to b \to a \to d \to e: 10$
$s \to c \to f \to t: 18$

$\mathbf{S} = \{s, b, a, c, d, f\}$
$\mathbf{U} = \{e, t\}$
$\mathbf{R} = \{\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s: 0$ $\quad$ $s \rightsquigarrow d: 6$

$s \rightsquigarrow b: 2$ $\quad$ $s \rightsquigarrow f: 7$

$s \rightsquigarrow a: 3$ $\quad$ $s \rightsquigarrow e: 10$

$s \rightsquigarrow c: 5$

Ausgehende Kanten:

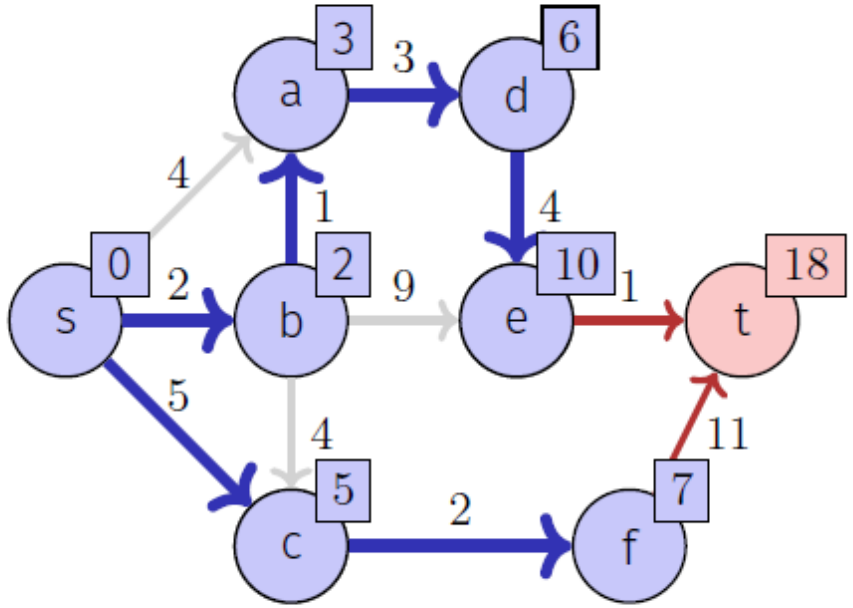
$s \to c \to f \to t: 18$

$\mathbf{S} = \{s, b, a, c, d, f, e\}$

$\mathbf{U} = \{t\}$

$\mathbf{R} = \{\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s: 0$ $\quad$ $s \rightsquigarrow d: 6$

$s \rightsquigarrow b: 2$ $\quad$ $s \rightsquigarrow f: 7$

$s \rightsquigarrow a: 3$ $\quad$ $s \rightsquigarrow e: 10$

$s \rightsquigarrow c: 5$

Ausgehende Kanten:

$s \to b \to a \to d \to e \to t: 11$

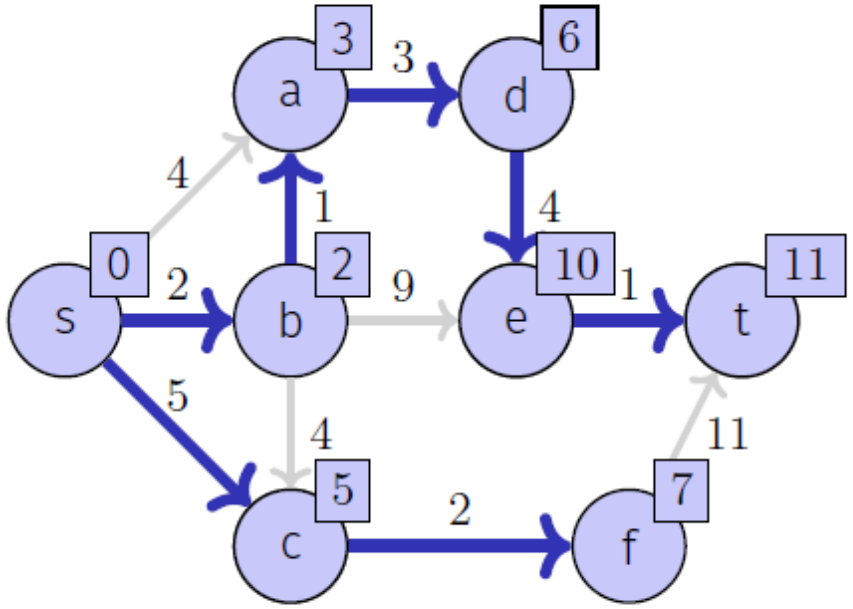
$s \to c \to f \to t: 18$

$\mathbf{S} = \{s, b, a, c, d, f, e\}$

$\mathbf{U} = \{t\}$

$\mathbf{R} = \{\}$

Dijkstra Algoritmus



Bekannte kürzeste Wege von s:

$s \rightsquigarrow s: 0$ $\quad$ $s \rightsquigarrow d: 6$

$s \rightsquigarrow b: 2$ $\quad$ $s \rightsquigarrow f: 7$

$s \rightsquigarrow a: 3$ $\quad$ $s \rightsquigarrow e: 10$

$s \rightsquigarrow c: 5$ $\quad$ $s \rightsquigarrow t: 11$

Ausgehende Kanten:

$\mathbf{S} = \{s, b, a, c, d, f, e, t\}$

$\mathbf{U} = \{\}$

$\mathbf{R} = \{\}$

Dijkstra Algoritmus

- Gewicht des kürzesten bisher gefundenen Pfades

$$d_s : V \to \mathbb{R}$$

- Vorgänger eines Knotens

$$\pi_s : V \to V$$

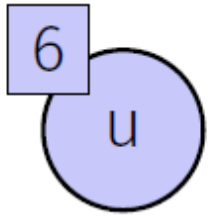
Dijkstra Code

Input: Positiv gewichteter Graph $G = (V, E, c)$,
Startpunkt $s \in V$

Output: Länge d_s der kürzesten Pfade von s und
Vorgänger π_s für jeden Knoten

foreach $u \in V$ **do**
  $d_s[u] \leftarrow \infty$; $\pi_s[u] \leftarrow$ **null**

$d_s[s] \leftarrow 0$; $U \leftarrow \{s\}$
while $U \neq \emptyset$ **do**
  $u \leftarrow \arg\min_{u \in U} d_s[u]$; $U \leftarrow U \setminus \{u\}$



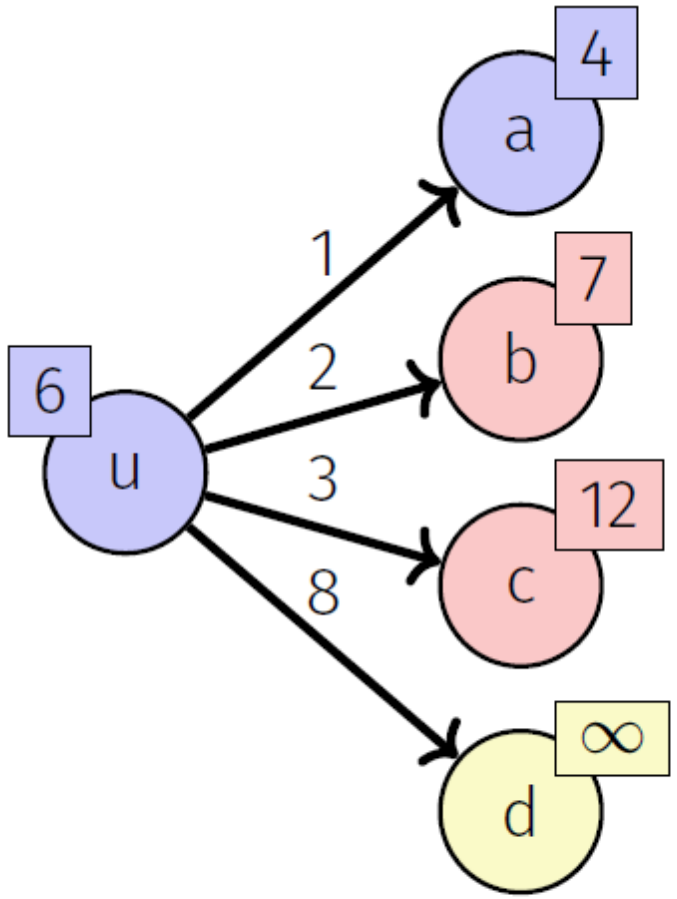
Dijkstra Code

Input: Positiv gewichteter Graph $G = (V, E, c)$,
Startpunkt $s \in V$

Output: Länge d_s der kürzesten Pfade von s und
Vorgänger π_s für jeden Knoten

foreach $u \in V$ **do**
  $d_s[u] \leftarrow \infty$; $\pi_s[u] \leftarrow$ **null**

$d_s[s] \leftarrow 0$; $U \leftarrow \{s\}$
while $U \neq \emptyset$ **do**
  $u \leftarrow \arg\min_{u \in U} d_s[u]$; $U \leftarrow U \setminus \{u\}$
  foreach $v \in N^+(u)$ **do**
    if $d_s[u] + c(u, v) < d_s[v]$ **then**
      $d_s[v] \leftarrow d_s[u] + c(u, v)$
      $\pi_s[v] \leftarrow u$
      $U \leftarrow U \cup \{v\}$



Nachfolgerknoten!

Dijkstra Code

Input: Positiv gewichteter Graph $G = (V, E, c)$,
Startpunkt $s \in V$

Output: Länge d_s der kürzesten Pfade von s und
Vorgänger π_s für jeden Knoten

foreach $u \in V$ **do**
  $d_s[u] \leftarrow \infty$; $\pi_s[u] \leftarrow$ **null**

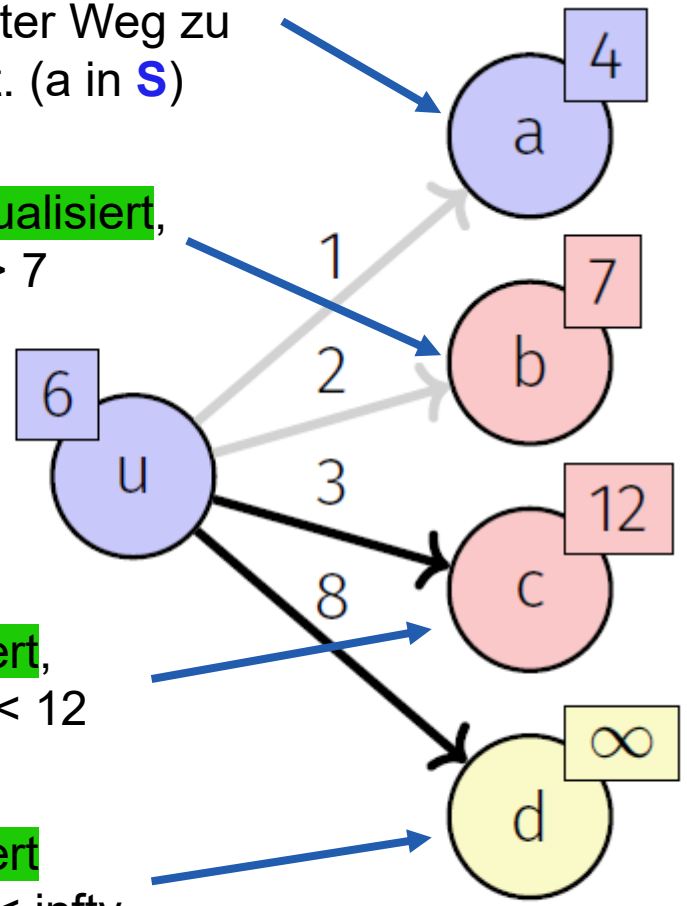
$d_s[s] \leftarrow 0$; $U \leftarrow \{s\}$
while $U \neq \emptyset$ **do**
  $u \leftarrow \arg\min_{u \in U} d_s[u]$; $U \leftarrow U \setminus \{u\}$
  foreach $v \in N^+(u)$ **do**
    if $d_s[u] + c(u, v) < d_s[v]$ **then**
      $d_s[v] \leftarrow d_s[u] + c(u, v)$
      $\pi_s[v] \leftarrow u$
      $U \leftarrow U \cup \{v\}$

Nicht aktualisiert,
da kürzester Weg zu
a bekannt. (a in **S**)

Nicht aktualisiert,
da 6 + 2 > 7

Aktualisiert,
da 6 + 3 < 12

Aktualisiert
da 6 + 8 < infty



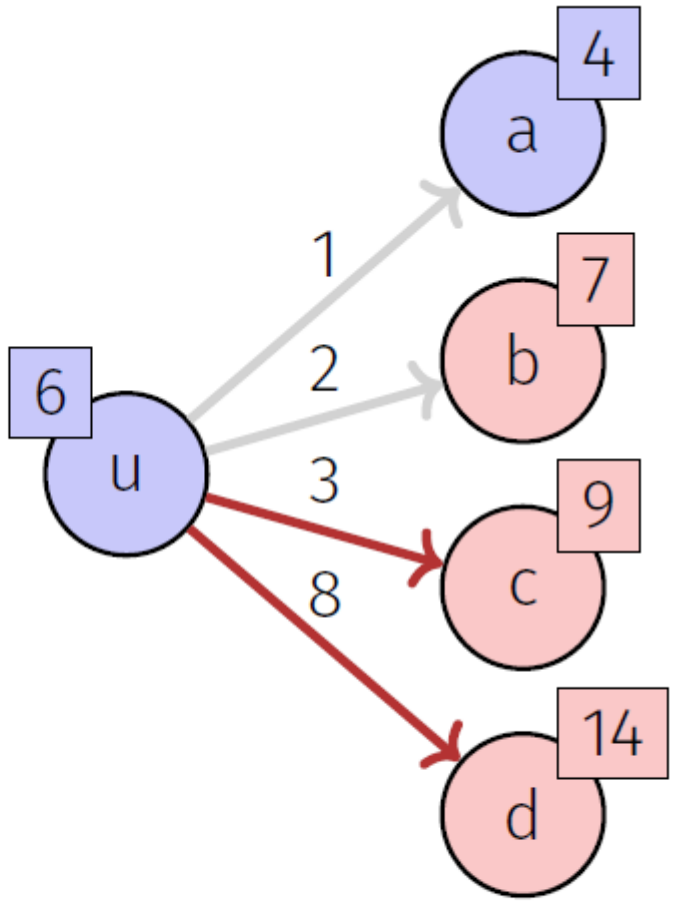
Dijkstra Code

Input: Positiv gewichteter Graph $G = (V, E, c)$,
Startpunkt $s \in V$

Output: Länge d_s der kürzesten Pfade von s und
Vorgänger π_s für jeden Knoten

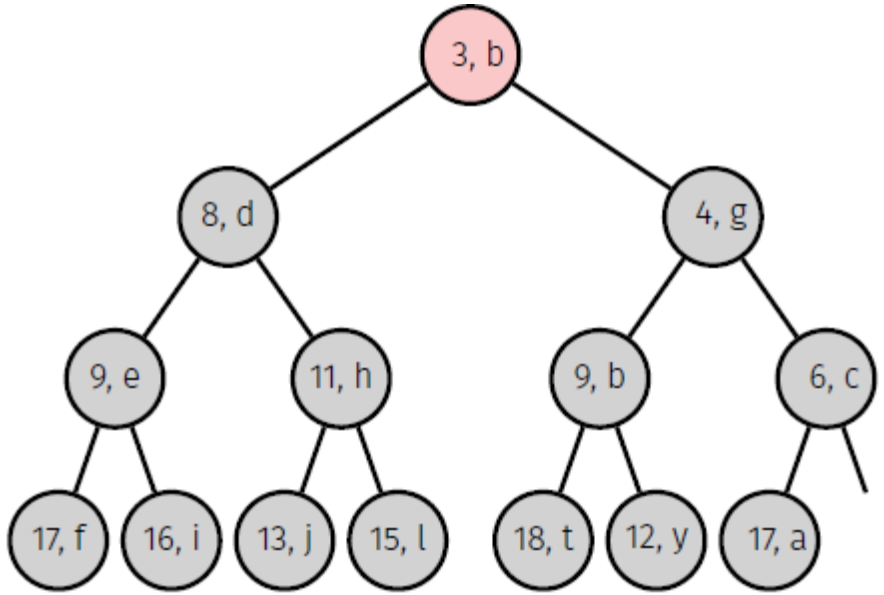
foreach $u \in V$ **do**
  $d_s[u] \leftarrow \infty$; $\pi_s[u] \leftarrow$ **null**

$d_s[s] \leftarrow 0$; $U \leftarrow \{s\}$
while $U \neq \emptyset$ **do**
  $u \leftarrow \arg\min_{u \in U} d_s[u]$; $U \leftarrow U \setminus \{u\}$
  foreach $v \in N^+(u)$ **do**
    if $d_s[u] + c(u, v) < d_s[v]$ **then**
      $d_s[v] \leftarrow d_s[u] + c(u, v)$
      $\pi_s[v] \leftarrow u$
      $U \leftarrow U \cup \{v\}$



Dijkstra benötigte Operationen

Konstruire die Menge von U (Knoten, die von s erreichbar sind) als eine minHeap



Benötigte Operationen:

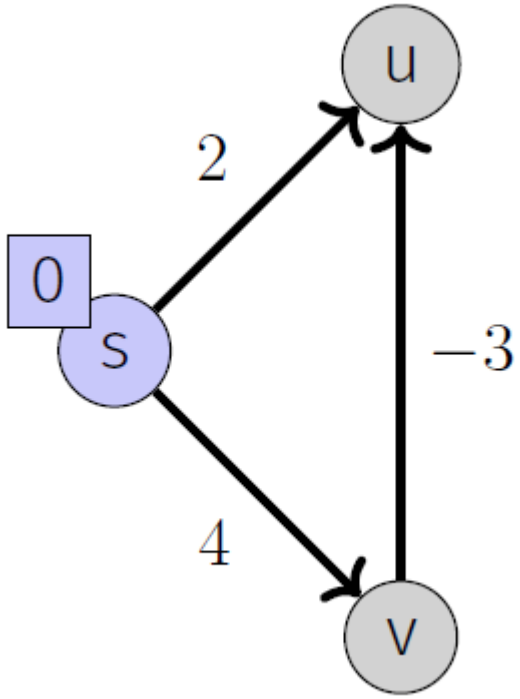
- `Insert((p, k))`: $\mathcal{O}(\log |V|)$
  Hinzufügen eines Schlüssels (Knoten) k mit Wert (obere Schranke) p
- `ExtractMin()`: $\mathcal{O}(\log |V|)$
  Entfernen des Elements mit kleinstem Wert
- `DecreaseKey((p, k))`: **`O(nlog(n))`→ `effizient`**
  Updaten des Werts von Schlüssel k zu p

Lazy Deletion: Knoten erneut einfügen, beim entnehmen prüfen, ob schon herausgenommen

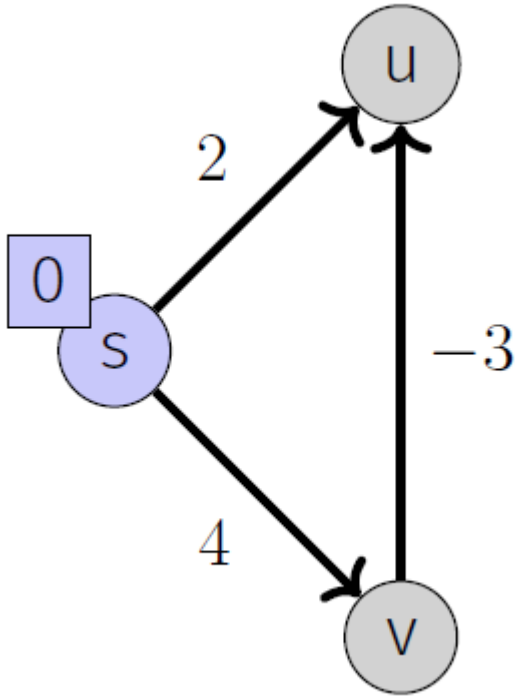
$\Rightarrow$ Speicherbedarf vom Heap wächst bis zu $\Theta(|V| + |E|)$ statt $\Theta(|V|)$

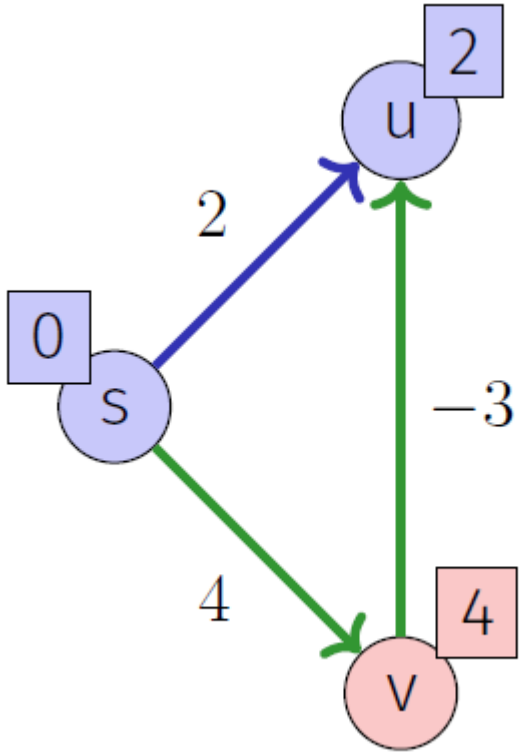
$\Rightarrow$ Da $|E| \leq |V|^2$: `Insert` und `ExtractMin` in $\mathcal{O}(\log |V|^2) = \mathcal{O}(\log |V|)$

Funktioniert Dijkstra hier?



Funktioniert Dijkstra hier?

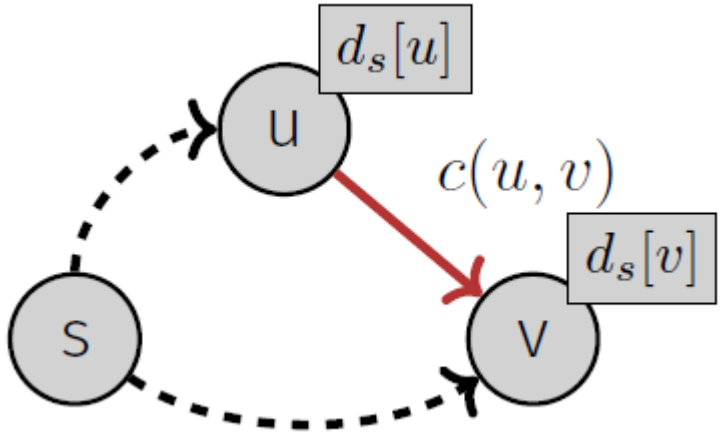




Nein, denn sobald es negative Kanten gibt, erkennt es den kürzesten Weg nicht.

01 – Theorie
01.1 Dijkstra
01.2 Bellman Ford

Relaxation von Kanten



$$d_s[v] \leq d_s[u] + c((u,v))$$

wegen der **Dreiecksungleichung**: $\delta(s,v) \leq \delta(s,u) + \delta(u,v)$

Ein kürzester Weg von s nach v (ohne weitere Einschränkungen) kann nicht länger sein als ein kürzester Weg von s nach v, der u enthalten muss.

Relation Algorithmus

1. Initialisiere d_s und π_s: $d_s[v] = \infty$, $\pi_s[v] =$ **null** für alle $v \in V$
2. Setze $d_s[s] \leftarrow 0$
3. Wähle eine Kante $(u, v) \in E$

Relax$(e = (u, v))$:

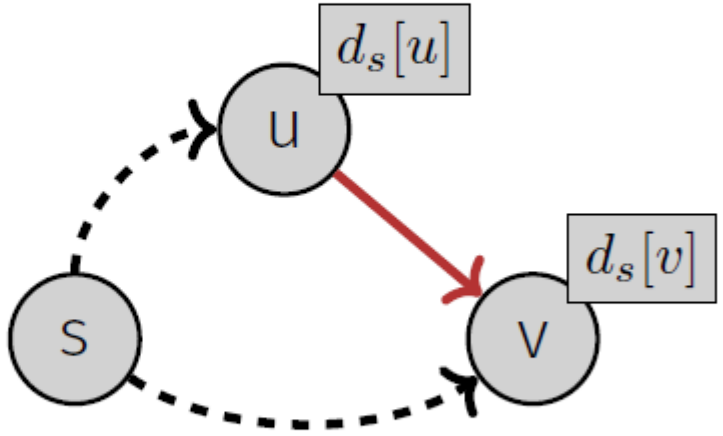
if $d_s[u] + c((u, v)) < d_s[v]$ **then**

$d_s[v] \leftarrow d_s[u] + c((u, v))$

$\pi_s[v] \leftarrow u$

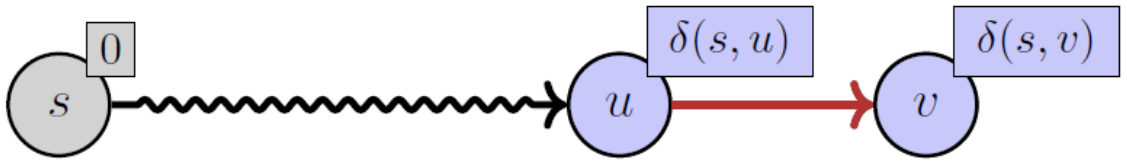
return true

return false

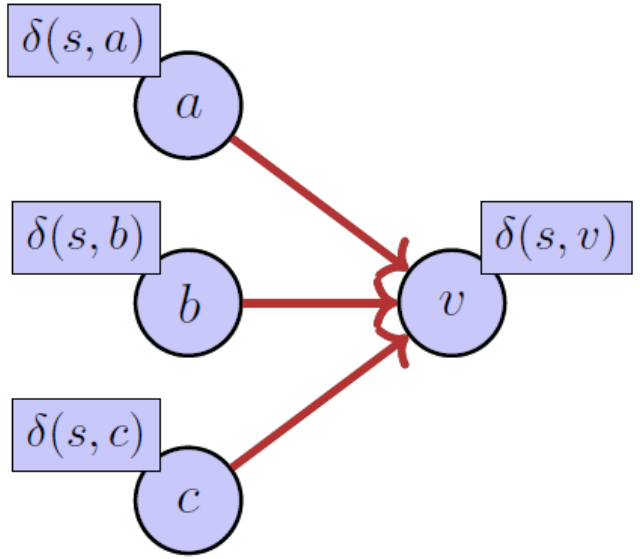


4. Wiederhole 3 bis nichts mehr relaxiert werden kann.
   (bis $d_s[v] \leq d_s[u] + c((u, v)) \quad \forall (u, v) \in E$)

Reihenfolge von Kanten



u der Vorgänger von v auf dem kürzesten Weg $s \overset{p}{\rightsquigarrow} v$



Sobald der kürzeste Weg $s \rightsquigarrow u$ bekannt ist, also $d_s[u] = \delta(s,u)$,
führt das Relaxieren von $e = (u,v)$ zum kürzesten Weg von s nach v:

$$d_s[v] \leq d_s[u] + c((u,v)) = \delta(s,u) + c((u,v)) = \delta(s,v)$$

Bellman Ford Algorithmus

Input: Graph $G = (V, E, c)$, Startpunkt $s \in V$
Output: Wenn Rückgabe true, Minimale Gewichte d der kürzesten Wege zu jedem Knoten, sonst kein kürzester Weg.

foreach $u \in V$ **do**
  $d_s[u] \leftarrow \infty$; $\pi_s[u] \leftarrow$ **null**
$d_s[s] \leftarrow 0$;
for $i \leftarrow 1$ **to** $|V|$ **do**
  $f \leftarrow$ false
  foreach $e \in E$ **do**
    $f \leftarrow f \vee \mathrm{Relax}(e)$
  if $f =$ false **then return** true
return false;

Laufzeit $\mathcal{O}(n \cdot m)$

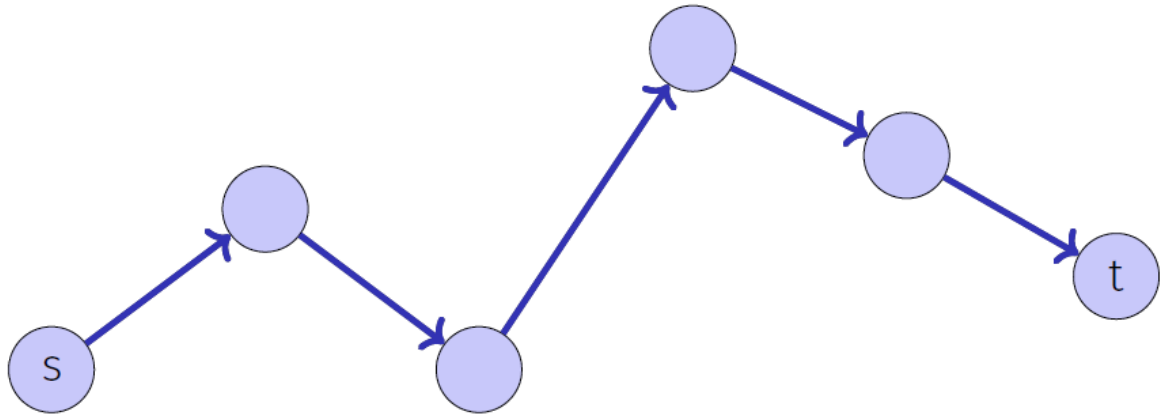
- Gewicht des kürzesten bisher gefundenen Pfades

$$d_s : V \to \mathbb{R}$$

- Vorgänger eines Knotens

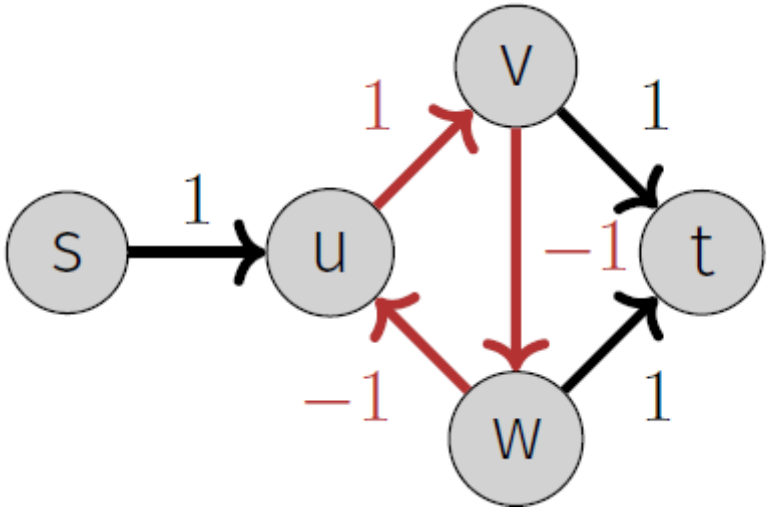
$$\pi_s : V \to V$$

Zu Beginn $\pi_s[v]$ undefiniert für jeden Knoten $v \in V$



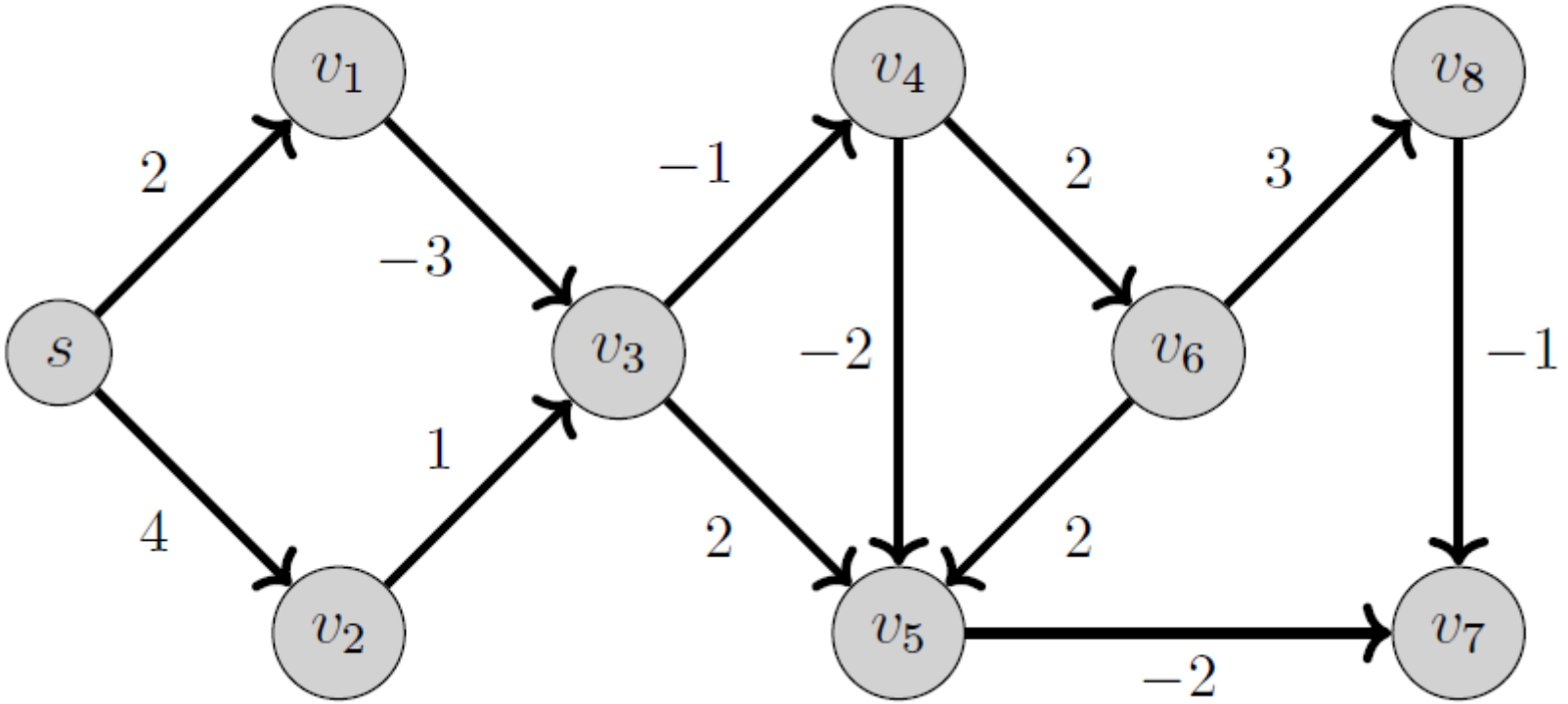
kürzester Weg $p = s \rightsquigarrow t$

Wann funktioniert Bellmann-Ford nicht?

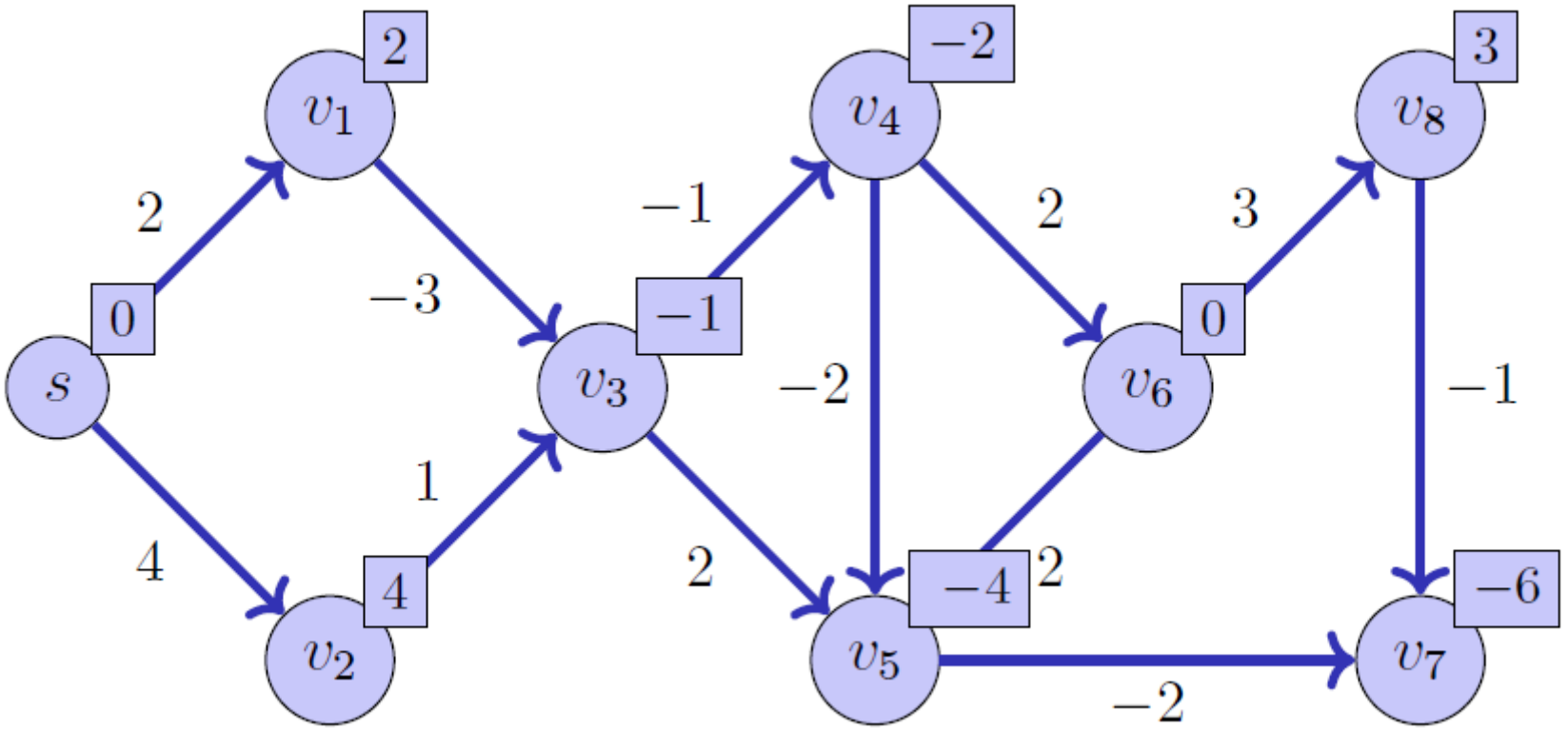


Negative Zyklen können unendlich durchgelaufen werden
→ Kein optimaler Weg möglich.

Lösung: Reihenfolge von Kanten



Lösung: Reihenfolge von Kanten



Topologische Sortierung gibt eine optimale Reihenfolge, die sicherstellt, dass keine negative Zyklen vorkommen:

`s, v1, v2, v3, v4, v6, v5, v8, v7`

Vergleich von Traversierungsalgorithmen

Problem	Methode	Laufzeit
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$
DAG[1] ($c \in \mathbb{R}$)	Top-Sort	$\mathcal{O}(m+n)$
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$
allgemein ($c \in \mathbb{R}$)	Bellman-Ford	$\mathcal{O}(m \cdot n)$

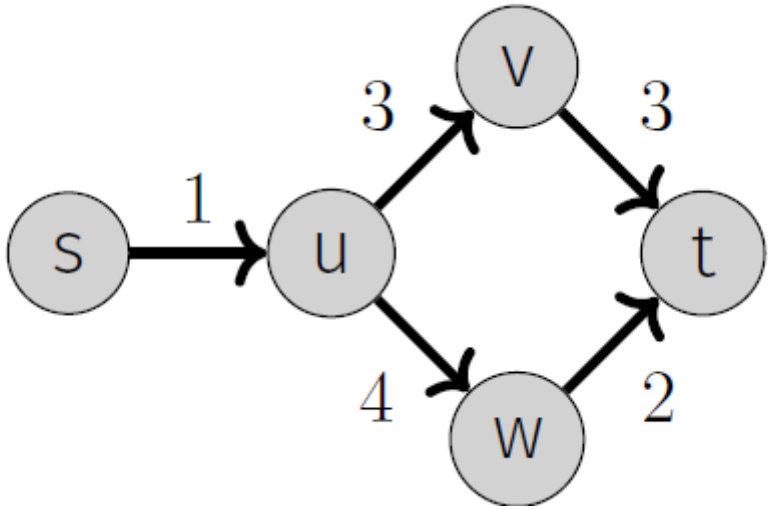
$n := |V|, m := |E|$

[1] Directed Acyclic Graph

02 – Quiz

Quiz Frage 1

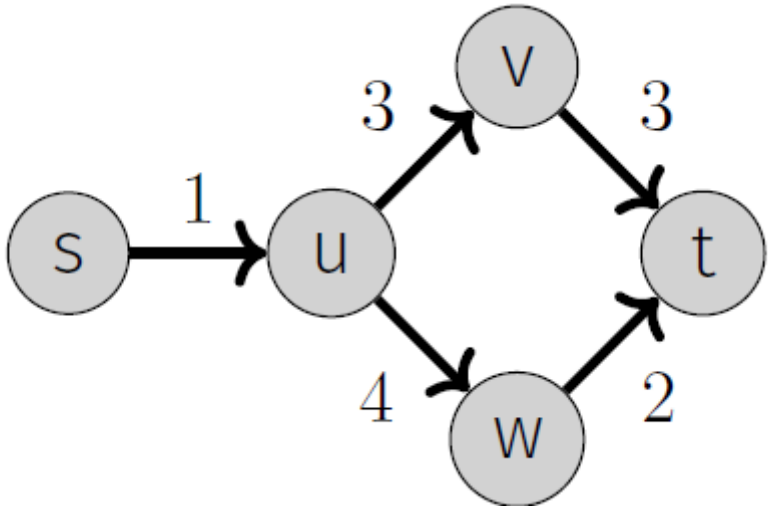
Ist da kürzeste mögliche Weg durch einen Graphen eindeutig?



a) Ja
b) Nein

Quiz Frage 1

Ist da kürzeste mögliche Weg durch einen Graphen eindeutig?

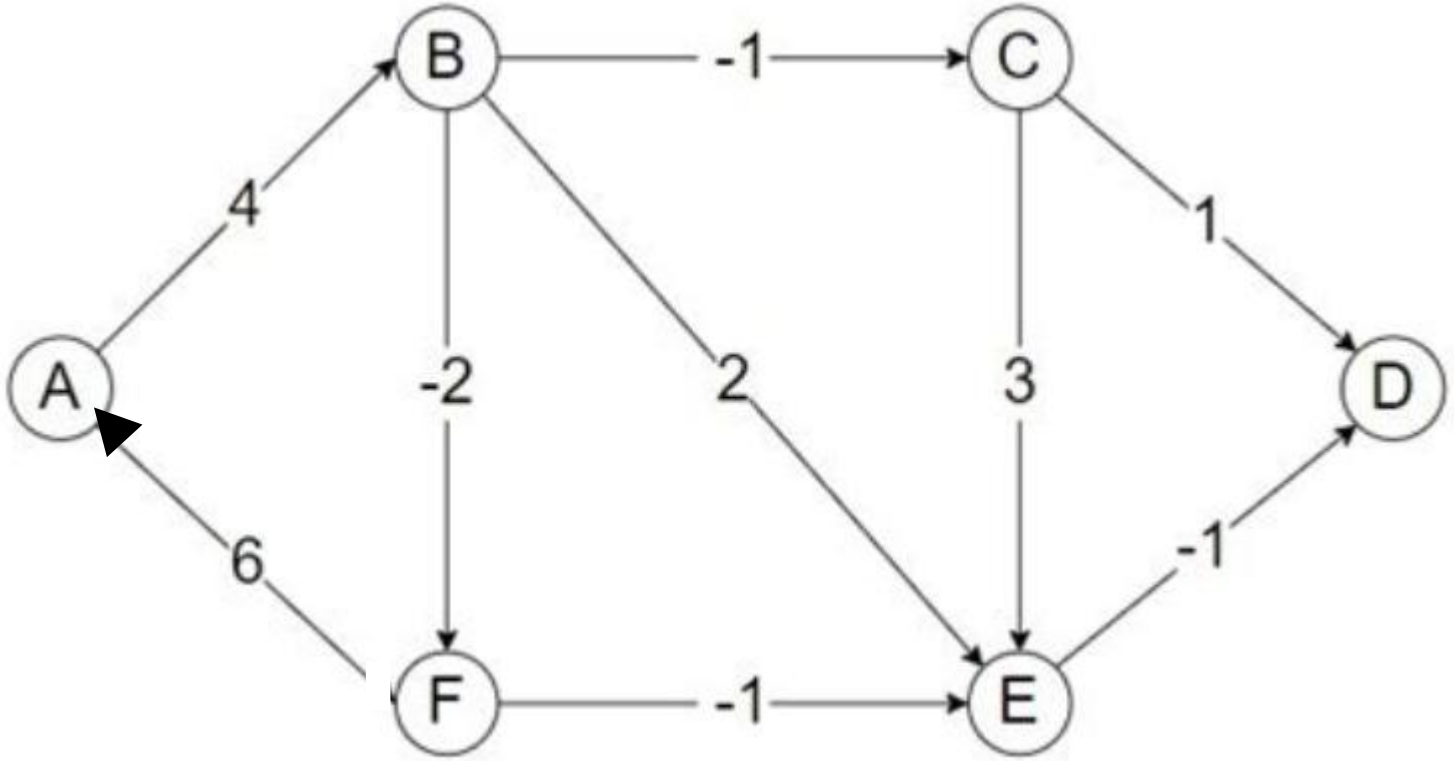


a) Ja
b) Nein

Quiz Frage 2

Welches Algorithmus würde funktionieren, um für diesen Graphen einen kürzesten Weg von A nach D zu finden?

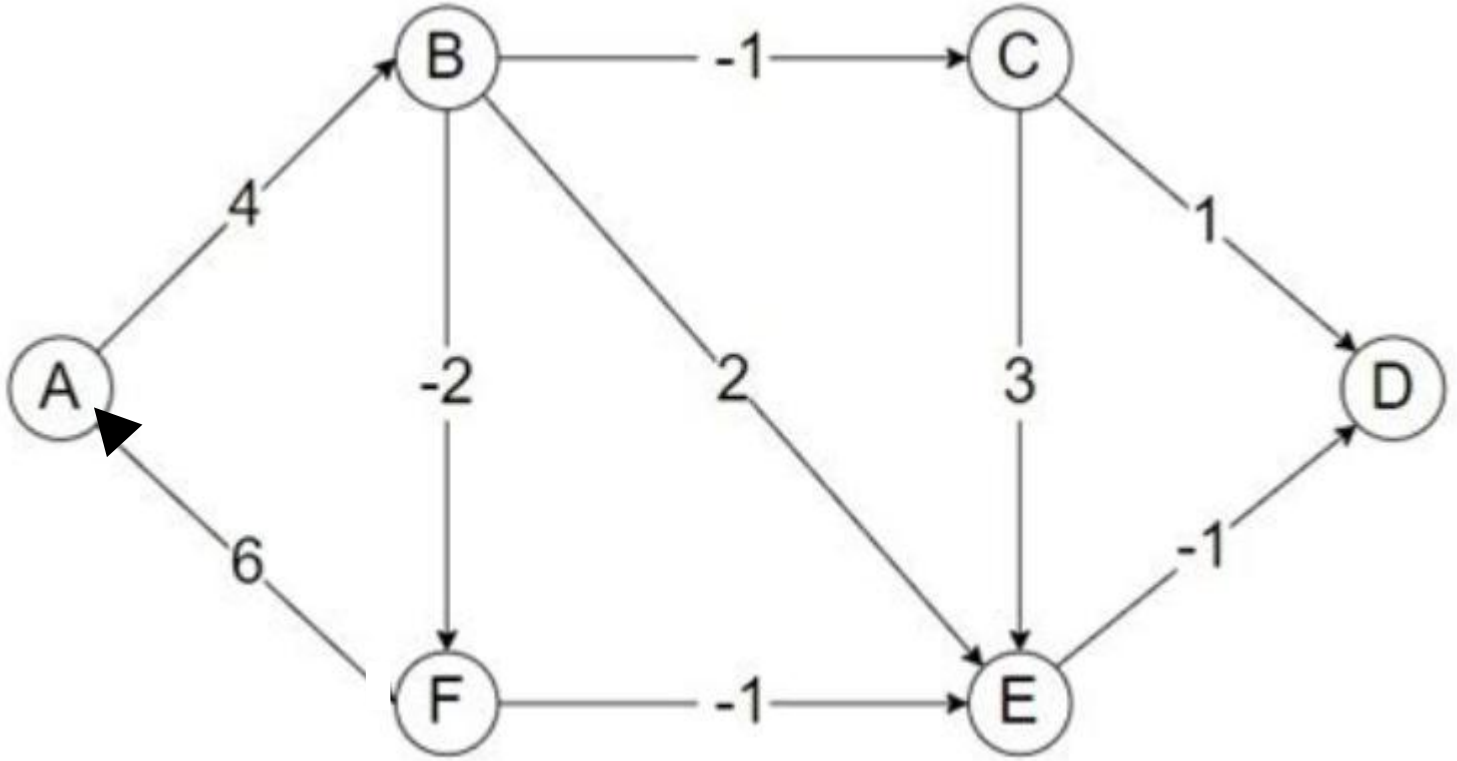
a) BFS
b) Dijkstra
c) Bellmann Ford
d) Topological Sort



Quiz Frage 2

Welches Algorithmus würde funktionieren, um für diesen Graphen einen kürzesten Weg von A nach D zu finden?

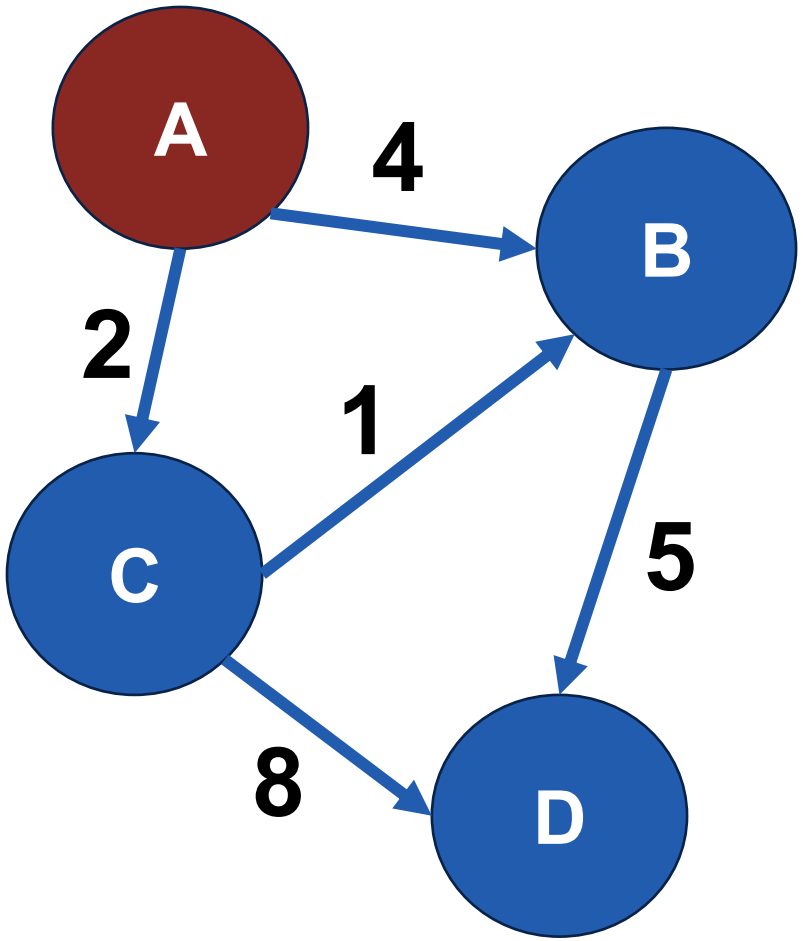
a) BFS
b) Dijkstra
c) Bellmann Ford
d) Topological Sort



Quiz Frage 3

Welcher Knoten wird nach dem Start von Dijkstra Algorithmus als zweiter in die Menge S aufgenommen?

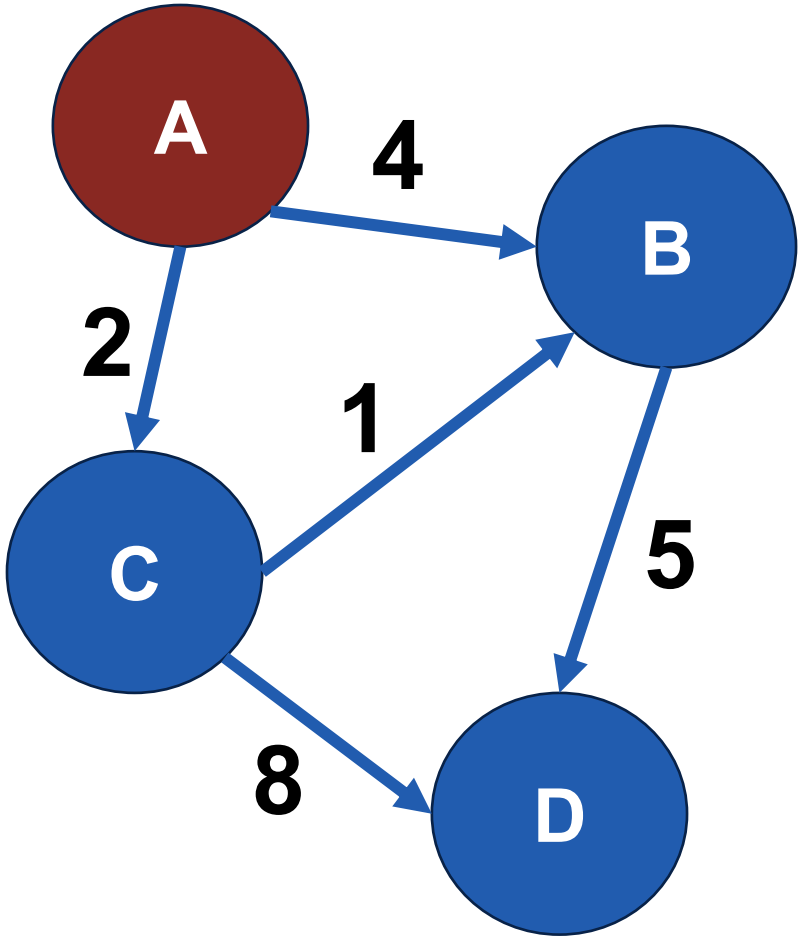
A) D
B) C
C) B
D) keiner, da Dijkstra hier nicht funktioniert



Quiz Frage 3

Welcher Knoten wird nach dem Start von Dijkstra Algorithmus als zweiter in die Menge S aufgenommen?

A) D
B) C
C) B
D) keiner, da Dijkstra hier nicht funktioniert



Quiz Frage 4

Welche Operationen müssen Dijkstra mit optimalen Laufzeit unterstützen?

A) Insert, ExtractMin, DecreaseKey
B) Insert, Delete, Merge
C) Pop, Push, Peek
D) Insert, Remove, Sort

Quiz Frage 4

Welche Operationen müssen Dijkstra mit optimalen Laufzeit unterstützen?

A) Insert, ExtractMin, DecreaseKey
B) Insert, Delete, Merge
C) Pop, Push, Peek
D) Insert, Remove, Sort

Quiz Frage 5

Welchen Einfluss hat Lazy Deletion auf Speicher- und Laufzeitkomplexität?

A) Speicherverbrauch sinkt, Laufzeit bleibt gleich
B) Speicherverbrauch steigt (mehr Einträge im Heap), Laufzeit sinkt
C) Speicherverbrauch und Laufzeit sinken beide
D) Speicherverbrauch bleibt gleich, Laufzeit wird steigt

Quiz Frage 5

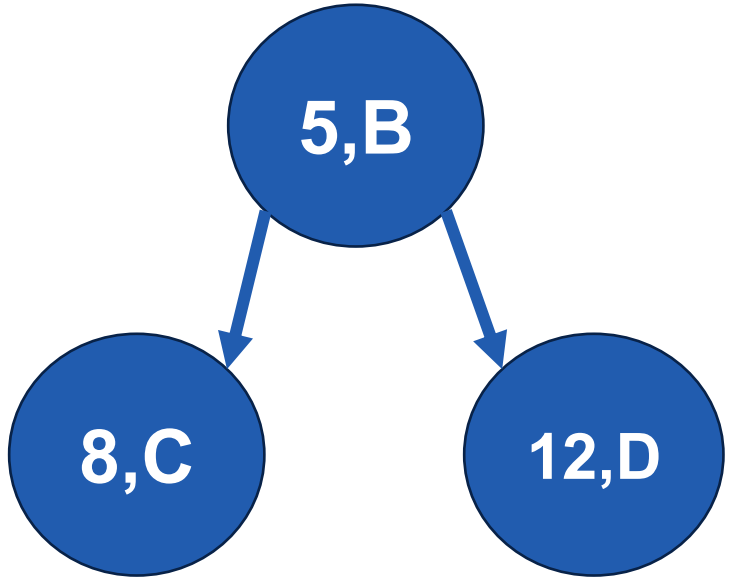
Welchen Einfluss hat Lazy Deletion auf Speicher- und Laufzeitkomplexität?

A) Speicherverbrauch sinkt, Laufzeit bleibt gleich
B) Speicherverbrauch steigt (mehr Einträge im Heap), Laufzeit sinkt
C) Speicherverbrauch und Laufzeit sinken beide
D) Speicherverbrauch bleibt gleich, Laufzeit wird steigt

Quiz Frage 6

Wir haben folgende Heap: (5, B), (8, C), (12, D) Es wurde eine neue Distanz von Dijkstra Algorithmus gefunden: Neue bessere Distanz: $d[B] = 3$ Wie sieht der Heap danach aus?

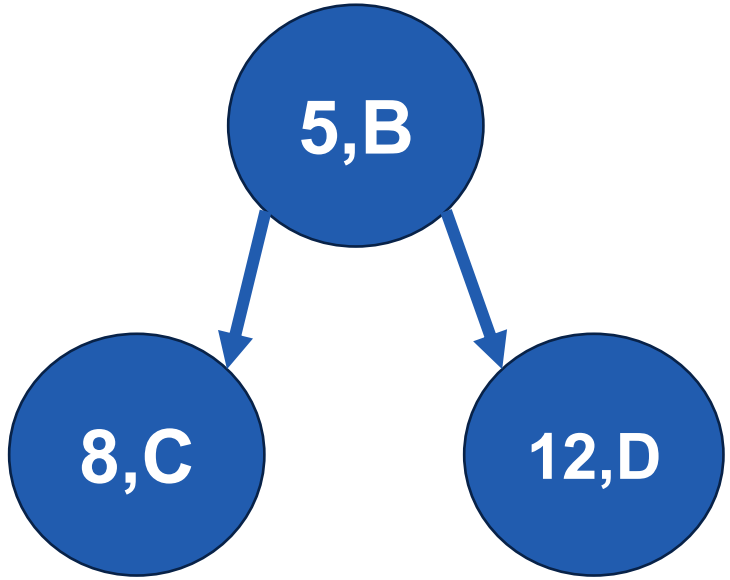
A) (3, B), (8, C), (12, D) — alter Eintrag gelöscht
B) (3, B), (5, B), (8, C), (12, D)
C) Nur (3, B) bleibt übrig
D) B wird komplett entfernt



Quiz Frage 6

Wir haben folgende Priority Queue: (5, B), (8, C), (12, D)
Es wurde eine neue Distanz von Dijkstra Algorithmus gefunden: Neue bessere Distanz: $d[B] = 3$ Wie sieht der Heap danach aus?

A) (3, B), (8, C), (12, D) — alter Eintrag gelöscht
B) (3, B), (5, B), (8, C), (12, D)
C) Nur (3, B) bleibt übrig
D) B wird komplett entfernt



Quiz Frage 7

Wann terminiert der Bellman-Ford-Algorithmus mit true?

A) Wenn ein negativer Zyklus gefunden wurde
B) Wenn in einer ganzen Iteration keine Kante mehr relaxiert wurde
C) Wenn alle Distanzen unendlich bleiben
D) Wenn die Anzahl der Iterationen grösser als |V| ist

Quiz Frage 7

Wann terminiert der Bellman-Ford-Algorithmus mit true?

A) Wenn ein negativer Zyklus gefunden wurde
B) Wenn in einer ganzen Iteration keine Kante mehr relaxiert wurde
C) Wenn alle Distanzen unendlich bleiben
D) Wenn die Anzahl der Iterationen grösser als |V| ist

Quiz Frage 8

d[u] = 5, d[v] = 11, c(u,v) = 3 Was ist nach der Relaxation der neue Wert von d[v]?

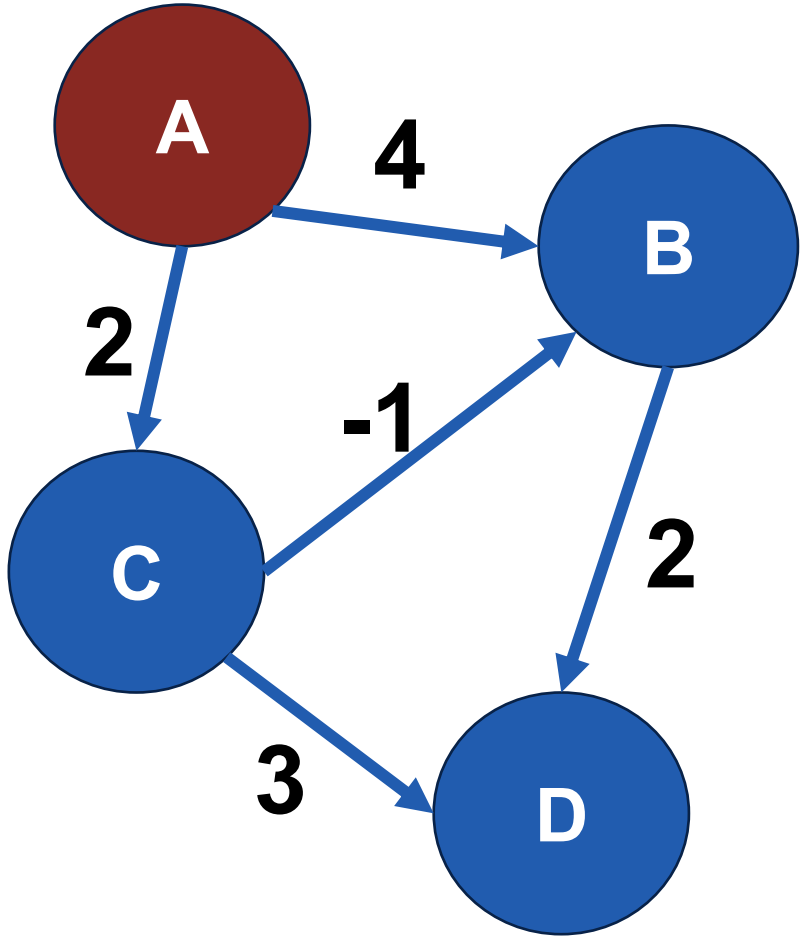
A) 11
B) 8
C) 5
D) 3

Quiz Frage 8

d[u] = 5, d[v] = 11, c(u,v) = 3 Was ist nach der Relaxation der neue Wert von d[v]?

A) 11
B) 8
C) 5
D) 3

Quiz Frage 9

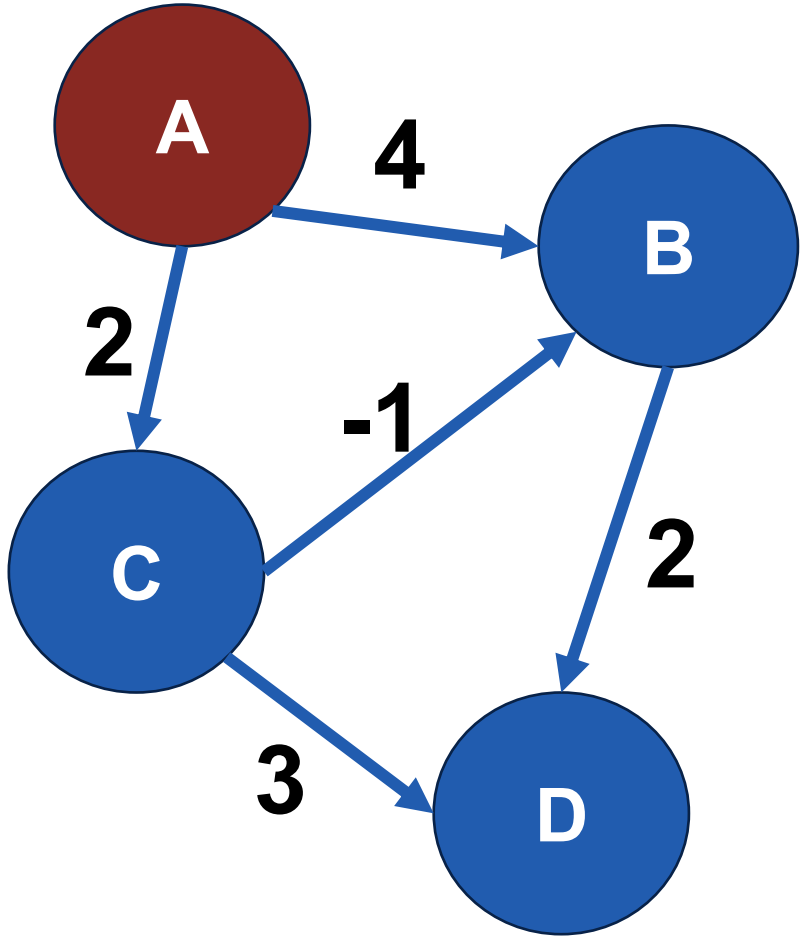


Welche Aussage über die erste Iteration von Bellmann-Ford-Algorithmus ist korrekt?

A) Nur A → C wird relaxiert
B) C → B verbessert d[B] von 4 auf 1
C) D wird in der ersten Iteration bereits endgültig
D) Kein Knoten wird verändert

Quiz Frage 9

Welche Aussage über die erste Iteration von Bellmann-Ford-Algorithmus ist korrekt?



A) Nur A → C wird relaxiert
B) C → B verbessert d[B] von 4 auf 1
C) D wird in der ersten Iteration bereits endgültig
D) Kein Knoten wird verändert

Quiz Frage 10

Was ist die Laufzeit des Bellman-Ford-Algorithmus?

A) $O(|E| \log |V|)$
B) $O(|V| \cdot |E|)$
C) $O(|V|^2)$
D) $O(|E| + |V| \log |V|)$

Quiz Frage 10

Was ist die Laufzeit des Bellman-Ford-Algorithmus?

A) $O(|E| \log |V|)$

B) $O(|V| \cdot |E|)$

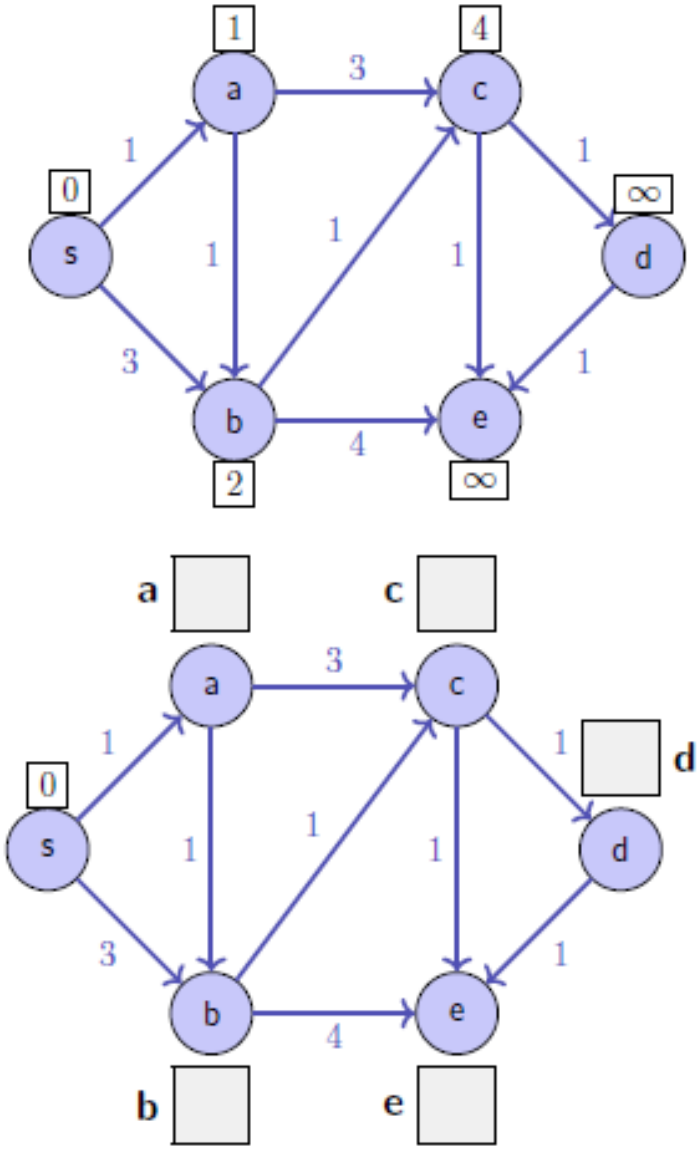
C) $O(|V|^2)$

D) $O(|E| + |V| \log |V|)$

Quiz Frage 11

Nach dem nächsten Schritt des Dijkstra-Algorithmus (ausgehend vom Startknoten s) — welche Schätzungen der kürzesten Wege gelten für die Knoten a, b, c, d, e?

A) $a = 1,\ b = 2,\ c = 3,\ d = \infty,\ e = 6$
B) $a = 1,\ b = 3,\ c = 2,\ d = 5,\ e = 7$
C) $a = 0,\ b = 1,\ c = 4,\ d = 3,\ e = 8$
D) $a = 2,\ b = 2,\ c = 3,\ d = 4,\ e = 5$



Quiz Frage 11

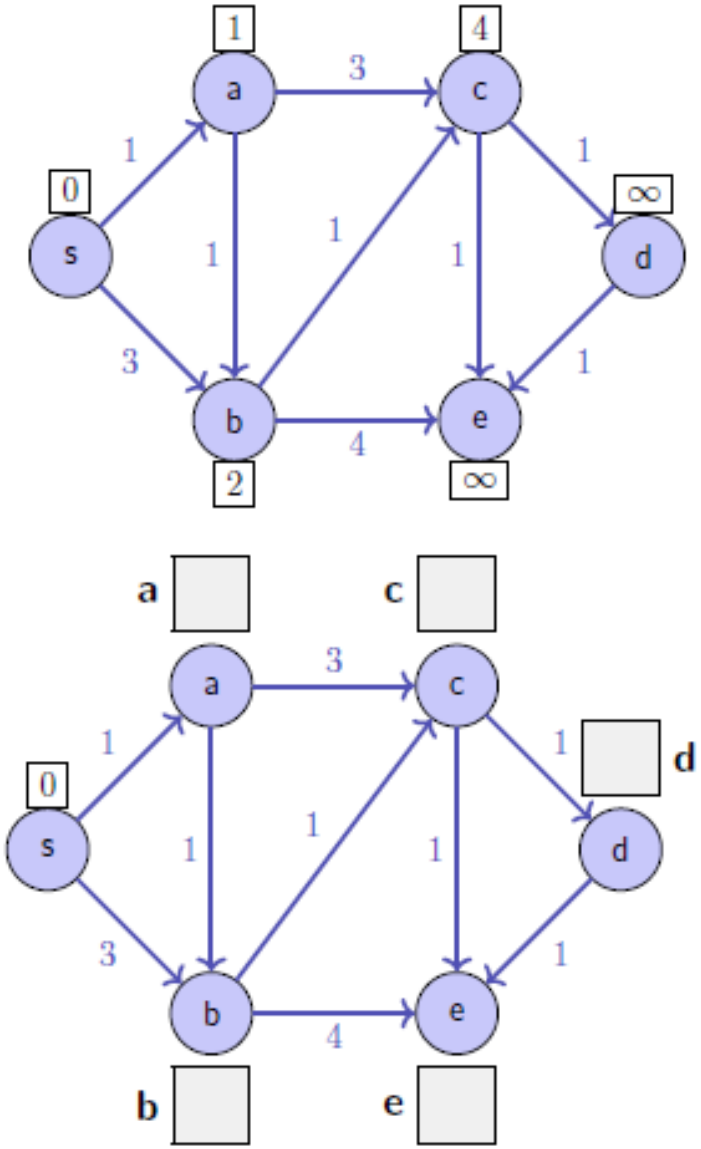
Nach dem nächsten Schritt des Dijkstra-Algorithmus (ausgehend vom Startknoten s) — welche Schätzungen der kürzesten Wege gelten für die Knoten a, b, c, d, e?

A) $\boldsymbol{a = 1}$, $\boldsymbol{b = 2}$, $\boldsymbol{c = 3}$, $\boldsymbol{d = \infty}$, $\boldsymbol{e = 6}$

B) $a = 1$, $b = 3$, $c = 2$, $d = 5$, $e = 7$

C) $a = 0$, $b = 1$, $c = 4$, $d = 3$, $e = 8$

D) $a = 2$, $b = 2$, $c = 3$, $d = 4$, $e = 5$



03 – Programmierübung

Programmierübung: Lazy Deletion

```cpp
#include <iostream>
#include <queue>
#include <memory> // for shared pointer

// some Sprinter record that stores a name together with a time
struct Sprinter{
  const std::string name;
  float time; // the currently known best time
  Sprinter (std::string n): name(n), time(1000) {}
};

// a shared pointer (so we do not need to care about the memory)
using SprinterP = std::shared_ptr<Sprinter>;

// Pair with comparator can be used for the heap with lazy deletion
using Pair = std::pair<SprinterP ,float>;
struct comparator {
  bool operator()(const Pair& a, const Pair& b) const { return a.second > b.second; }
};
// a heap: use std::priority_queue as a wrapper for C++ make_heap (etc)
using Heap = std::priority_queue<Pair, std::vector<Pair>, comparator>;

class HeapWithLazyDeletion {
public:
  HeapWithLazyDeletion(){}

  // insert a Sprinter together with his or her best known sprint time into the heap
  void insert(SprinterP n){
    heap.push(Pair(n,n->time));
  }
```

1. Geh auf CodeExpert → Code Examples → Lazy Deletion
2. Implementiere `extract_min`, das die kürzeste Laufzeit extrahiert und die veraltete Einträge ignoriert.

```
extracted Usain Bolt with time 9.58
  (lazy deletion of Usain Bolt with outdated time 9.62)
extracted Tyson Gay with time 9.71
  (lazy deletion of Usain Bolt with outdated time 9.75)
  (lazy deletion of Tyson Gay with outdated time 9.78)
extracted Asafa Powell with time 9.84
  (lazy deletion of Usain Bolt with outdated time 10)
  (lazy deletion of Asafa Powell with outdated time 10.5)
  (lazy deletion of Tyson Gay with outdated time 11)
```

ETH zürich

Michal Oskedra
Teaching Assistent D-ITET
moskedra@ethz.ch