ETH zürich

Übung 10

Datenstrukturen und Algorithmen, D-MATH, ETH Zurich

Programm von heute

Feedback letzte Übungen

Wiederholung Theorie
- Graphen
- Kürzeste Wege
- Dijkstra
- Heaps, DecreaseKey und Lazy Deletion
- Laufzeiten der Algorithmen
- Dijkstra und negative Kantengewichte?
- Bellman-Ford

1. Feedback letzte Übungen

2. Wiederholung Theorie

2.1 Graphen

Quiz: Laufzeiten einfacher Operationen

Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden		
$v \in V$ ohne Nachbar/Nachfolger finden		
$(v, u) \in E$?		
Kante einfügen		
Kante löschen		

Quiz: Laufzeiten einfacher Operationen

Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	
$v \in V$ ohne Nachbar/Nachfolger finden		
$(v, u) \in E$?		
Kante einfügen		
Kante löschen		

Quiz: Laufzeiten einfacher Operationen

Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	$\Theta(\deg^+ v)$
$v \in V$ ohne Nachbar/Nachfolger finden		
$(v, u) \in E$?		
Kante einfügen		
Kante löschen		

Quiz: Laufzeiten einfacher Operationen

Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	$\Theta(\deg^+ v)$
$v \in V$ ohne Nachbar/Nachfolger finden	$\Theta(n^2)$	
$(v, u) \in E$?		
Kante einfügen		
Kante löschen		

Quiz: Laufzeiten einfacher Operationen

Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	$\Theta(\deg^+ v)$
$v \in V$ ohne Nachbar/Nachfolger finden	$\Theta(n^2)$	$\Theta(n)$
$(v, u) \in E$?		
Kante einfügen		
Kante löschen		

Quiz: Laufzeiten einfacher Operationen

Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	$\Theta(\deg^+ v)$
$v \in V$ ohne Nachbar/Nachfolger finden	$\Theta(n^2)$	$\Theta(n)$
$(v, u) \in E$?	$\Theta(1)$	
Kante einfügen		
Kante löschen		

Quiz: Laufzeiten einfacher Operationen

Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	$\Theta(\deg^+ v)$
$v \in V$ ohne Nachbar/Nachfolger finden	$\Theta(n^2)$	$\Theta(n)$
$(v, u) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Kante einfügen		
Kante löschen		

Quiz: Laufzeiten einfacher Operationen

Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	$\Theta(\deg^+ v)$
$v \in V$ ohne Nachbar/Nachfolger finden	$\Theta(n^2)$	$\Theta(n)$
$(v, u) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Kante einfügen	$\Theta(1)$	
Kante löschen		

Quiz: Laufzeiten einfacher Operationen

Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	$\Theta(\deg^+ v)$
$v \in V$ ohne Nachbar/Nachfolger finden	$\Theta(n^2)$	$\Theta(n)$
$(v, u) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Kante einfügen	$\Theta(1)$	$\Theta(1)$
Kante löschen		

Quiz: Laufzeiten einfacher Operationen

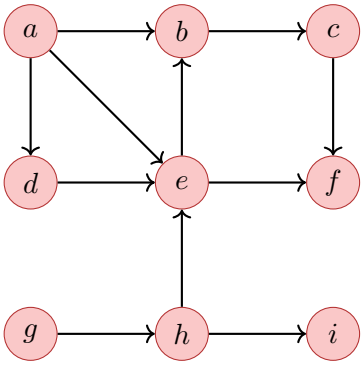
Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	$\Theta(\deg^+ v)$
$v \in V$ ohne Nachbar/Nachfolger finden	$\Theta(n^2)$	$\Theta(n)$
$(v, u) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Kante einfügen	$\Theta(1)$	$\Theta(1)$
Kante löschen	$\Theta(1)$	

Quiz: Laufzeiten einfacher Operationen

Operation	Matrix	Liste
Nachbarn/Nachfolger von $v \in V$ finden	$\Theta(n)$	$\Theta(\deg^+ v)$
$v \in V$ ohne Nachbar/Nachfolger finden	$\Theta(n^2)$	$\Theta(n)$
$(v, u) \in E$?	$\Theta(1)$	$\Theta(\deg^+ v)$
Kante einfügen	$\Theta(1)$	$\Theta(1)$
Kante löschen	$\Theta(1)$	$\Theta(\deg^+ v)$

Breitensuche BFS

BFS von a aus:

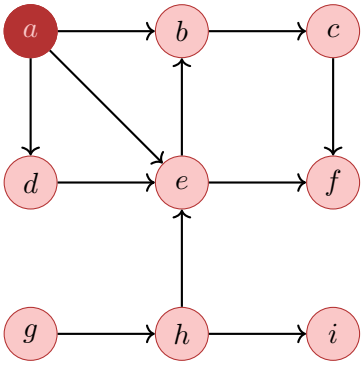


BFS-Baum: Distanzen und Vorgänger

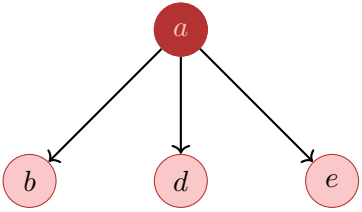
Distanz 0

Breitensuche BFS

BFS von a aus:



BFS-Baum: Distanzen und Vorgänger

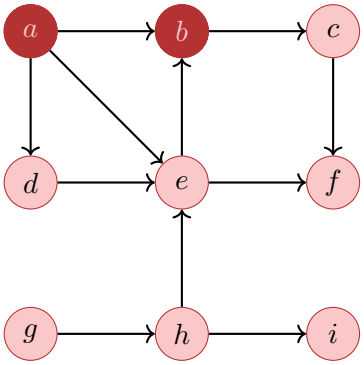


Distanz 0

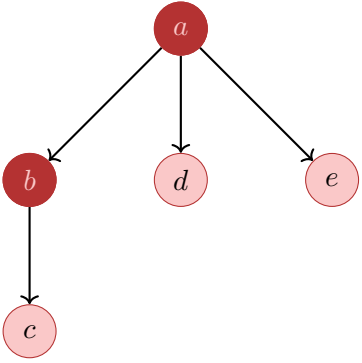
Distanz 1

Breitensuche BFS

BFS von a aus:



BFS-Baum: Distanzen und Vorgänger



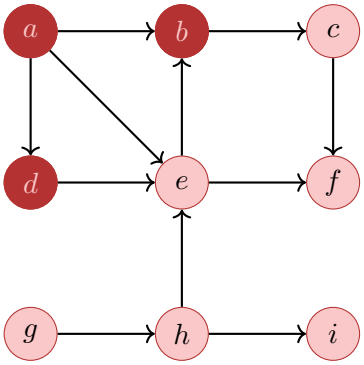
Distanz 0

Distanz 1

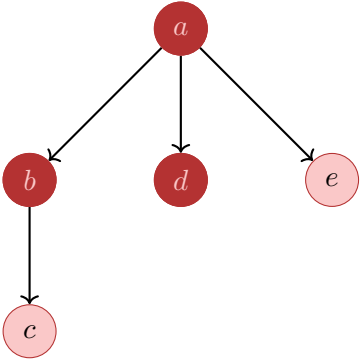
Distanz 2

Breitensuche BFS

BFS von a aus:



BFS-Baum: Distanzen und Vorgänger



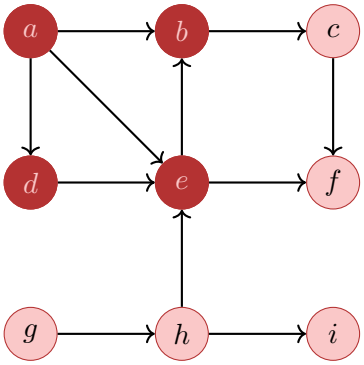
Distanz 0

Distanz 1

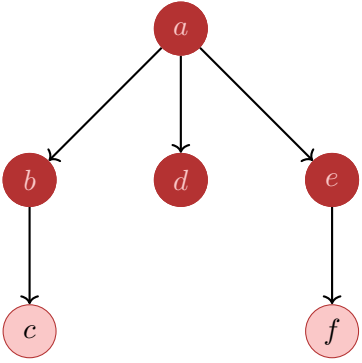
Distanz 2

Breitensuche BFS

BFS von a aus:



BFS-Baum: Distanzen und Vorgänger



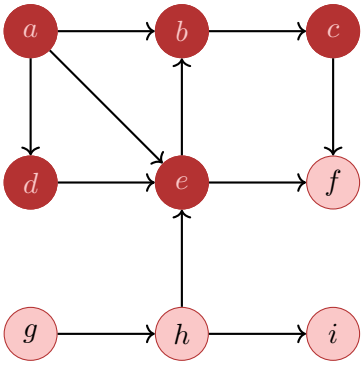
Distanz 0

Distanz 1

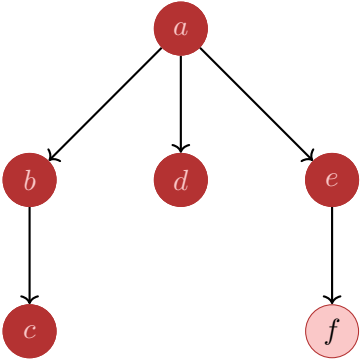
Distanz 2

Breitensuche BFS

BFS von a aus:



BFS-Baum: Distanzen und Vorgänger



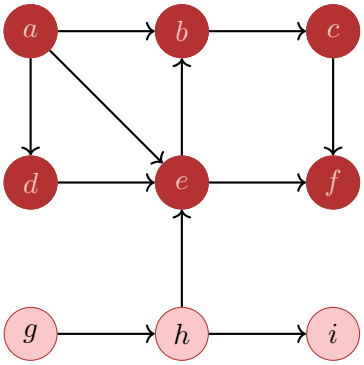
Distanz 0

Distanz 1

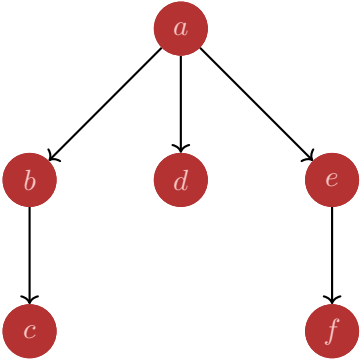
Distanz 2

Breitensuche BFS

BFS von a aus:



BFS-Baum: Distanzen und Vorgänger



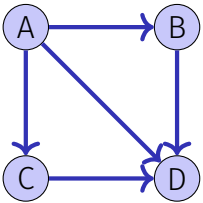
Distanz 0

Distanz 1

Distanz 2

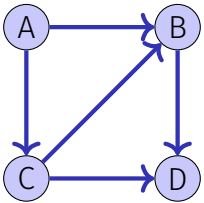
Quiz: Topologisch Sortieren

Auf wie viele Arten können die folgenden gerichteten Graphen jeweils topologisch sortiert werden?



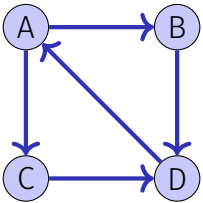
Anzahl Sortierungen

?



Anzahl Sortierungen

?

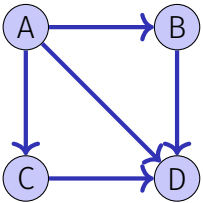


Anzahl Sortierungen

?

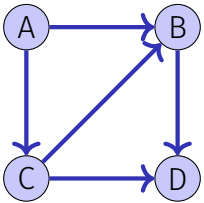
Quiz: Topologisch Sortieren

Auf wie viele Arten können die folgenden gerichteten Graphen jeweils topologisch sortiert werden?



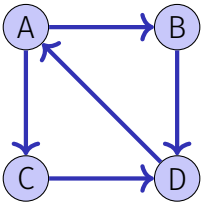
Anzahl Sortierungen

2



Anzahl Sortierungen

1



Anzahl Sortierungen

0

2.2 Kürzeste Wege

Allgemeiner Algorithmus

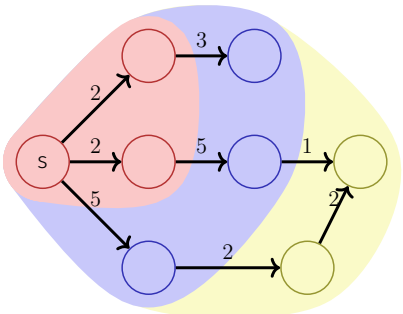
1. Initialisiere d_s und π_s: $d_s[v] = \infty$, $\pi_s[v] = \text{null}$ für alle $v \in V$
2. Setze $d_s[s] \leftarrow 0$
3. Wähle eine Kante $(u, v) \in E$

   Relaxiere (u, v):
   if $d_s[v] > d[u] + c(u, v)$ then
   $d_s[v] \leftarrow d_s[u] + c(u, v)$
   $\pi_s[v] \leftarrow u$
4. Wiederhole 3 bis nichts mehr relaxiert werden kann.
   (bis $d_s[v] \leq d_s[u] + c(u, v) \quad \forall (u, v) \in E$)

Dijkstra (positive Kantengewichte)

Menge V aller Knoten wird unterteilt in

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



Algorithmus Dijkstra(G, s)

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

Output: Minimale Gewichte d der kürzesten Pfade und Vorgängerknoten für jeden Knoten.

foreach $u \in V$ **do**
    $d_s[u] \leftarrow \infty$; $\pi_s[u] \leftarrow null$
$d_s[s] \leftarrow 0$; $R \leftarrow \{s\}$
while $R \neq \emptyset$ **do**
    $u \leftarrow$ ExtractMin(R)
    foreach $v \in N^+(u)$ **do**
        if $d_s[u] + c(u, v) < d_s[v]$ **then**
            $d_s[v] \leftarrow d_s[u] + c(u, v)$
            $\pi_s[v] \leftarrow u$
            $R \leftarrow R \cup \{v\}$

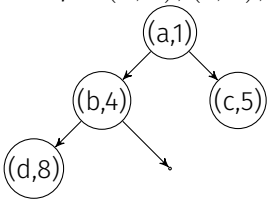
Zur Implementation: Datenstruktur für R?

Relaxieren bei Dijkstra:

if $d_s[u] + c(u, v) < d_s[v]$ **then**
    $d_s[v] \leftarrow d_s[u] + c(u, v)$
    $\pi_s[v] \leftarrow u$
    if $v \notin R$ **then**
        Add(R, v) // Einfügen eines neuen $(v, d(v))$ im Heap zu R
    else
        DecreaseKey(R, v) // Update eines $(v, d(v))$ im Heap zu R

DecreaseKey ?

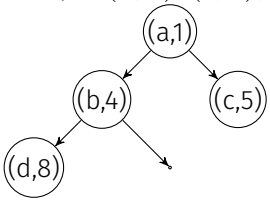
Heap ($(a,1),(b,4),(c,5),(d,8)$) =



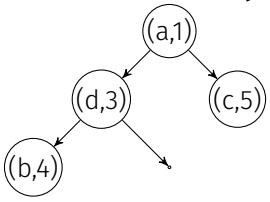
nach DecreaseKey($d,3$):

DecreaseKey ?

Heap ($(a,1),(b,4),(c,5),(d,8)$) =



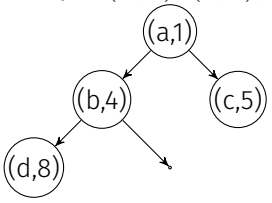
nach DecreaseKey$(d,3)$:



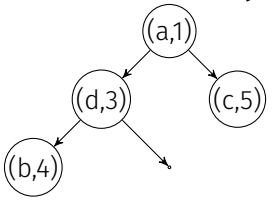
2 Probleme:

DecreaseKey ?

Heap ($(a,1),(b,4),(c,5),(d,8)$) =



nach DecreaseKey($d,3$):

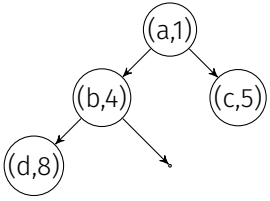


2 Probleme:

- Position von d zuerst nicht bekannt. Suche: $\Theta(n)$
- Position der Knoten kann bei DecreaseKey ändern

Lazy Deletion !

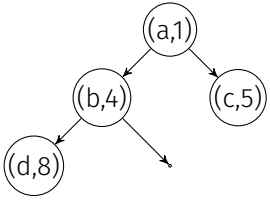
Heap ($(a,1),(b,4),(c,5),(d,8)$) =



Insert$(d,3)$:

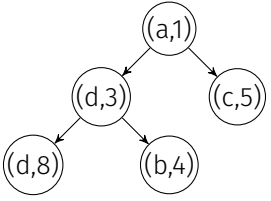
Lazy Deletion !

Heap ($(a,1),(b,4),(c,5),(d,8)$) =



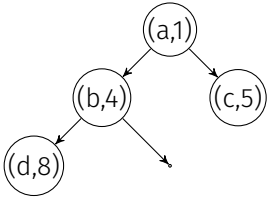
ExtractMin()

Insert($d,3$):

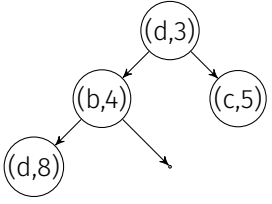


Lazy Deletion !

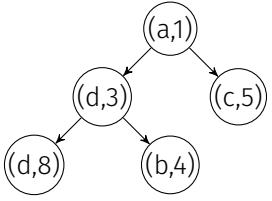
Heap ($(a,1),(b,4),(c,5),(d,8)$) =



ExtractMin() $\rightarrow (a,1)$



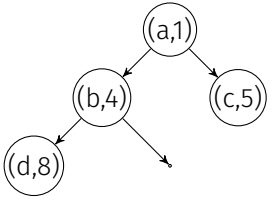
Insert$(d,3)$:



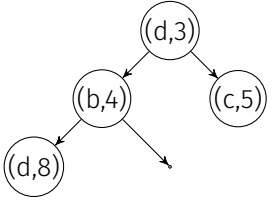
ExtractMin()

Lazy Deletion !

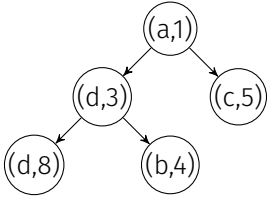
Heap ($(a,1),(b,4),(c,5),(d,8)$) =



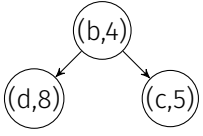
ExtractMin() $\rightarrow (a,1)$



Insert$(d,3)$:



ExtractMin() $\rightarrow (d,3)$



Späteres ExtractMin() $\rightarrow (d,8)$ muss ignoriert werden

Laufzeit Dijkstra

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

- $n\times$ ExtractMin: $\mathcal{O}(n \log n)$
- $m\times$ Insert oder DecreaseKey: $\mathcal{O}(m \log n)$
- $1\times$ Init: $\mathcal{O}(n)$
- Insgesamt: $\mathcal{O}((n + m) \log n)$. Für zusammenhängende Graphen: $\mathcal{O}(m \log n)$.

2.5 Laufzeiten der Algorithmen

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS			

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$		

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG	Top-Sort			

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG	Top-Sort	$\mathcal{O}(m+n)$		

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG	Top-Sort	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG	Top-Sort	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
$c \geq 0$	Dijkstra			

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG	Top-Sort	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$		

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG	Top-Sort	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$	$\mathcal{O}(n^2 \log n)$	

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG	Top-Sort	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n \log n)$
allgemein	Bellman-Ford	$\mathcal{O}(m \cdot n)$		

Quiz: Single Source Shortest Paths

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

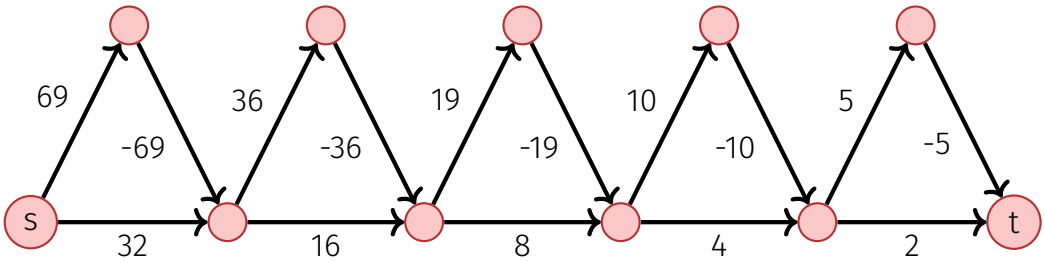
Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG	Top-Sort	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n \log n)$
allgemein	Bellman-Ford	$\mathcal{O}(m \cdot n)$	$\mathcal{O}(n^3)$	

Quiz: Single Source Shortest Paths

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

Problem	Methode	Laufzeit	dicht $m \in \mathcal{O}(n^2)$	dünn $m \in \mathcal{O}(n)$
$c \equiv 1$	BFS	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
DAG	Top-Sort	$\mathcal{O}(m+n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
$c \geq 0$	Dijkstra	$\mathcal{O}((m+n)\log n)$	$\mathcal{O}(n^2 \log n)$	$\mathcal{O}(n \log n)$
allgemein	Bellman-Ford	$\mathcal{O}(m \cdot n)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$

Quiz: Ein Interessanter Graph



Funktioniert Dijkstra?

Antwort

Antwort

Dijkstra (so wie wir es präsentiert haben) funktioniert auch für Graphen mit negativen Kantengewichten, solange keine negativen Zyklen vorhanden sind. Dijkstra kann dann aber exponentielle Laufzeit haben.

Allgemeine Bewertete Graphen

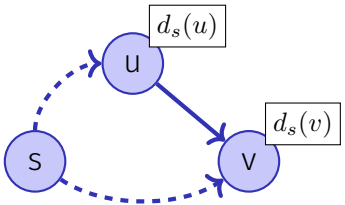
Relax(u, v) $(u, v \in V, (u, v) \in E)$

if $d_s(v) > d_s(u) + c(u, v)$ **then**

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

  return true

return false



Problem: Zyklen mit negativen Gewichten können Weg verkürzen: es muss keinen kürzesten Weg mehr geben

Dynamic-Programming-Ansatz (Bellman)

Induktion über Anzahl Kanten. $d_s[i, v]$: Kürzeste Weglänge von s nach v über maximal i Kanten.

$$d_s[i, v] = \min\{d_s[i-1, v], \min_{(u,v)\in E}(d_s[i-1, u] + c(u, v))$$

$$d_s[0, s] = 0, d_s[0, v] = \infty \; \forall v \neq s.$$

Algorithmus Bellman-Ford(G, s)

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

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 $(u, v) \in E$ **do**
        $f \leftarrow f \vee \mathrm{Relax}(u, v)$
    if $f =$ false **then return** true
return false;