

Wiederholung:

Grundlegende Algorithmen aus der Vorlesung ADS

Übersicht

1. Graphdurchlaufstrategien

1.1 Tiefensuche

Beispiel

Pseudocode

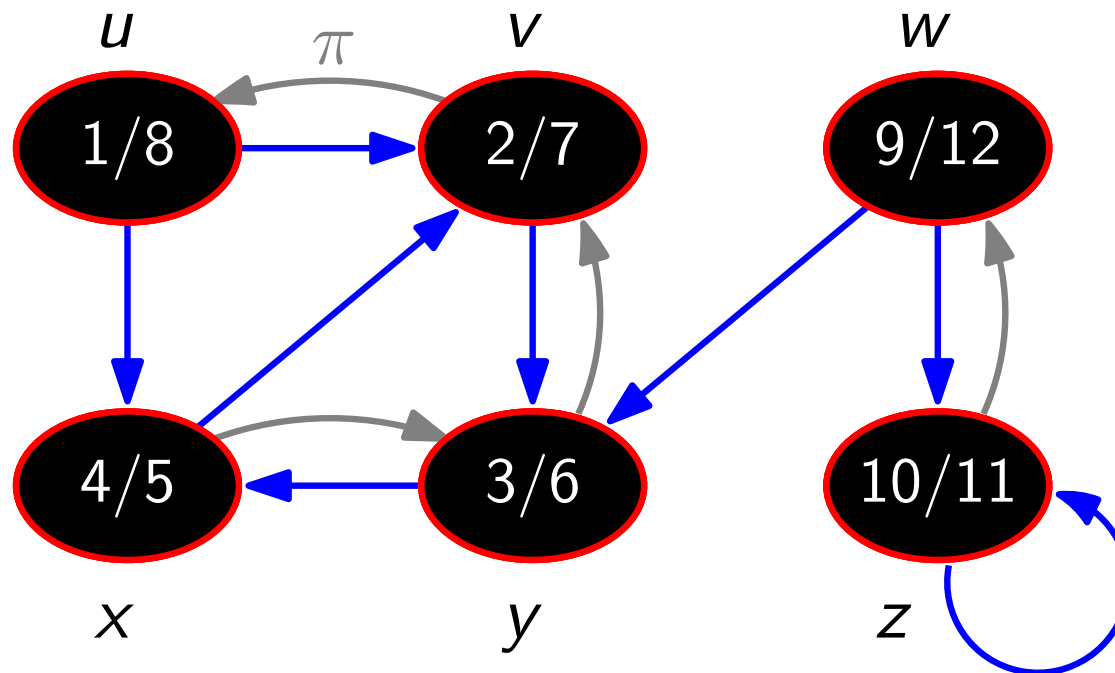
Anwendung

Tiefensuche – Beispiel

Eingabe: (un)gerichteter Graph G

Ausgabe: – Besuchsintervalle $(\overset{\text{discovery}}{u.d} / \overset{\text{finish}}{u.f})$

– DFS-Wald $(\overset{\pi}{\leftarrow})$



Tiefensuche – Pseudocode

DFS(Graph $G = (V, E)$)

foreach $u \in V$ **do**

┌ $u.color = white$
└ $u.\pi = nil$

$time = 0$ // globale Variable!

foreach $u \in V$ **do**

┌ **if** $u.color == white$ **then** DFSVisit(G, u)

DFSVisit(Graph G , Vertex u)

$time = time + 1$

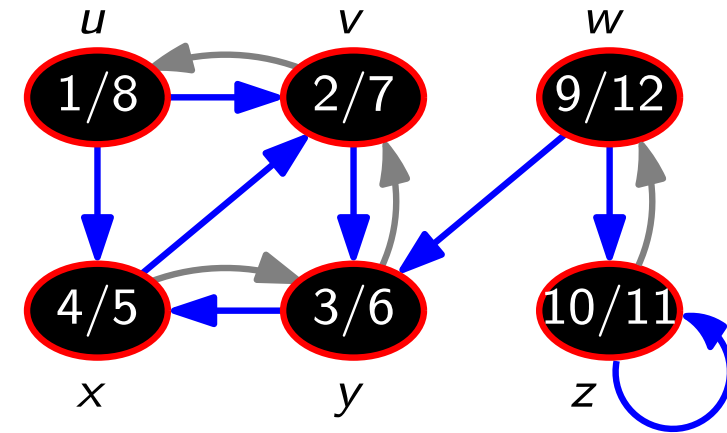
$u.d = time$; $u.color = gray$

foreach $v \in Adj[u]$ **do**

┌ **if** $v.color == white$ **then**
└ $v.\pi = u$; DFSVisit(G, v)

$time = time + 1$

$u.f = time$; $u.color = black$



Laufzeit?

- DFSVisit wird nur für weiße Knoten aufgerufen.
 - In DFSVisit wird der neue Knoten sofort gefärbt.
- ⇒ DFSVisit wird für jeden Knoten genau 1× aufgerufen.
- Jede Kante wird *insgesamt* höchstens 2× betrachtet.

DFS gesamt $O(V + E)$ Zeit

Tiefensuche – Anwendung

Topologische Sortierung: Lineare Ordnung der Knoten, so dass aus $(u, v) \in E$ folgt: u kommt vor v .

\Rightarrow kreisfrei

```
TopologicalSort(DirectedGraph G)
```

```
  L = new List()
```

```
  DFS(G) mit folgender Änderung:
```

```
    Wenn ein Knoten schwarz gefärbt wird,  
    häng ihn vorne an die Liste L an.
```

```
  return L
```

Laufzeit?
 $O(V + E)$

Übersicht

1. Graphdurchlaufstrategien

1.1 Tiefensuche

1.2 Breitensuche

Beispiel

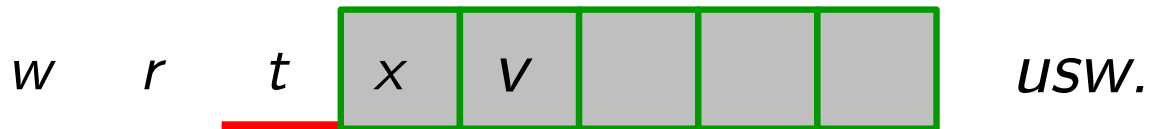
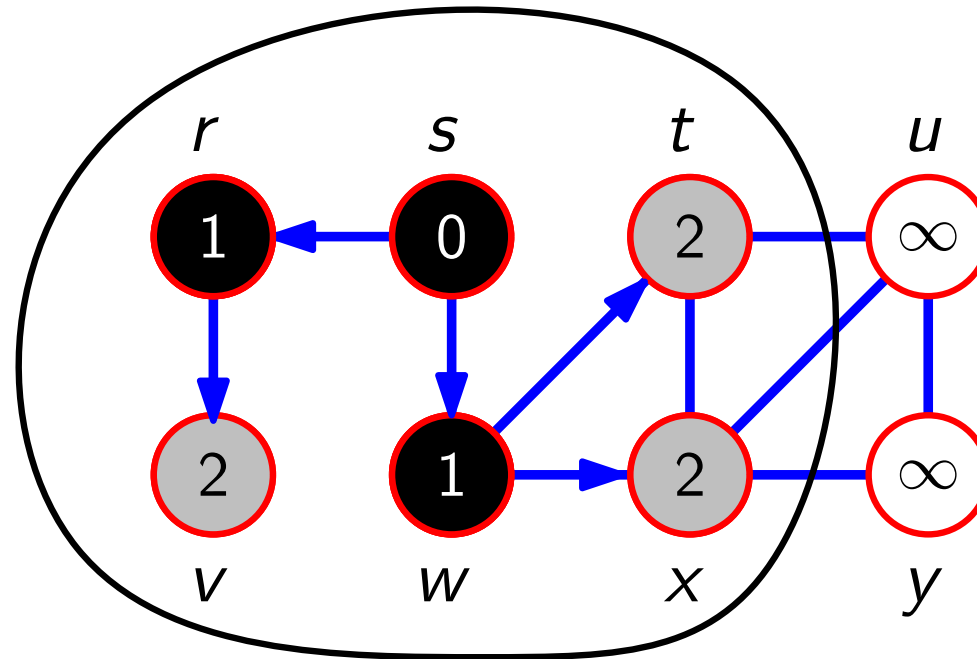
Pseudocode

Anwendung

Breitensuche – Beispiel

Eingabe: (un)gerichteter Graph G

Ausgabe: – Detektionszeitpunkte
– BFS-Wald



Breitensuche – Pseudocode

BFS(Graph G , Vertex s)

Initialize(G , s)

$Q = \text{new Queue}()$

$Q.\text{Enqueue}(s)$

while $Q \neq \emptyset$ **do**

$u = Q.\text{Dequeue}()$

foreach $v \in \text{Adj}[u]$ **do**

if $v.\text{color} == \text{white}$ **then**

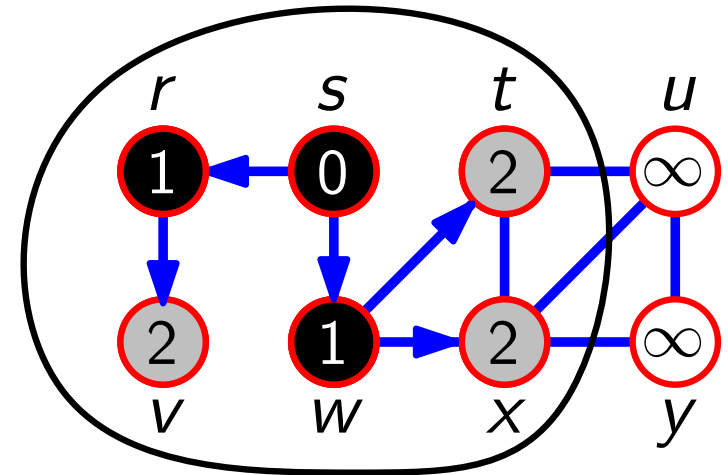
$v.\text{color} = \text{gray}$

$v.d = u.d + 1$

$v.\pi = u$

$Q.\text{Enqueue}(v)$

$u.\text{color} = \text{black}$



Initialize(Graph G , Vertex s)

foreach $u \in V$ **do**

$u.\text{color} = \text{white}$

$u.d = \infty$

$u.\pi = \text{nil}$

$s.\text{color} = \text{gray}$

$s.d = 0$

Laufzeit?

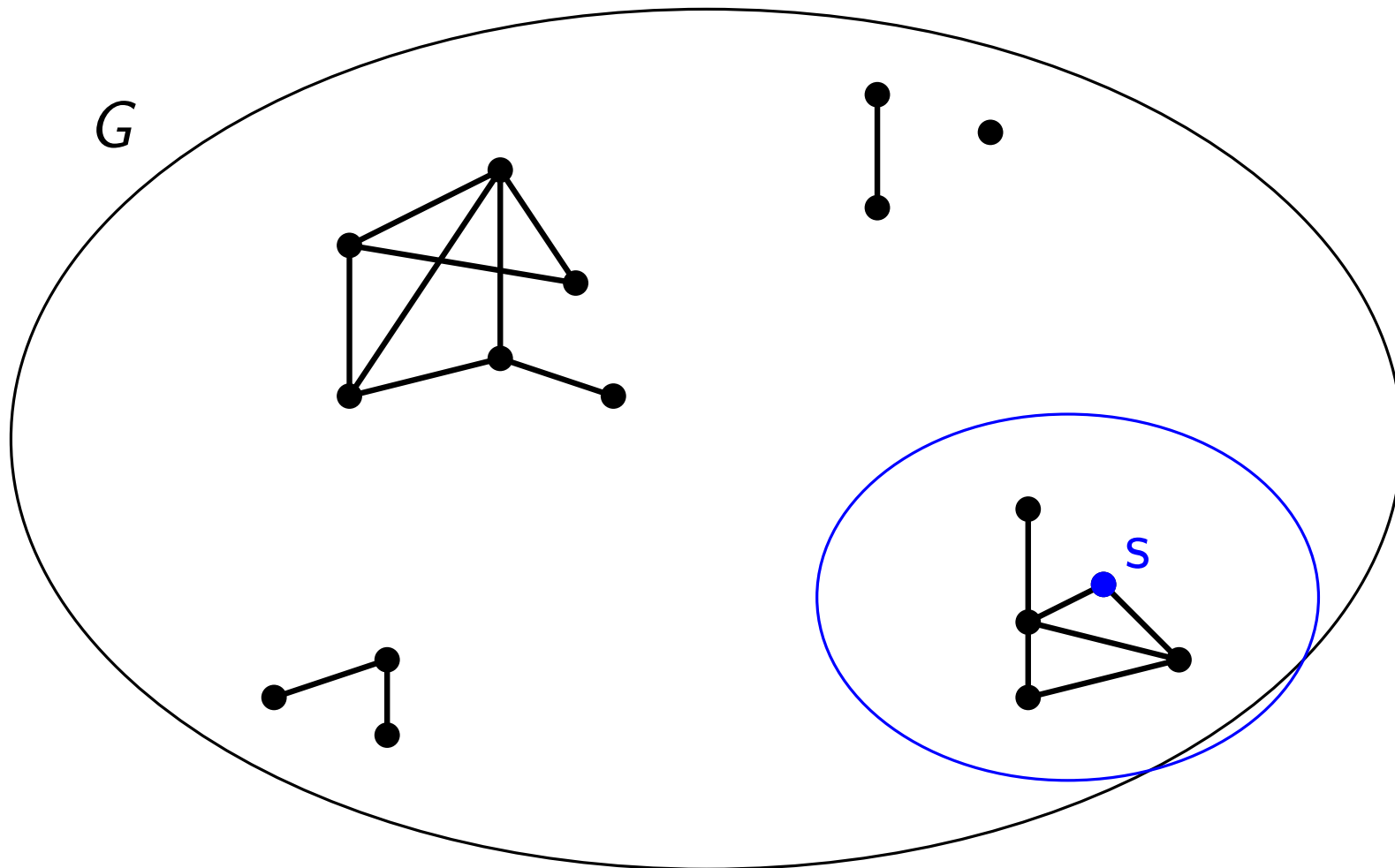
Initialize En-/Dequeues Adjazenzlisten (foreach-Schleifen)

$$O(|V|) + O(|V|) + O(|E|) = O(|V| + |E|)$$

Breitensuche – Anwendung

Zusammenhangskomponente:

Maximale Teilmenge von Knoten, die über Wege miteinander verbunden sind.



Übersicht

2. Kürzeste Wege

2.1 Breitensuche

2.2 Dijkstra

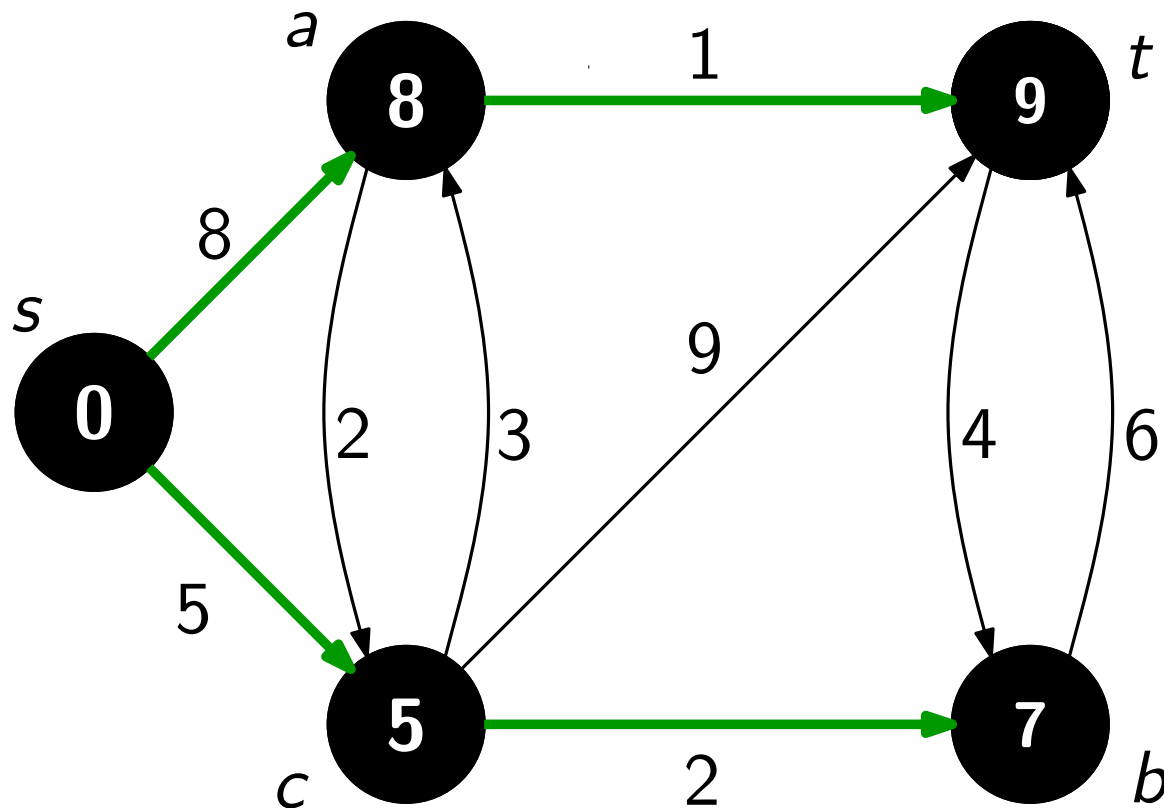
Beispiel

Pseudocode

Dijkstra – Beispiel

Eingabe: gerichteter, zusammenhängender Graph $G(V, E)$ mit nicht-negativen Kantengewichten und Knoten s

Ausgabe: kürzeste s - t -Wege in G mit Vorgänger-Zeiger π



Dijkstra – Pseudocode

Dijkstra(WeightedGraph G , Vertex s)

Initialize(G, s)

$Q = \text{new PriorityQueue}(V, d)$

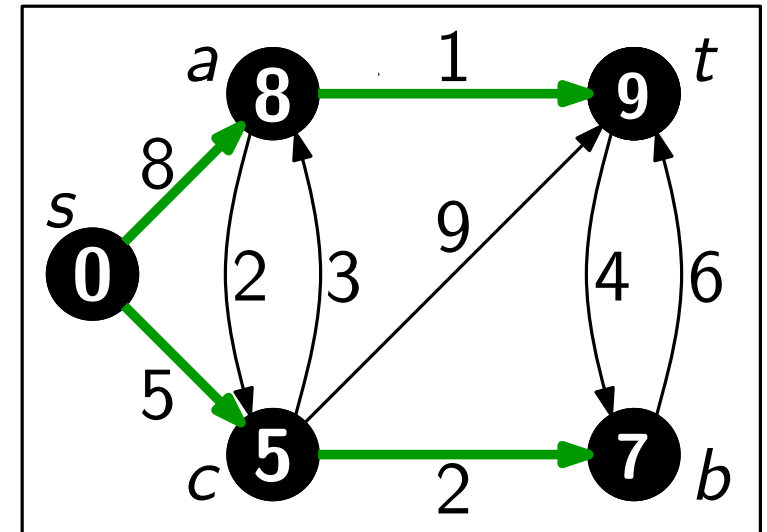
while not $Q.\text{Empty}()$ **do**

$u = Q.\text{ExtractMin}()$

foreach $v \in \text{Adj}[u]$ **do**

 Relax($u, v; w$) (if $u.\text{color} \neq \text{black}$)

$u.\text{color} = \text{black}$



Relax($u, v; w$)

if $v.d > u.d + w(u, v)$ **then**

$v.\text{color} = \text{gray}$

$v.d = u.d + w(u, v)$

$v.\pi = u$

$Q.\text{UpdateKey}(v, v.d)$

Initialize(Graph G , Vertex s)

foreach $u \in V$ **do**

$u.\text{color} = \text{white}$

$u.d = \infty$

$u.\pi = \text{nil}$

$s.\text{color} = \text{gray}$

$s.d = 0$

Dijkstra – Pseudocode

Laufzeit?

Dijkstra(WeightedGraph G , Vertex s)

Initialize(G, s)

$Q = \text{new PriorityQueue}(V, d)$

while not $Q.\text{Empty}()$ **do**

$u = Q.\text{ExtractMin}()$

foreach $v \in \text{Adj}[u]$ **do**

 Relax($u, v; w$)

$u.\text{color} = \text{black}$

$O(V)$ Zeit

Genau $|V|$ mal.

Für jeden Knoten $u \in V$
genau $|\text{Adj}[u]|$ ($= \text{deg } u$)
mal, also insg. $2|E|$ mal.

Relax($u, v; w$)

if $v.d > u.d + w(u, v)$ **then**

$v.\text{color} = \text{gray}$

$v.d = u.d + w(u, v)$

$v.\pi = u$

$Q.\text{UpdateKey}(v, v.d)$

Also wird UpdateKey
 $\leq 2|E|$ mal aufgerufen.

Gesamt: implementierungsabh.
beste: $O(E + V \log V)$

Übersicht

3. Minimale Spann bäume

3.1 Prim

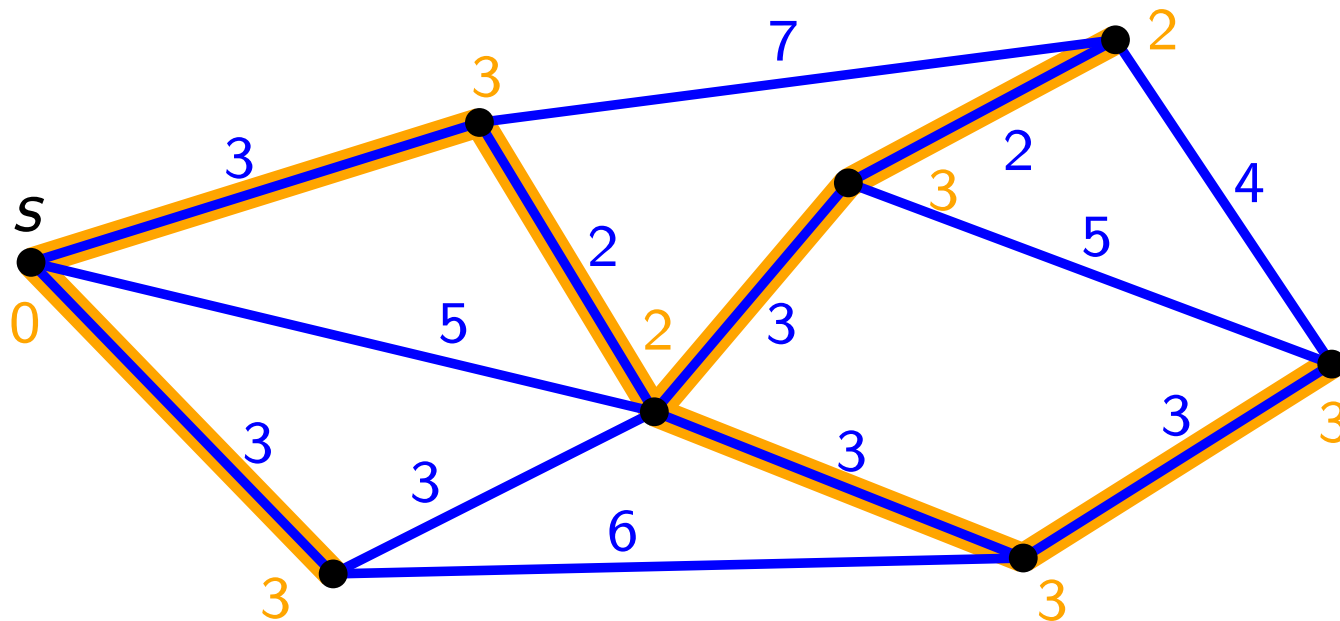
Beispiel

Pseudocode

Prim – Beispiel

Eingabe: ungerichteter, zusammenhängender Graph $G(V, E)$ mit Kantengewichten und Knoten s

Ausgabe: gewichtsminimaler Baum T , der G aufspannt



Prim – Pseudocode

JarníkPrimMST ↪ Undirected

~~Dijkstra~~(WeightedGraph $G = (V, E; w)$, Vertex s)

Initialize(G, s)

$Q = \text{new PriorityQueue}(V, d)$

while not $Q.\text{Empty}()$ **do**

$u = Q.\text{ExtractMin}()$

foreach $v \in \text{Adj}[u]$ **do**

$\lfloor \text{Relax}'(u, v; w)$

$v \in Q$ and ...

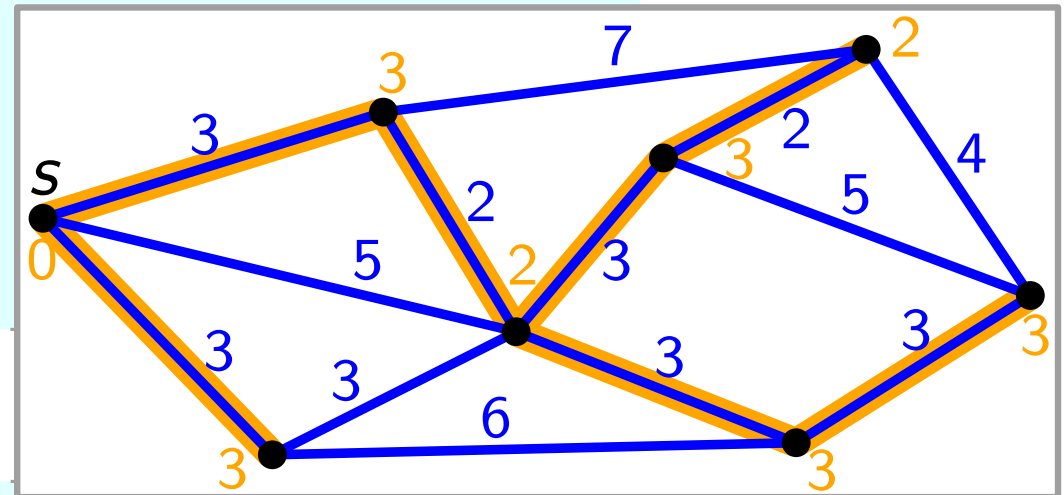
$\text{Relax}'(u, v; w)$

if $v.d > \cancel{u.d} + w(u, v)$ **then**

$v.d = \cancel{u.d} + w(u, v)$

$v.\pi = u$

$Q.\text{UpdateKey}(v, v.d)$



Laufzeit?

→ s. Dijkstra

→ $O(E + V \log V)$

Übersicht

3. Minimale Spann bäume

3.1 Prim

3.2 Kruskal

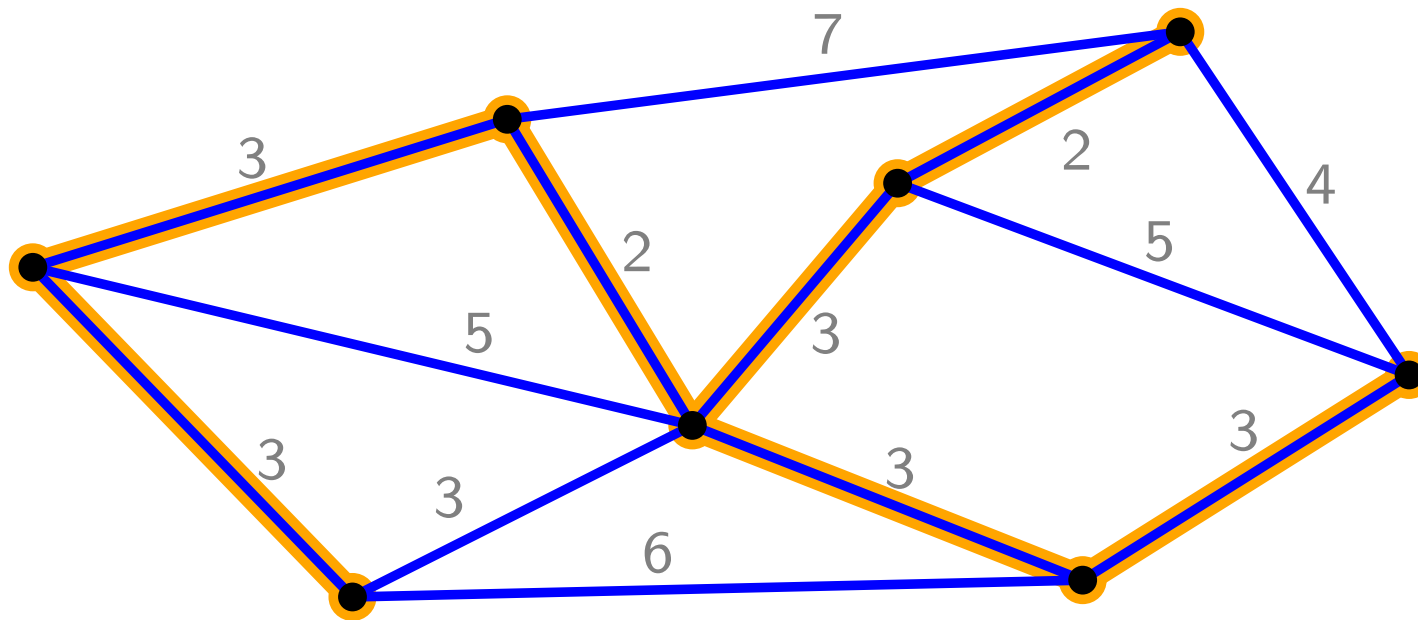
Beispiel

Pseudocode

Kruskal – Beispiel

Eingabe: ungerichteter, zusammenhängender Graph $G(V, E)$
mit Kantengewichten

Ausgabe: gewichtsminimaler Baum T , der G aufspannt



Kruskal – Pseudocode

KruskalMST(WeightedUndirectedGraph $G = (V, E; w)$)

$A = \emptyset$

foreach $v \in V$ **do**

└─ MakeSet(v)

Sortiere E nicht-absteigend nach Gewicht w

foreach $uv \in E$ **do**

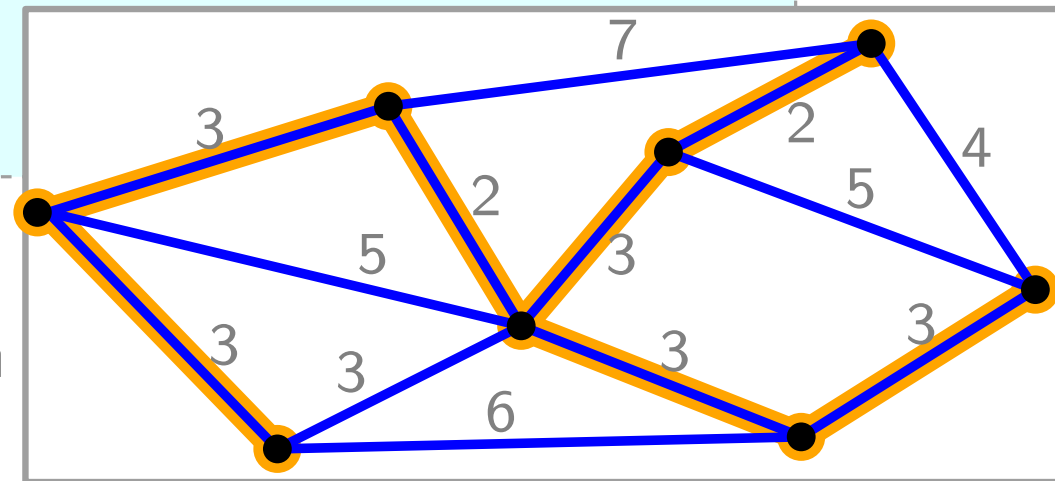
┌─ **if** FindSet(u) \neq FindSet(v) **then**

└─ $A = A \cup \{uv\}$

└─ Union(u, v)

Laufzeit?

$|V| \cdot \text{MakeSet} + (|V| - 1) \cdot \text{Union}$
 $+ 2|E| \cdot \text{FindSet} + \text{Sort}(E)$
 $= O(E \log V)$



vgl. ADS-Skript

Viel Erfolg...

... beim Lösen des
1. Übungsblattes