Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und
wissensbasierte Systeme

Institut für Informatik

Algorithmen und Datenstrukturen

Wintersemester 2018/19

21. Vorlesung

Minimale Spannbäume

Prof. Dr. Alexander Wolff

Lehrstuhl für Informatik I

Motivation

*) Kantengewichte $w\colon E \to \mathbb{R}_{>0}$

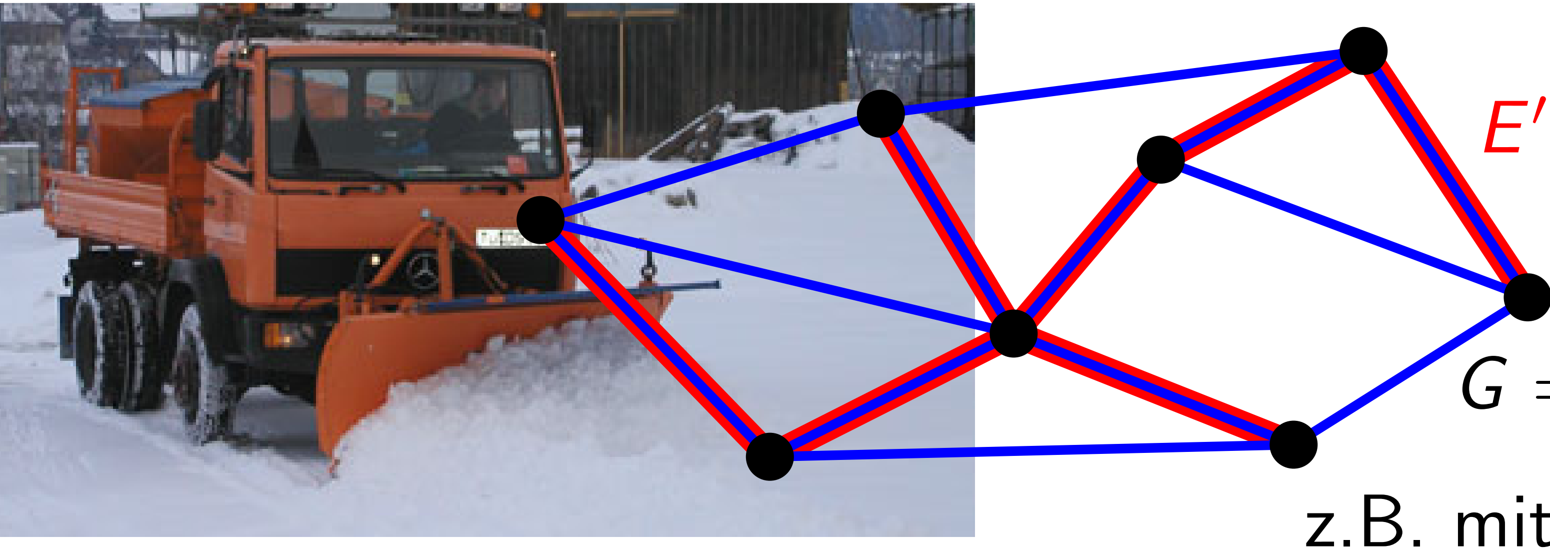
**) $w(E') := \sum_{e \in E'} w(e)$

Gegeben: zusammenhängendes Straßennetz $G = (V, E; w^{\star})$, das eine Menge V von n Städten verbindet.

Gesucht: Teilnetz $G' = (V, E')$ mit $E' \subseteq E$, so dass

(1) man von jeder Stadt in G' zu jeder anderen kommen kann („G' spannt G auf“) und

(2) die „Schneeräumkosten“ $w(E')^{\star\star}$ minimal sind unter allen Teilnetzen, die (1) erfüllen.



z.B. mit $w \equiv \begin{cases} \text{euklid.} \\ \text{Abstände} \end{cases}$

Beobachtung

Wegen der Minimalität von $w(E')$ gilt:

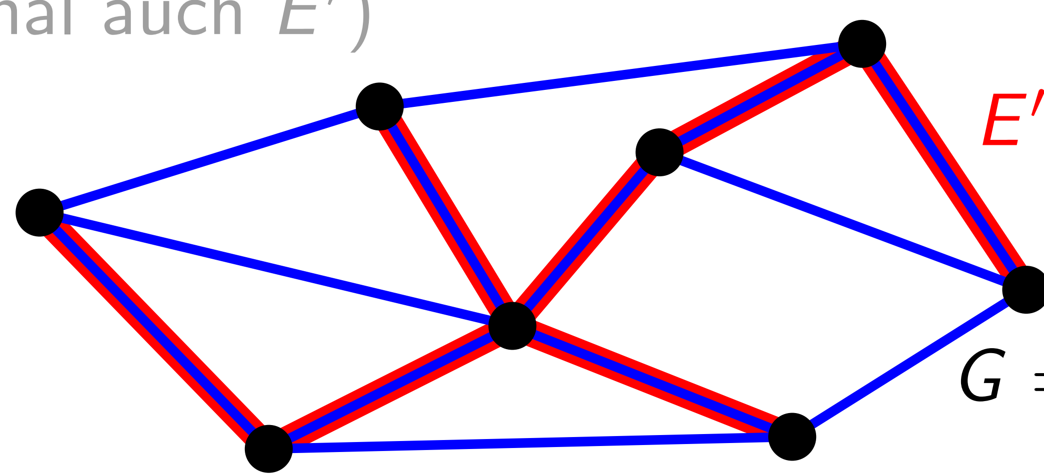
G' hat keine Kreise $\Rightarrow$ G' ist ein Wald.

G' „erbt“ Zusammenhang von G $\Rightarrow$ G' Baum.

G' spannt G auf $\Rightarrow$ G' ist Spannbaum von G.

G' hat minimales Gewicht unter *allen* Spannbäumen von G.

Wir nennen $\underbrace{G'\text{ kurz}}_{\text{(manchmal auch } E')}$ *minimalen Spannbaum* von G.



$E' \subseteq E$

$G = (V, E; w)$

z.B. mit $w \equiv \begin{cases} \text{euklid.} \\ \text{Abstände} \end{cases}$

Beob. $|E'| = |V| - 1$

Generischer Min.-Spannbaum-Algorithmus

GenericMST(UndirectedConnectedGraph G, EdgeWeights w)

$A = \emptyset$

while $|A| < |V| - 1$ **do**

// *Invariante:* A ist Teilmenge eines min. Spannbaums von G

finde Kante uv, die *sicher* für A ist

$A = A \cup \{uv\}$

return A

Wir sagen uv ist *sicher* für A, falls Invariante für $A \cup \{uv\}$ gilt.

Beob. Dies ist ein sogenannter *Greedy-Algorithmus*!

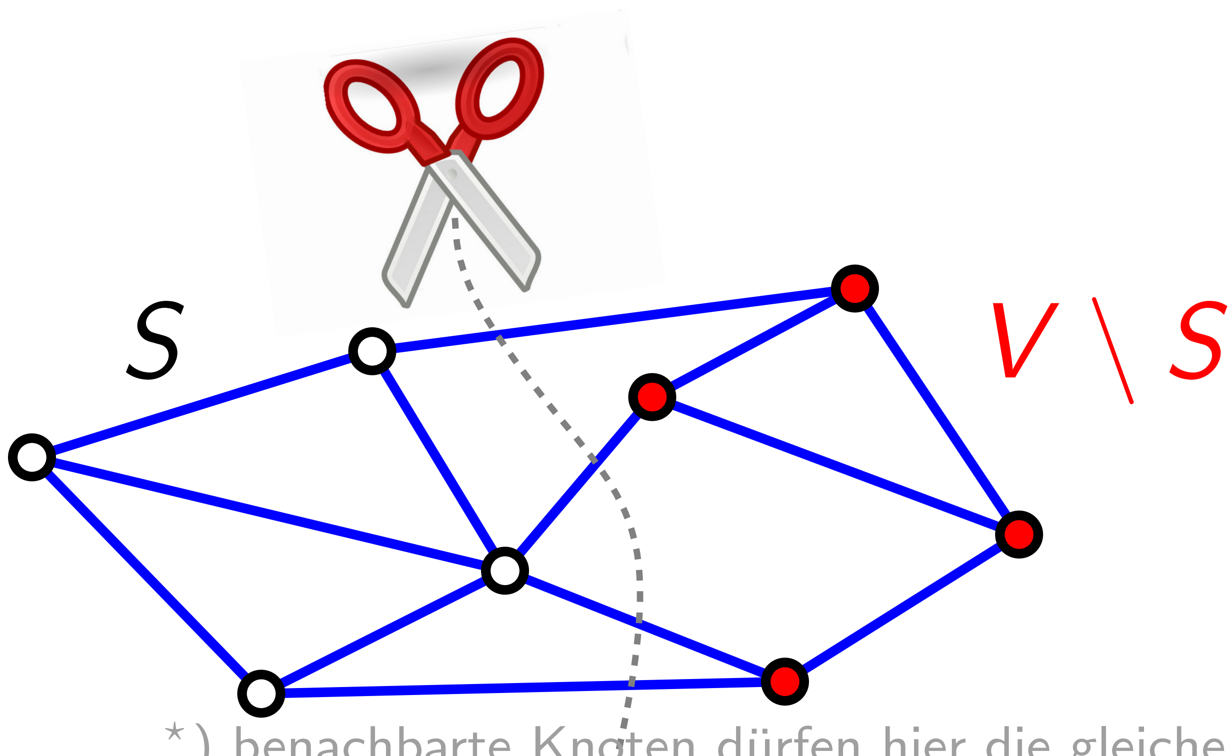
Frage: Gibt's überhaupt immer eine sichere Kante?

Antwort: Ja! – *Per Induktion!*

Frage: Aber wie findet man eine – ohne schon einen minimalen Spannbaum zu kennen?

Schnitte und leichte Kanten

Def. Ein *Schnitt* $(S, V \setminus S)$ eines ungerichteten Graphen $G = (V, E)$ ist eine Zerlegung (od. Zweifärbung*) von V.

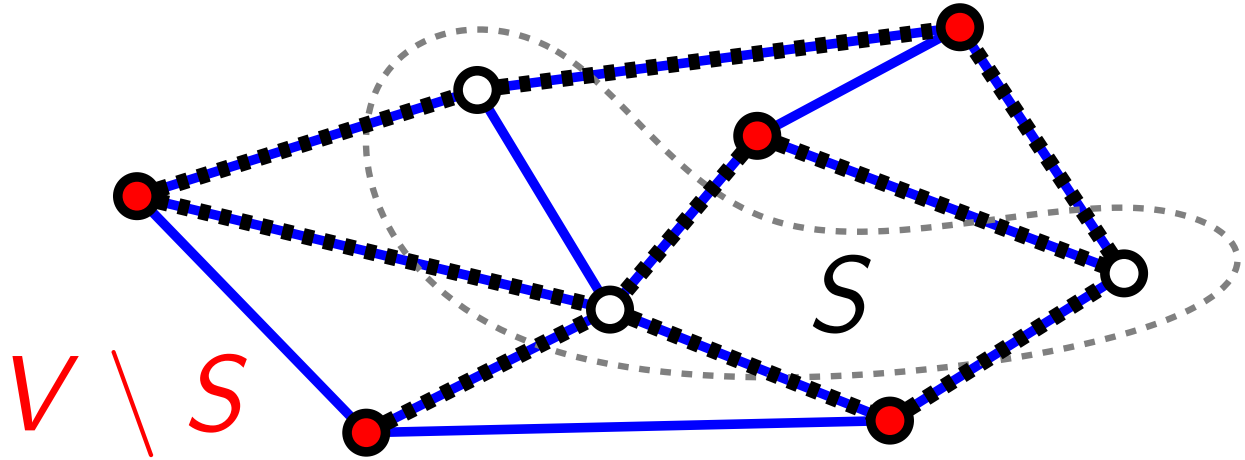


*) benachbarte Knoten dürfen hier die gleiche Farbe haben.

Schnitte und leichte Kanten

Def. Ein *Schnitt* $(S, V \setminus S)$ eines ungerichteten Graphen $G = (V, E)$ ist eine Zerlegung (od. Zweifärbung*) von V.

Eine Kante e *kreuzt* $(S, V \setminus S)$, wenn ein Endpunkt von e in S und der andere in $V \setminus S$ liegt.

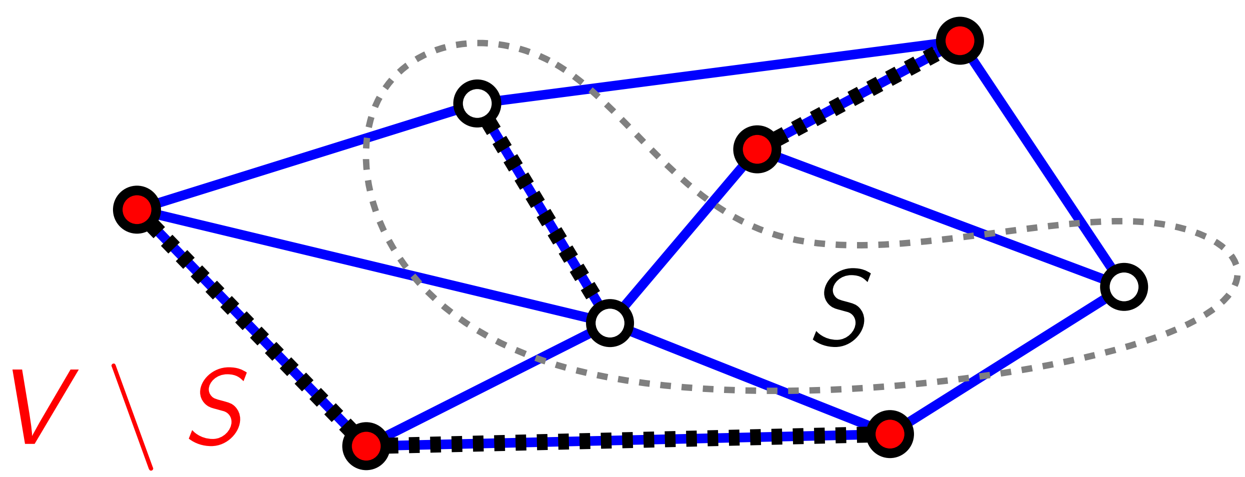


Schnitte und leichte Kanten

Def. Ein *Schnitt* $(S, V \setminus S)$ eines ungerichteten Graphen $G = (V, E)$ ist eine Zerlegung (od. Zweifärbung*) von V.

Eine Kante e *kreuzt* $(S, V \setminus S)$, wenn ein Endpunkt von e in S und der andere in $V \setminus S$ liegt.

Ein Schnitt *respektiert* eine Kantenmenge A, wenn keine Kante in A den Schnitt kreuzt.



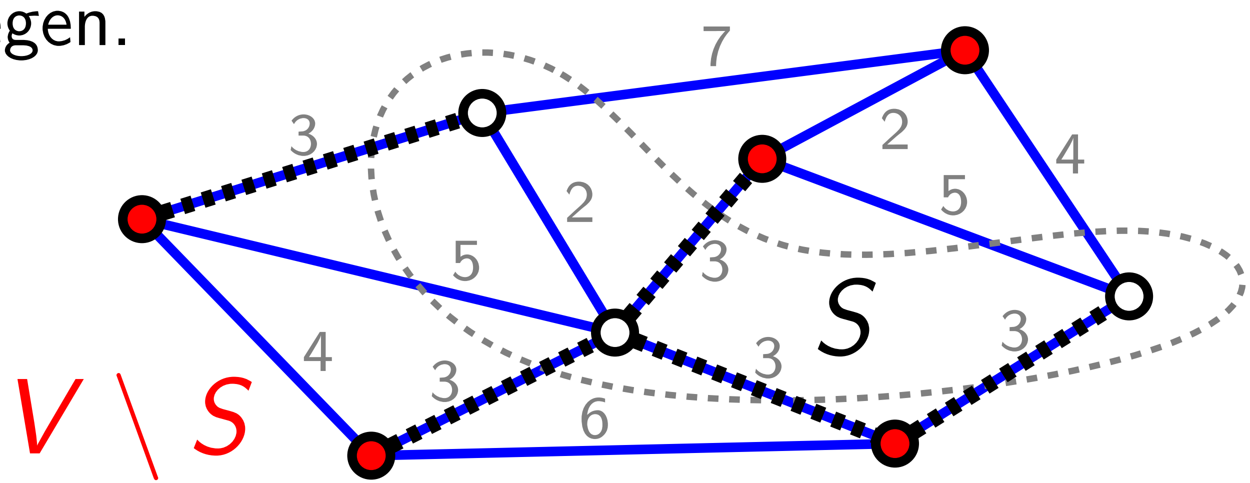
Schnitte und leichte Kanten

Def. Ein *Schnitt* $(S, V \setminus S)$ eines ungerichteten Graphen $G = (V, E)$ ist eine Zerlegung (od. Zweifärbung*) von V.

Eine Kante e *kreuzt* $(S, V \setminus S)$, wenn ein Endpunkt von e in S und der andere in $V \setminus S$ liegt.

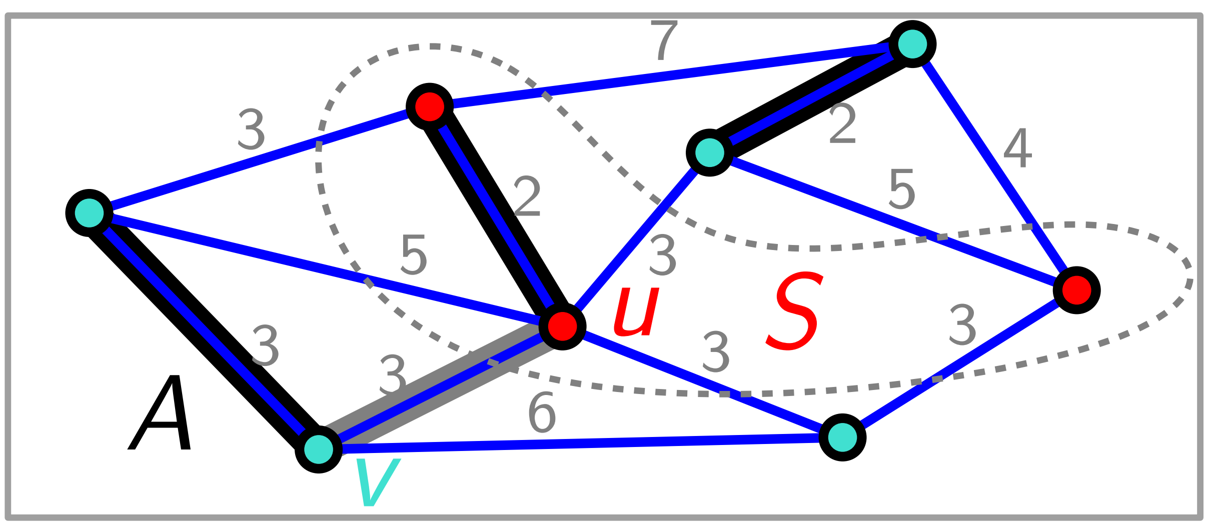
Ein Schnitt *respektiert* eine Kantenmenge A, wenn keine Kante in A den Schnitt kreuzt.

Eine Kante e, die einen Schnitt kreuzt, ist *leicht*, wenn alle Kanten, die den Schnitt kreuzen, mindestens $w(e)$ wiegen.



Erweiterungssatz

Satz. Sei $G = (V, E; w)$ ein zshg., gewichteter, unger. Graph.
Sei T Kantenmenge eines min. Spannbaums von G.
Sei **A** Teilmenge von T.
Sei $(S, V \setminus S)$ ein Schnitt, der **A** respektiert.
Sei $uv \in E$ leicht bzgl. $(S, V \setminus S)$.
Dann ist uv sicher für **A**,
d.h. G hat einen min. Spannbaum, der $A \cup \{uv\}$ enthält.

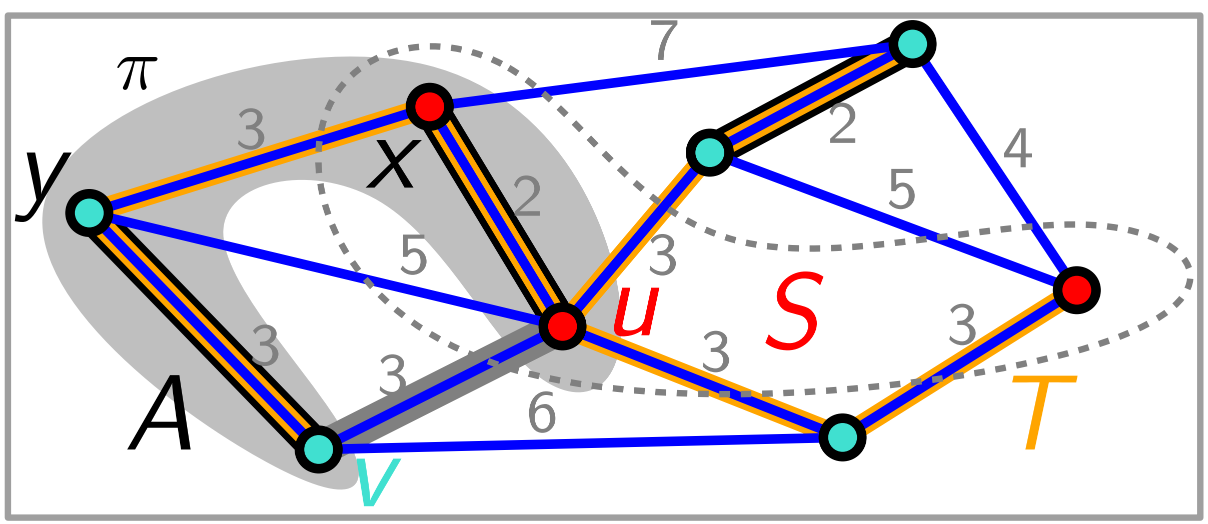


Beweis

Satz. … Dann ist uv sicher für A.

Beweis. *Zeige:* G hat min. Spannbaum, der $A \cup \{uv\}$ enthält.
Falls $uv \in T$, fertig. Also $uv \notin T$. Sei π u-v-Pfad in T.
$\Rightarrow \pi + uv$ ist Kreis (wobei uv $(S, V \setminus S)$ kreuzt)
$\Rightarrow$ Kreis enthält zweite Kante xy, die $(S, V \setminus S)$ kreuzt.
$\Rightarrow T' = (T \cup \{uv\}) \setminus \{xy\}$ ist auch Spannbaum von G.

$$w(T') = w(T) + \underbrace{w(uv) - w(xy)}_{\leq 0\text{, da } uv \text{ leicht bzgl. } (S, V \setminus S)} \leq w(T)$$



$\Rightarrow T'$ ist *minimaler* Spannbaum von G.

Und: $A \cup \{uv\} \subseteq T'$.

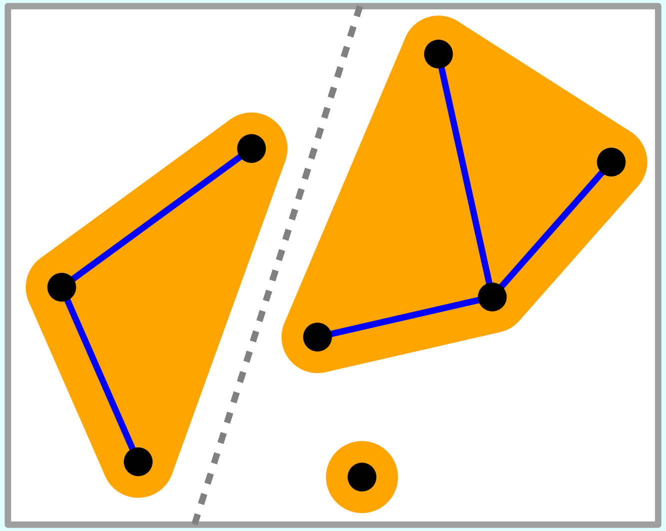
$\Rightarrow uv$ ist sicher für A. $\square$

Zurück zum Algorithmus

Satz. Sei $G = (V, E; w)$ ein zshg., gewichteter, unger. Graph.
Sei T Kantenmenge eines min. Spannbaums von G.
Sei A Teilmenge von T.
Sei $(S, V \setminus S)$ ein Schnitt, der A respektiert.
Sei $uv \in E$ leicht bzgl. $(S, V \setminus S)$.
Dann ist uv sicher für A.

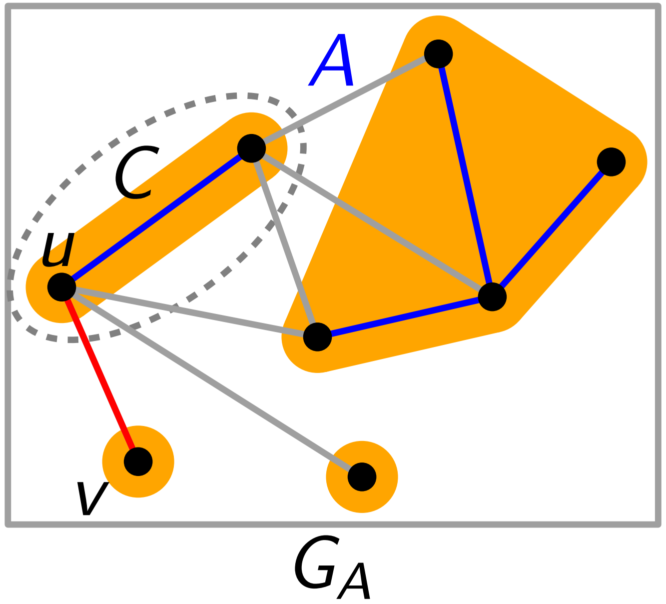
GenericMST(UndirectedConnectedGraph G, EdgeWeights w)

$A = \emptyset$
while $|A| < |V| - 1$ **do**
    // *INV:* $A \subseteq$ min. Spannbaum von G
    finde Kante uv, die *sicher* für A ist
    $A = A \cup \{uv\}$
return A



Zusammenhangskomponenten

Def. Eine *Zusammenhangs-komponente* eines Graphen ist ein Teilgraph, der von einer nicht vergrößerbaren („*inklusionsmaximalen*“) zusammenhängenden Menge von Knoten *induziert* wird.



Korollar. $G = (V, E)$ wie gehabt.
$A \subseteq E$ in einem min. Spannbaum von G enthalten.
$C = (V_C, E_C)$ Zshgskomp. des Waldes $G_A = (V, A)$.
uv leicht bzgl. $(V_C, V \setminus V_C)$

Dann gilt: uv ist sicher für A.

Der Algorithmus von Jarník-Prim (1930/1957)

JarníkPrimMST Undirected

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

  Initialize(G, s)

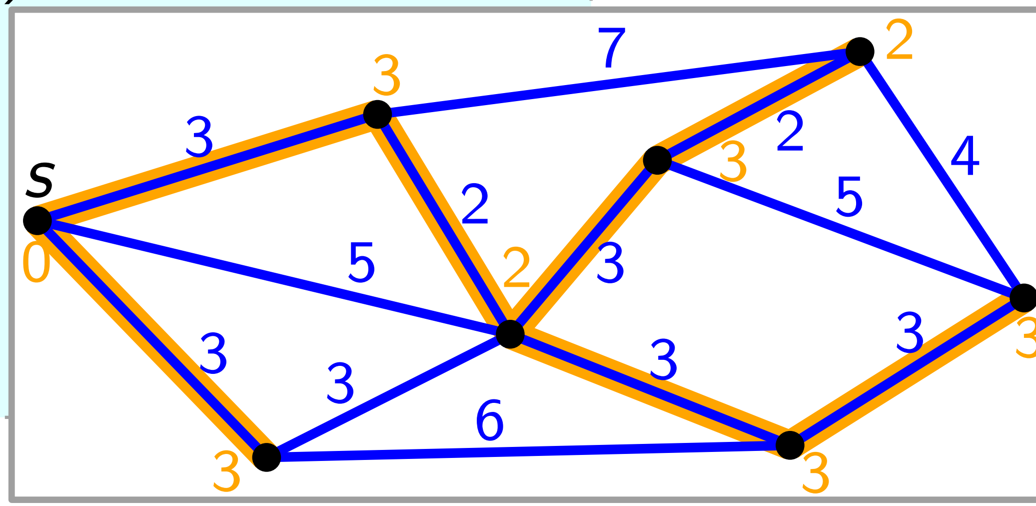
  $Q =$ **new** PriorityQueue(V, d) // Gewichtung

  while not Q.Empty() **do**

    $u = Q$.ExtractMin()

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

      Relax′($u, v; w$)



$v \in Q$ **and** ...

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.DecreaseKey($v, v.d$)

Korrektheit? ✓

Folgt aus Korollar: $A = \{\{u, u.\pi\} : u \notin Q\}$, Kante $\{u, u.\pi\}$ immer sicher bzgl. $(Q^\star, V \setminus Q^\star)$, wobei $Q^\star = Q \cup \{u\}$.

Laufzeit?

$O(|E| \cdot \mathrm{DecreaseKey} + |V| \cdot \mathrm{ExtractMin})$

$\Rightarrow O((E + V) \log V)$ [Heap/RS-Baum]

$\Rightarrow O(E + V \log V)$ [Fibonacci-Heap]

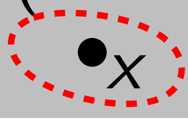
Einschub: halbdynamische Mengen

(wachsen nur, schrumpfen nicht)

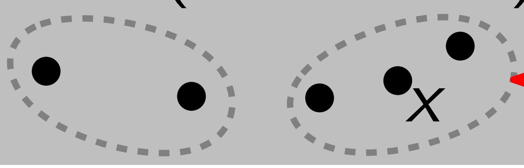
Die halbdyn. Mengen zerlegen immer eine Grundmenge X.

Drei Operationen:

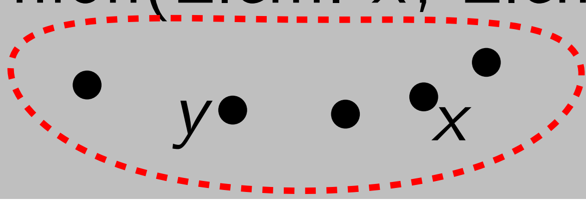
MakeSet(Element x) — legt die Menge $\{x\}$ an.



FindSet(Element x) — liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



Union(Elem. x, Elem. y) — vereinigt die Mengen, die momentan x und y enthalten.



Satz. Eine Folge von m MakeSet-, Union- und FindSet-Oper., von denen n MakeSet-Oper. sind, benötigt $O(m \cdot \alpha(n))$ Zeit, wobei $\alpha(n) \leq 4$ für alle $n \leq 10^{80}$. Insbesondere $\alpha(n) \ll \log_{10} n$ für $n > 1$.

funktionales Inverses der extrem schnellwachsenden Ackermann-Funktion $A(n, n)$

Der Algorithmus von Kruskal

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)

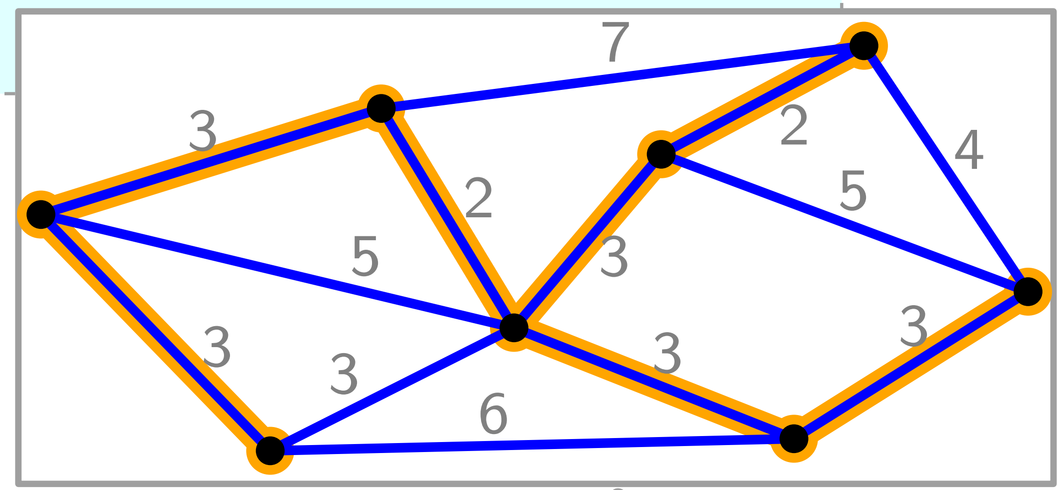
Laufzeit?

$|V|\cdot$MakeSet $+ (|V| - 1)\cdot$Union

$+ 2|E|\cdot$FindSet $+$ Sort(E)

$\in O(E \log V + E \log E)$

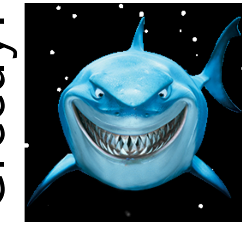
$= O(E \log V)$! Warum??



Weil $O(\log E) \subseteq O(\log V^2) = O(\log V)$. Ah...

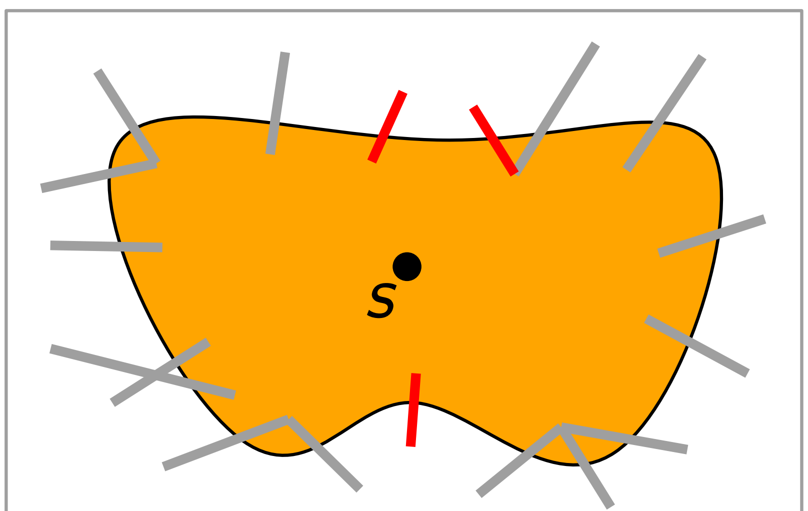
Übersicht: Algo. für min. Spannbäume

Greedy!



Jarník-Prim

- geht (wie Dijkstra / BFS) wellenförmig von einem Startknoten aus
- aktuelle Kantenmenge zusammenhängend
- Laufzeit $O(E + V \log V)$



Kruskal

- bearbeitet Kanten nach aufsteigendem (genauer: nicht-absteig.) Gewicht
- nach Einfügen der i. Kante gibt es $n - i$ Zshgskomp.
- Laufzeit $O(E \log V)$