

Informatik II - SS 2014

(Algorithmen & Datenstrukturen)

Vorlesung 16 (2.7.2014)

Graphtraversierung II,
Minimale Spannäume I



**UNI
FREIBURG**

Fabian Kuhn

Algorithmen und Komplexität

Tiefensuche: Pseudocode

DFS-Traversal:

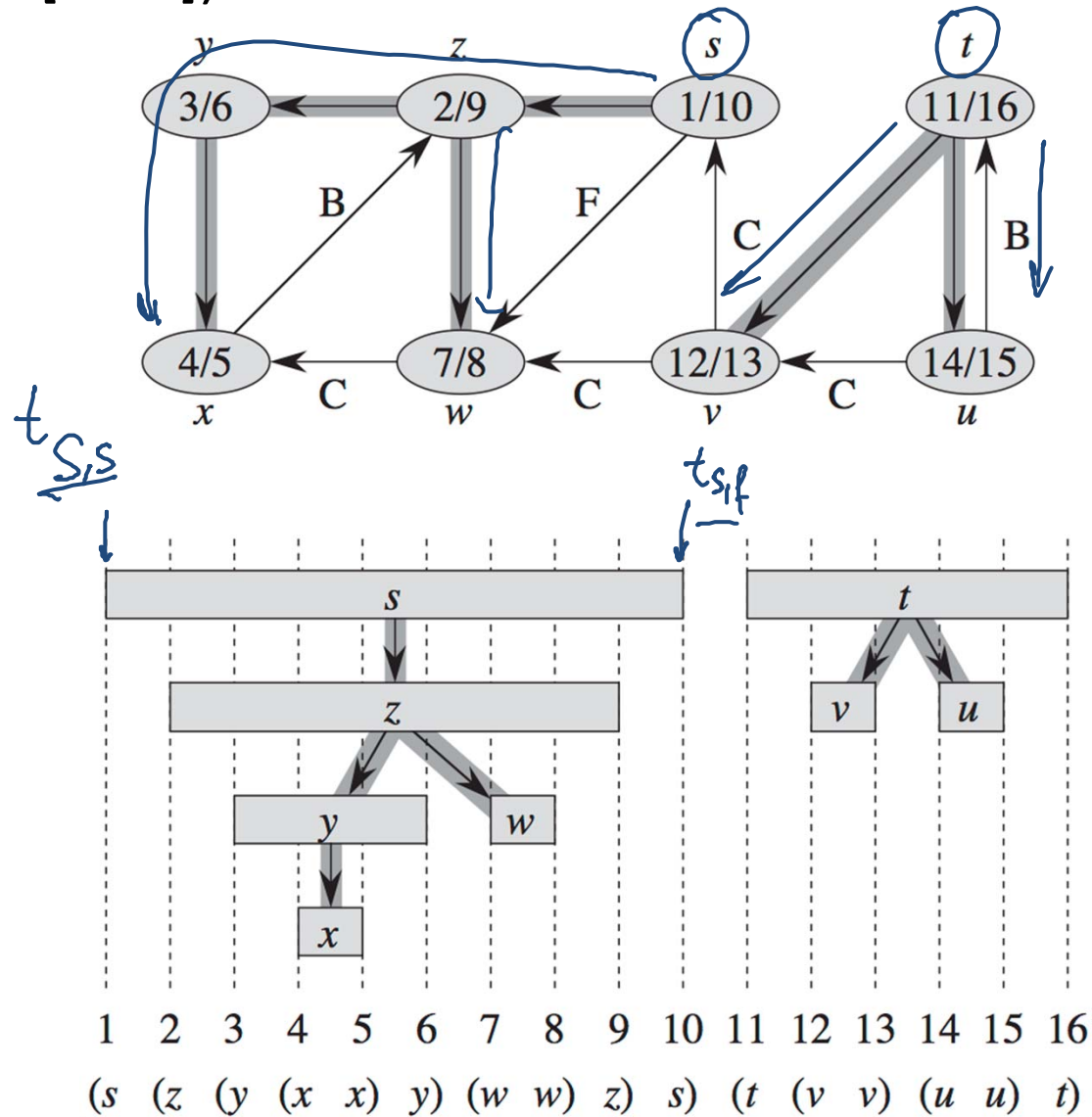
```
for all u in V: u.color = white;
DFS-visit(root, NULL)
```

DFS-visit(u, p):

```
u.color = gray;
u.parent = p;
for all v in u.neighbors do
    if v.color = white
        DFS-visit(v, u)
visit node u;
u.color = black;
```

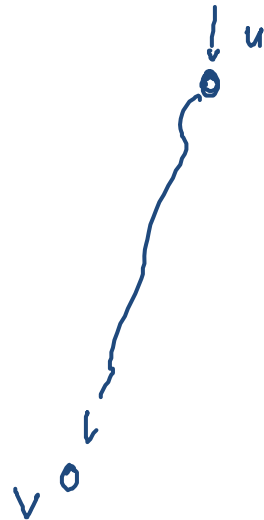
DFS- "Klammer"-Theorem

Beispiel (aus [CLRS])



Weißer Pfade

Theorem: In einem DFS-Baum ist ein Knoten v genau dann im Teilbaum eines Knoten u , falls unmittelbar vor dem Markieren von u , ein komplett weißer Pfad von u nach v besteht.



Klassifizierung der Kanten (bei DFS-Suche)

Baumkanten:

- (u, v) ist eine Baumkante, falls v von u aus entdeckt wird

v weiss

Rückwärtskanten:

- (u, v) ist eine Rückwärtskante, falls v eine Vorgängerknoten von u ist

v grau

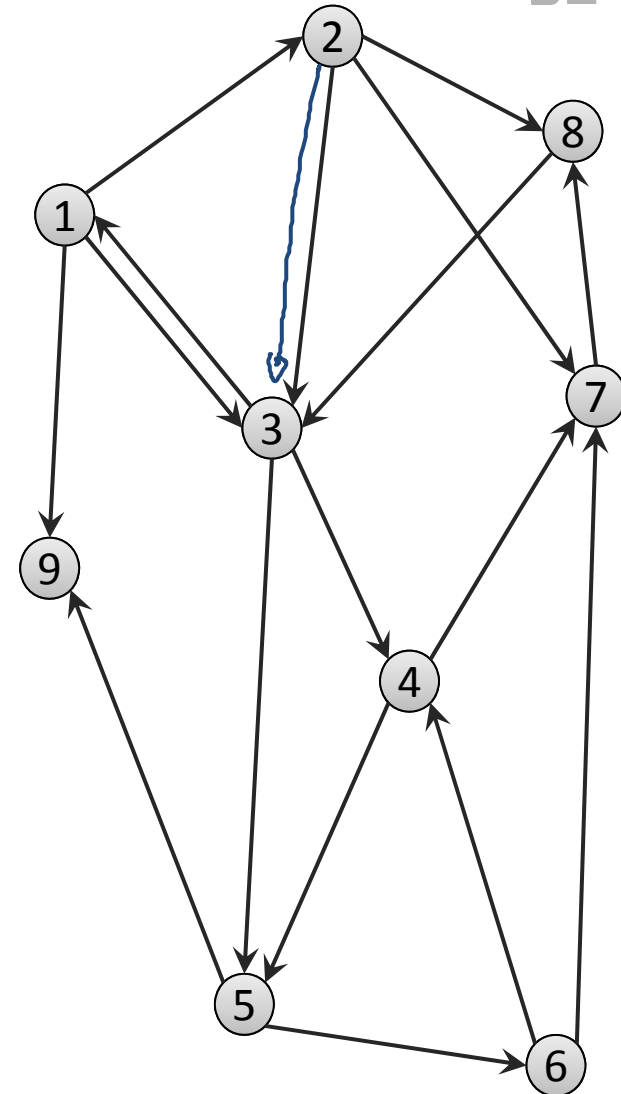
Vorwärtskanten:

- (u, v) ist eine Vorwärtskante, falls v ein Nachfolgerknoten von u ist

v schwarz

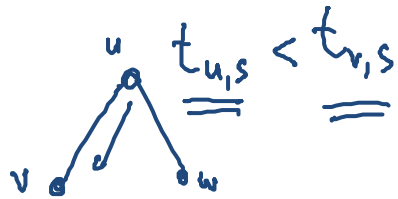
Querkanten:

- Alle übrigen Kanten



DFS – Ungerichtete Graphen

Theorem: Bei einer DFS-Suche in ungerichteten Graphen ist jede Kante entweder eine Baumkante oder eine Rückwärtskante.



DFS-visit (u)

$\left[\begin{array}{l} u \text{ wird grau} \\ \text{gehe durch Nachb. von } u \\ u \text{ wird schwarz} \end{array} \right]$

$\{u, v\}$ zuerst "von u aus" betrachtet:

v weiss $\rightarrow \{u, v\}$ Baumkante

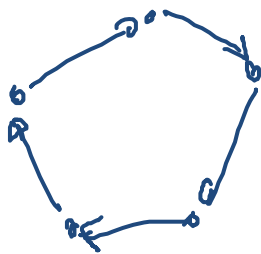
$\{u, v\}$ zuerst "von v aus" betrachtet:

u grau $\rightarrow \{u, v\}$ Rückwärtskante

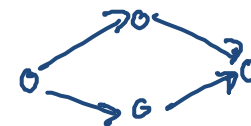
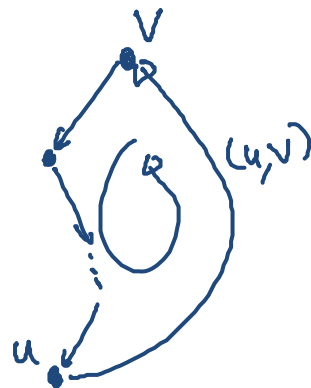
DFS – Gerichtete Graphen

Theorem: Ein gerichteter Graph hat genau dann keine Zyklen, falls es bei der DFS-Suche keine Rückwärtskanten gibt.

Zyklus



Rückwärtskante $(u,v) \rightarrow$ Zyklus

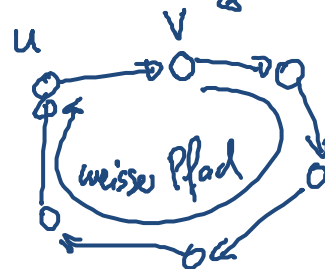


Implikation:

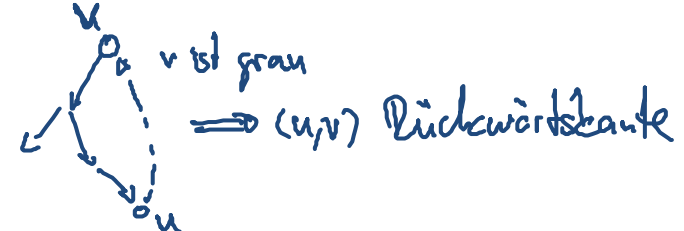
Man kann in $O(m+n)$ erkennen, ob ein gegebener ger. Graph zyklenfrei ist.

Zyklus \rightarrow Rückwärtskante (u,v)

erster betrachteter Knoten im Zyklus



\Rightarrow u ist im Teilbaum von v

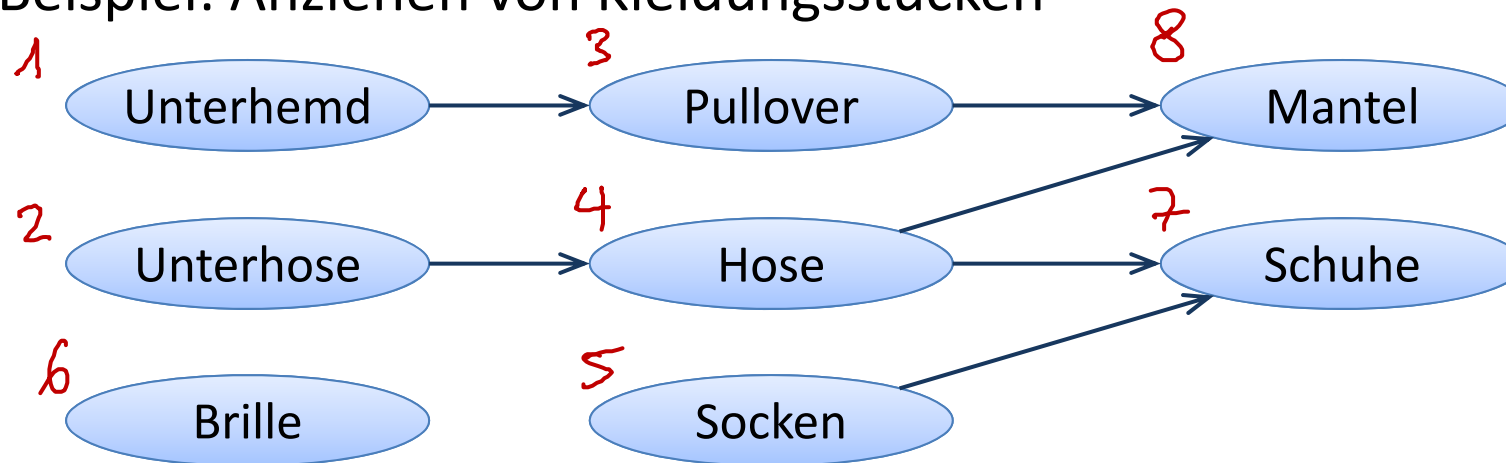


\Rightarrow (u,v) Rückwärtskante

Anwendung: Topologische Sortierung

Zyklenfreie, gerichtete Graphen:

- **DAG**: directed acyclic graph
- Modellieren z.B. zeitliche Abhängigkeiten von Aufgaben
- Beispiel: Anziehen von Kleidungsstücken



Topologische Sortierung:

- Sortiere die Knoten eines DAGs so, dass u vor v erscheint, falls ein gerichteter Pfad von u nach v existiert
- Im Beispiel: Finde eine mögliche Anziehreihenfolge

Topologische Sortierung: Etwas formaler...

Zyklenfreie, gerichtete Graphen:

- repräsentieren partielle Ordnungsrelationen
 - asymmetrisch: $a < b \implies \neg(b < a)$
 - transitiv: $a < b \wedge b < c \implies a < c$
 - partielle Ordnung: nicht alle Paare müssen vergleichbar sein

- Beispiel: Teilmengenrelation bei Mengen

$$\{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{1,2,3\}$$
$$\{1\} \subset \{1,2\} \subset \{1,2,3\} \quad \{1\} \subset \{1,3\} \subset \{1,2,3\}$$

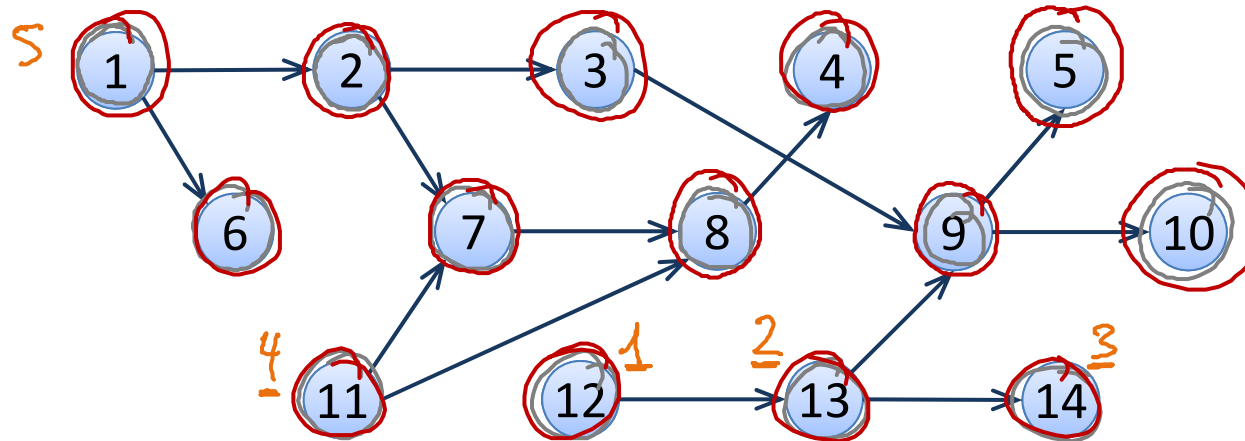
Topologische Sortierung:

- Sortiere die Knoten eines DAGs so, dass u vor v erscheint, falls ein gerichteter Pfad von u nach v existiert
- Erweitere eine partielle Ordnung zu einer totalen Ordnung

$$\{1\} < \{2\} < \{3\} < \{1,2\} < \{1,3\} < \{1,2,3\}$$

Topologische Sortierung: Algorithmus

Führe DFS aus...



$t_{1,s}, t_{2,s}, t_{3,s}, t_{9,s}, t_{5,s}, t_{5,f}, t_{10,s}, t_{10,f}, t_{9,f}, t_{3,f}, t_{7,s}, t_{8,s}, t_{4,s}, t_{4,f},$
 $t_{8,f}, t_{7,8}, t_{2,f}, t_{6,s}, t_{6,f}, t_{1,f}, t_{11,s}, t_{11,f}, t_{12,s}, t_{13,s}, t_{14,s}, t_{14,f}, t_{13,f}, t_{12,f}$

Beobachtung:


- Knoten ohne Nachfolger werden als erstes besucht (schwarz gef.)
- Besuchreihenfolge ist umgekehrte topologische Sortierung

Topologische Sortierung: Algorithmus

Theorem: Umgekehrte "Visit"-Reihenfolge (schwarz färben) der Knoten bei DFS-Traversierung ergibt topologische Sortierung

$u \rightarrow v$ Umgekehrt nach $t_{v,f}$ sortieren ergibt top. Sortierung
Zu zeigen $t_{v,f} < t_{u,f}$

$u \rightarrow v$ wenn DFS die Kante betrachtet $\rightarrow v$ ist weiss oder schwarz

v weiss: (u,v) wird Baumkante  v wird schwarz bevor wir zurückkehren (zur u)
 $t_{v,f} < t_{u,f}$ (Klammer-Thm.)

v schwarz: u ist grau
 $\hookrightarrow t_{v,f} < t_{u,f}$



Topologische Sortierung: Algorithmus

Theorem: Umgekehrte “Visit”-Reihenfolge (schwarz färben) der Knoten bei DFS-Traversierung ergibt topologische Sortierung

Stark zusammenhängende Komponenten

- Stark zus.-hängende Komponente eines gerichteten Graphen:
“Maximale Knoten-Teilmenge, so dass jeder jeden erreichen kann”

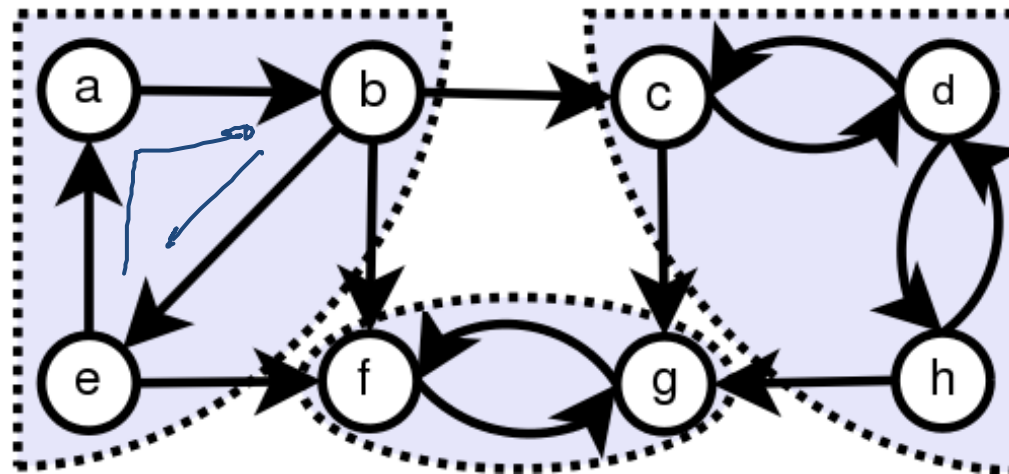


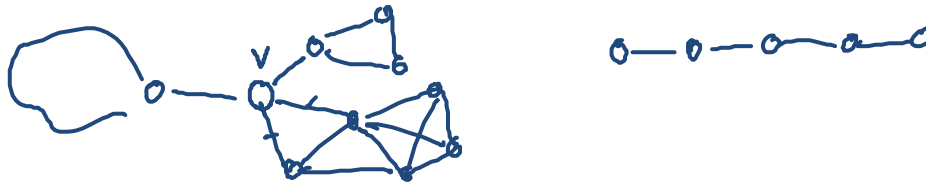
Bild: Wikipedia

- Benötigt 2 DFS-Traversierungen (Zeit $\in O(m + n)$)
 - auf G und auf G^T (alle Kanten umgedreht)
 - G und G^T haben die gleichen stark zus.-hängenden Komponenten
- Details z.B. in [CLRS]

DFS-Traversierung: Weitere Anwendungen

Artikulationsknoten, Brücken, Biconnected Components

- Annahme: ungerichteter Graph
- **Artikulationsknoten v :**
 v entfernen vergrößert die Anzahl Komponenten



- **Brücke e :**
Kante e entfernen vergrößert die Anzahl Komponenten



Biconnected Components

- Komponenten, welche übrig bleiben, wenn man alle Brücken entfernt

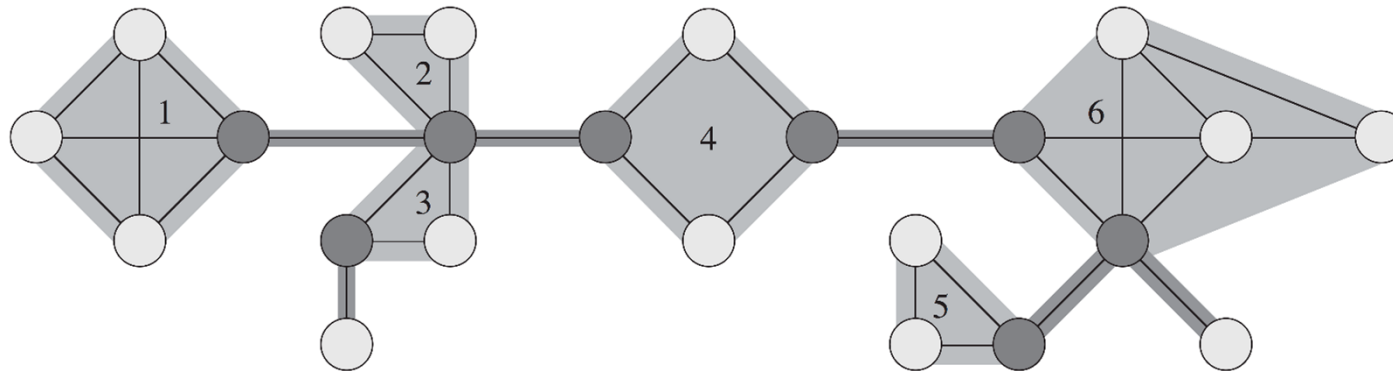


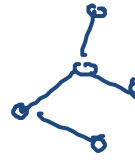
Bild: [CLRS]

- Artikulationsknoten und Brücken können mit einer DFS-Traversierung in $O(m + n)$ Zeit gefunden werden
 - Algorithmus von Hopcroft, Tarjan (1973)
- Zerlegung in Biconnected Components daher in der gleichen Zeit

Bäume

- Als ungerichtete Graphen (mit n Knoten) betrachtet...

Baum:



- Zusammenhängender ungerichteter Graph, ohne Zyklen
 - Ein nicht zus.-hängender zyklensfreier (unger.) Graph heisst Wald
 - Anzahl Kanten: $n - 1$ (jede Kante reduziert die #Komponenten um 1)

minimal: Kante entfernen \rightarrow nicht zus.-h.

Äquivalente Definitionen:

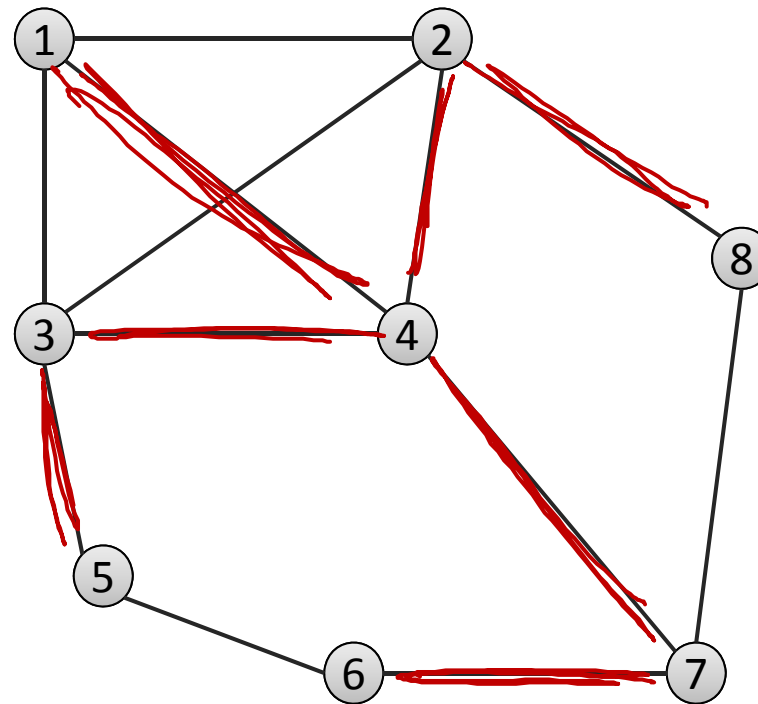
- Minimaler zusammenhängender Graph
- Maximaler zyklensfreier Graph
- Eindeutiger Pfad zwischen jedem Knotenpaar
- Zusammenhängender Graph mit $n - 1$ Kanten

Spannbaum

Gegeben: Zusammenhängender, ungerichteter Graph $G = (V, E)$

Spannbaum $T = (V, E_T)$: Teilgraph ($E_T \subseteq E$)

- T ist ein Baum, welcher alle Knoten von G enthält
- Alternativ: T ist ein Baum mit $n - 1$ Kanten aus E



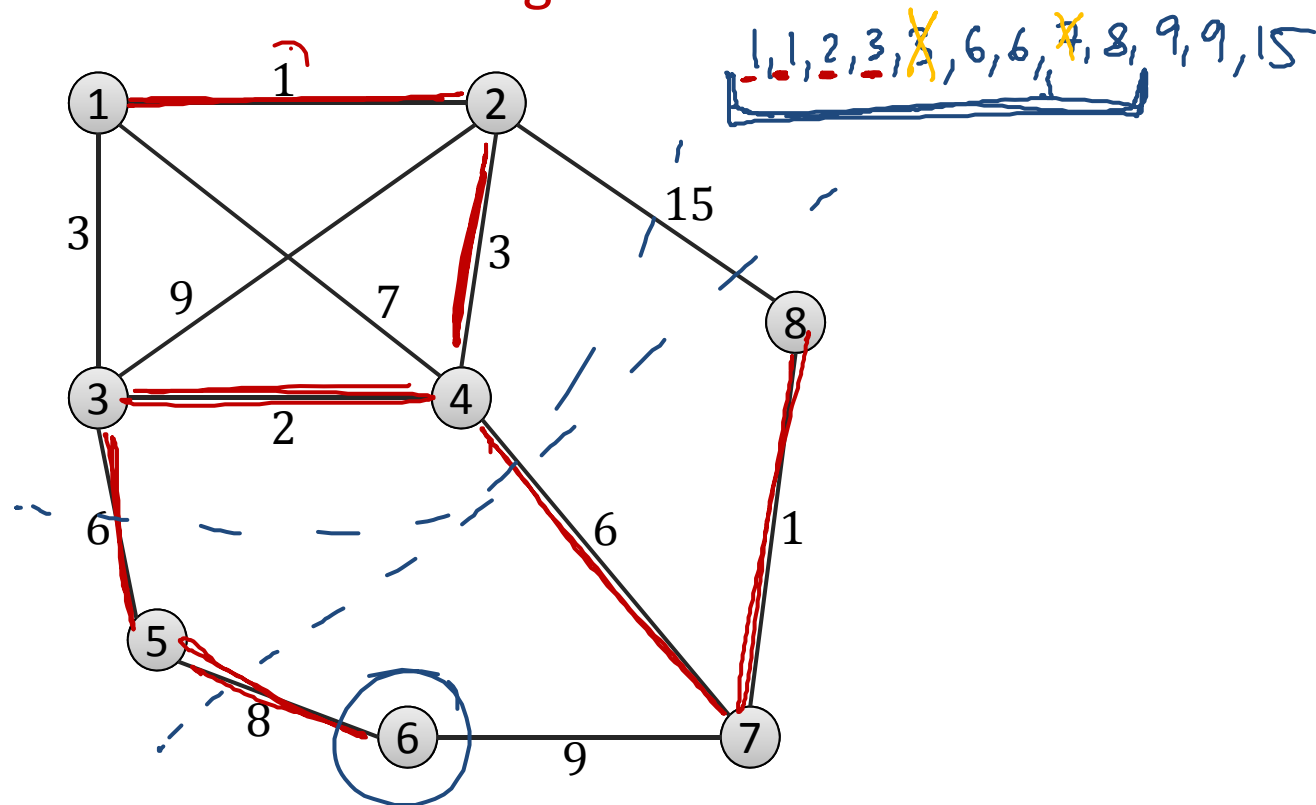
Minimaler Spannbaum

$e \in E$ hat Gewicht $w(e)$

Gegeben: Zus.-hängender, ungerichteter Graph $G = (V, E, \underline{w})$ mit Kantengewichten $w : E \rightarrow \mathbb{R}$

Minimaler Spannbaum $T = (V, E_T)$:

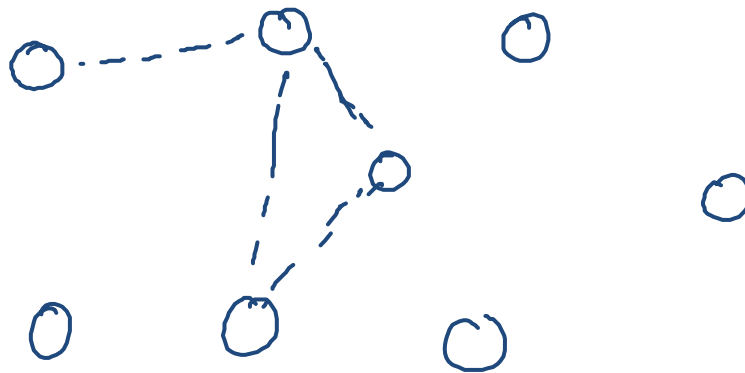
- Spannbaum mit kleinstem Gesamtgewicht



Minimale Spann­b­au­m­e

Ziel: Gegeben ein gewichteter, ungerichteter Graph G , finde einen Spannbaum mit minimalem Gesamtgewicht.

- **Minimaler Spannbaum = Minimum Spanning Tree = MST**
- Ein grundlegendes Optimierungsproblem auf Graphen
 - eines von sehr vielen Optimierungsproblemen auf Graphen
- kommt oft als Teilproblem vor
- ist aber auch interessant an sich



Basis-MST-Algorithmus

Idee: Starte mit leerer Kantenmenge und füge die Kanten schrittweise hinzu, bis es ein Spannbaum ist

Invariante:

Algorithmus hat zu jeder Zeit eine Kantenmenge A , so dass A Teilmenge eines minimalen Spannbaums ist.

- Am Anfang ist $A = \emptyset$
- Danach wird jeweils eine Kante hinzugefügt, ohne die Invariante zu verletzen
- Wir nennen eine Kante, für welche wir sicher sein können, dass wir sie zu A hinzufügen können eine sichere Kante für A
- Wie man sichere Kanten findet, werden wir sehen...

Basis-MST-Algorithmus

Invariante:

Algorithmus hat zu jeder Zeit eine Kantenmenge A , so dass A Teilmenge eines minimalen Spannbaums ist.

Basis-MST-Algorithmus:

$A = \emptyset$

while A ist kein Spannbaum **do**

Finde sichere Kante $\{u, v\}$ für A

$A = A \cup \{u, v\}$

return A

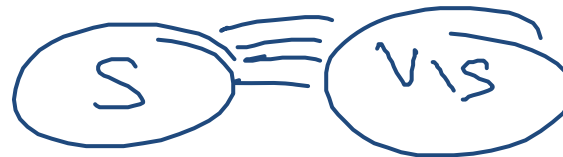
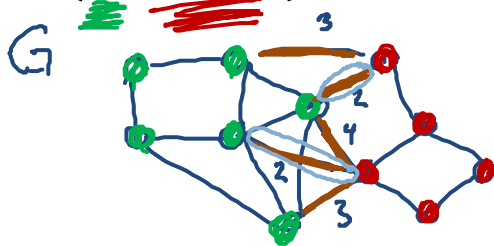
n-1 Wiederholungen

- Invariante ist eine gültige Schleifeninvariante
- **Invariante + Abbruchbedingung \Rightarrow A ist ein MST!**

Wie findet man sichere Kanten?

- Invariante \rightarrow es gibt immer mindestens eine sicher Kante
 - A ist Teilmenge eines MST und kann daher zu einem MST erweitert werden
- Zuerst benötigen wir ein paar Begriffe...

Schnitt $(S, V \setminus S)$:



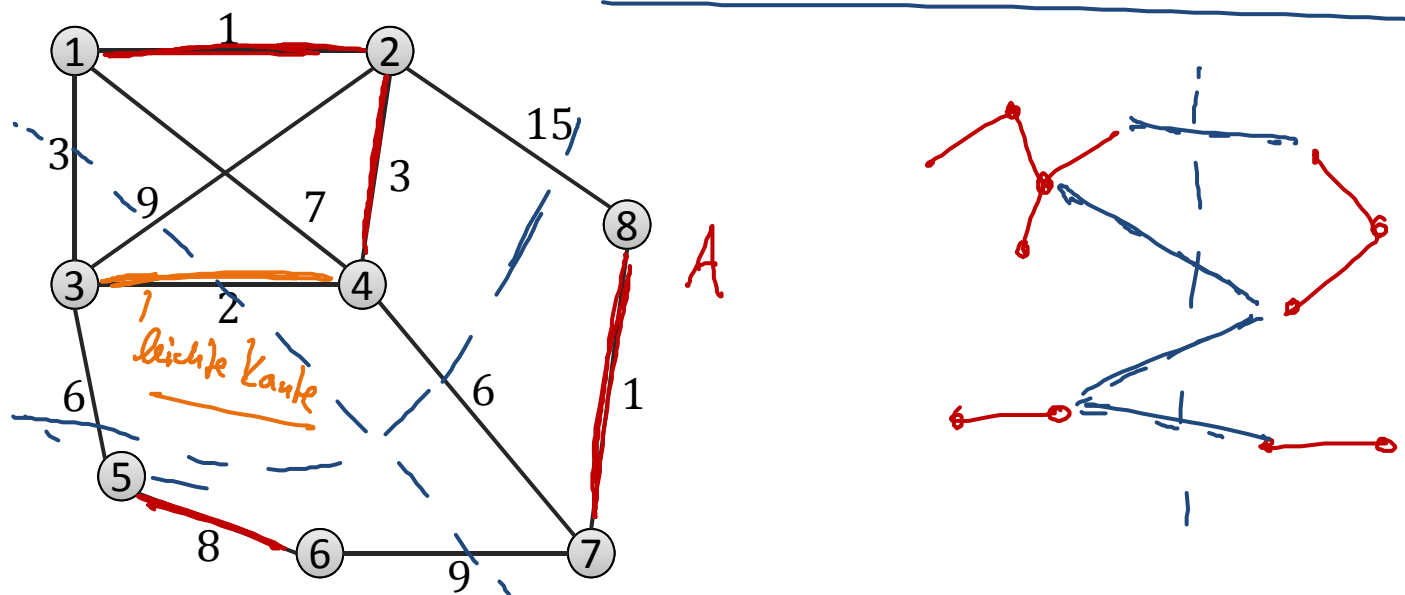
- Kante $\{u, v\} \in E$ ist eine Schnittkante bezüglich $(S, V \setminus S)$, falls ein Ende in S und ein Ende in $V \setminus S$ ist.
- Wir nennen Kante $\{u, v\}$ eine leichte Schnittkante bez. $(S, V \setminus S)$, falls sie das **kleinste Gewicht** von allen Schnittkanten hat

Sichere Kanten

Annahmen:

- $G = (V, E, w)$ ist zus.-h., unger. Graph mit Kantengewichten $w(e)$
- A ist Teilmenge (Teilgraph) eines MST

Theorem: Sei $(S, V \setminus S)$ ein Schnitt, so dass A keine Schnittkanten enthält und sei $\{u, v\}$, $u \in S$, $v \in V \setminus S$ eine leichte Schnittkante bezüglich $(S, V \setminus S)$. Dann ist $\{u, v\}$ eine sichere Kante für A .



Sichere Kanten

Theorem: Sei $(S, V \setminus S)$ ein Schnitt, so dass A keine Schnittkanten enthält und sei $\{u, v\}$, $u \in S$, $v \in V \setminus S$ eine leichte Schnittkante bezüglich $(S, V \setminus S)$. Dann ist $\{u, v\}$ eine sichere Kante für A .

