Technische Universität Braunschweig

Netzwerkalgorithmen – Vorlesung #2

Arne Schmidt

Organisation

Vorlesung

Gr. Übung, kl. Übung

Nähere Information demnächst per Mail!

Wiederholung

Wiederholung

Problem 2.1: Minimal aufspannender Baum (kurz: MST)
Gegeben:
Zshgd. Graph $G = (V, E)$
Kostenfunktion $c\colon E \to \mathbb{R}$
Gesucht:
Kantenmenge $F \subseteq E$ mit $\sum_{e \in F} c(e)$ minimal und $T = (V, F)$ zusammenhängend.

Beobachtung:
Jede optimale Lösung ist kreisfrei.

Satz 2.5
Für einen Graphen $T = (V, E)$ sind folgende Aussagen äquivalent:

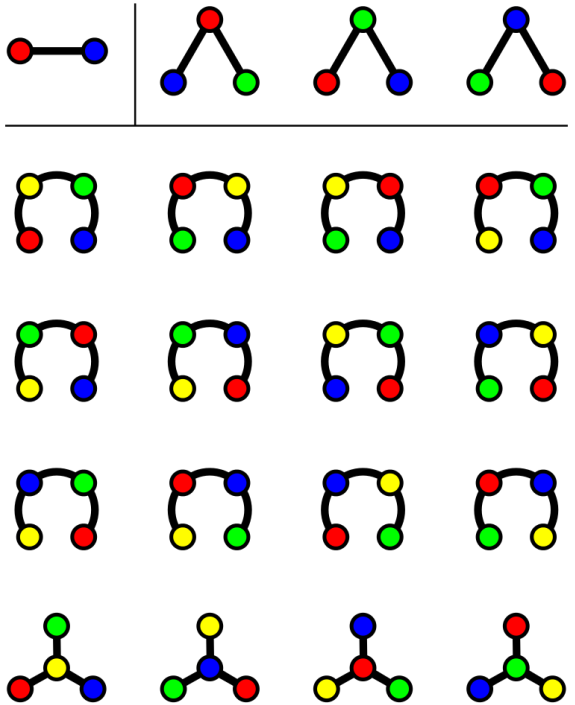
1. T ist ein Baum (zshgd. + kreisfrei).
2. T besitzt $n - 1$ Kanten und ist zshgd.
3. T besitzt $n - 1$ Kanten und ist kreisfrei.
4. T ist maximal kreisfrei (d.h. das Hinzufügen einer bel. Kante erzeugt einen Kreis).
5. T ist minimal zshgd (d.h. das Löschen einer bel. Kante erzeugt einen nicht-zshgd. Graphen).
6. T enthält einen eindeutigen Pfad zwischen jedem Paar von Knoten.

2.2 Berechnung minimaler Spannbäume

Vorüberlegungen

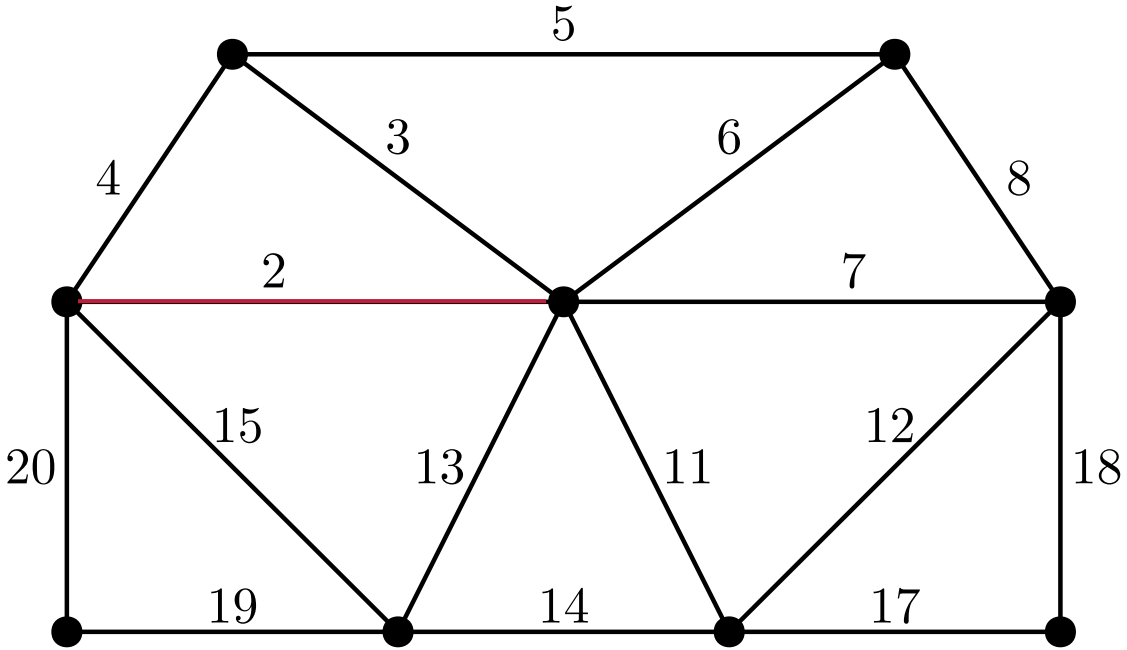
- Enumeration aller Spannbäume?

Satz 2.6 (Formel von Cayley)
Sei G ein vollständiger Graph mit n Knoten. Dann gibt es n^{n-2} viele verschiedene Spannbäume von G.

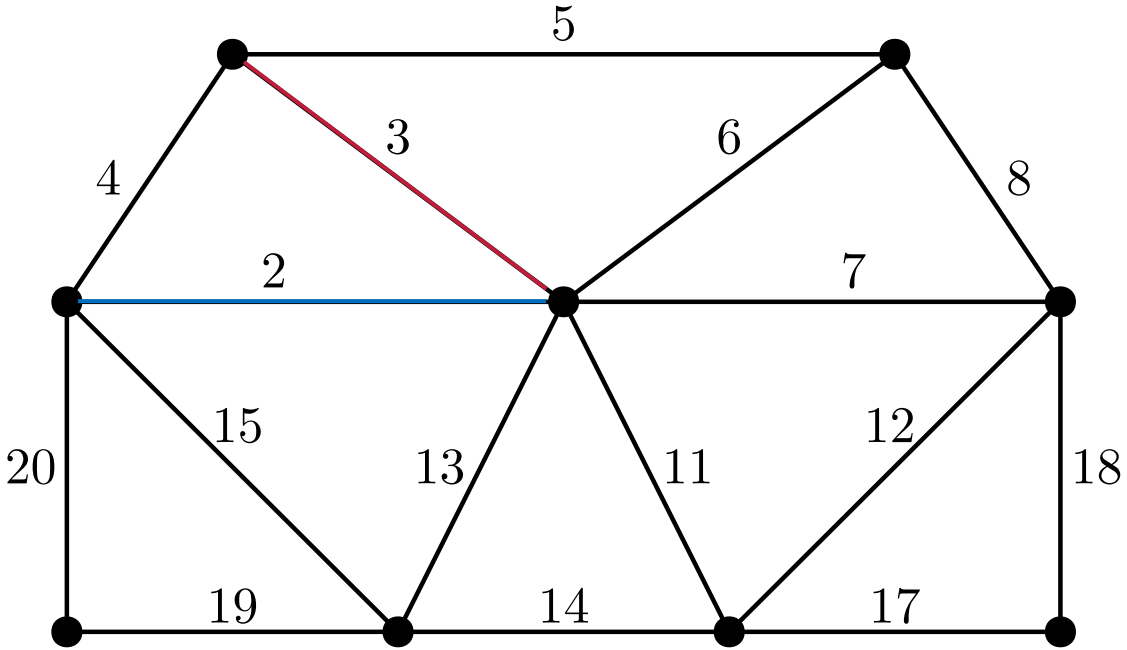


- Iterativ Kanten hinzufügen?
  - Kleinste Kante zuerst?
  - Dann die zweitkleinste Kante?
  - Was passiert, wenn Kreise entstehen?
  - Reicht es die teuerste Kante zu löschen?

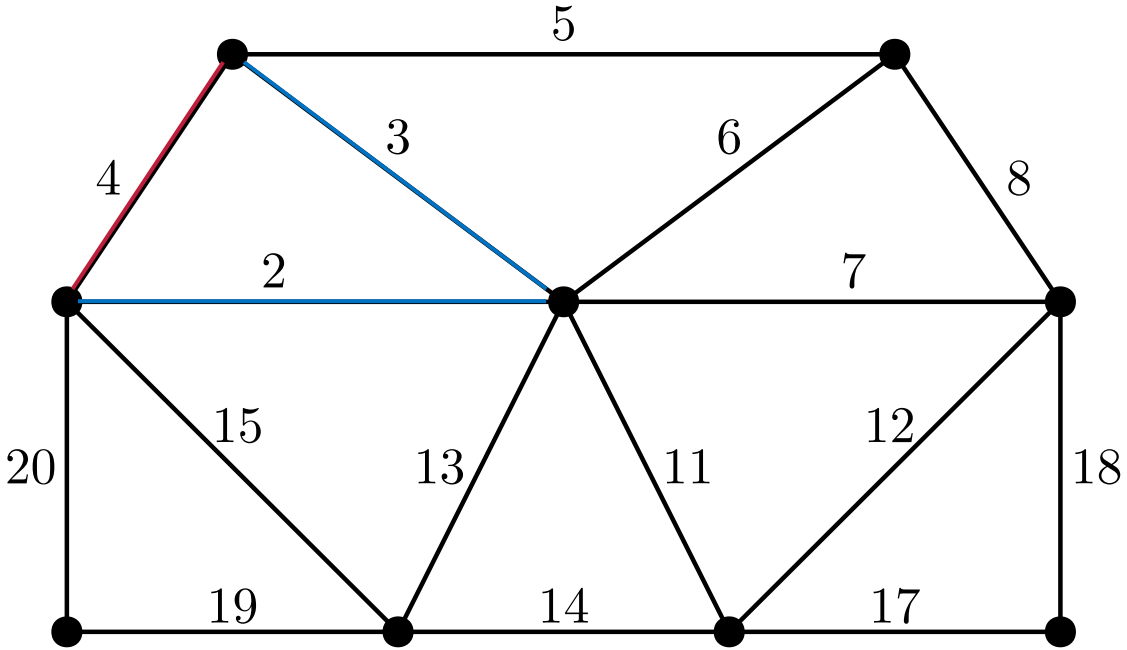
Ein Beispiel



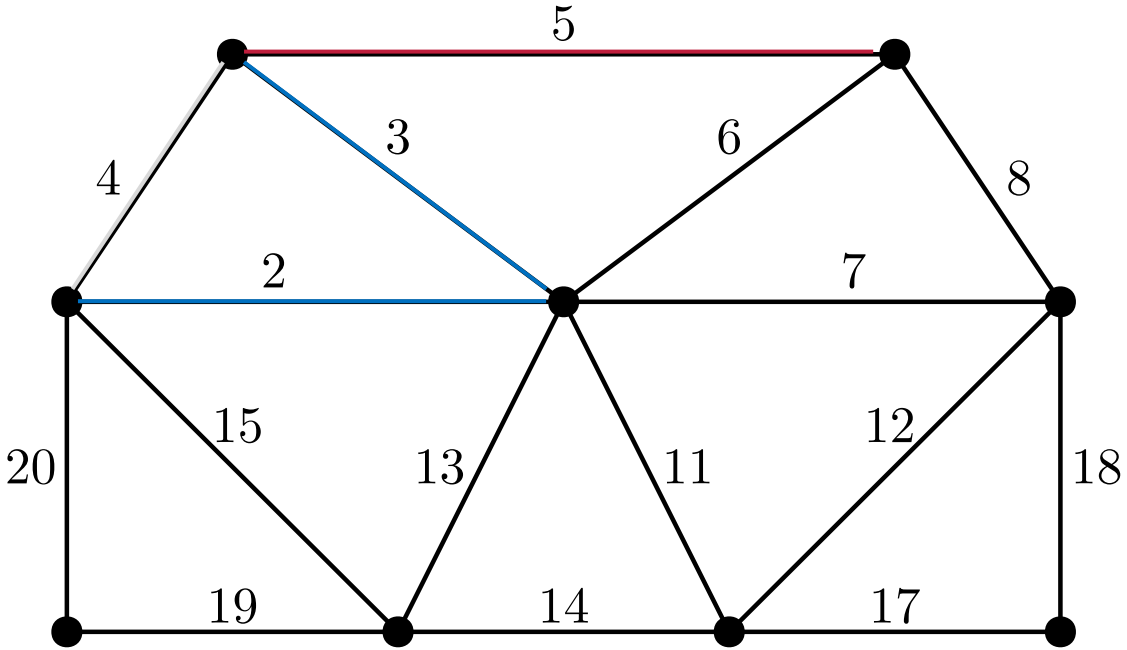
Ein Beispiel



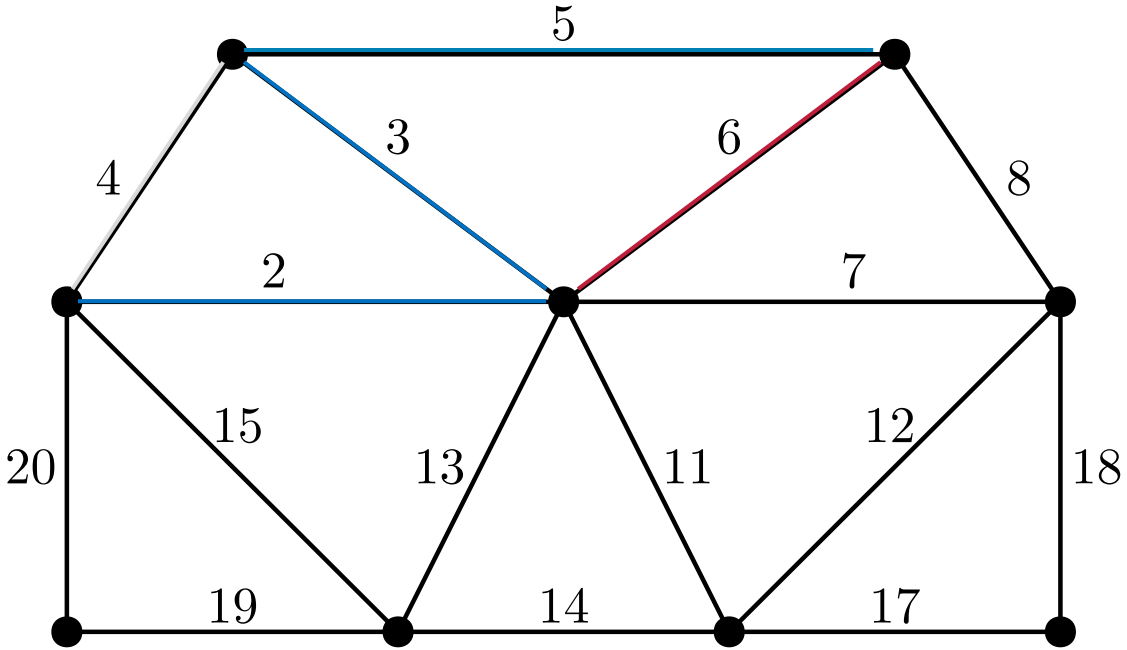
Ein Beispiel



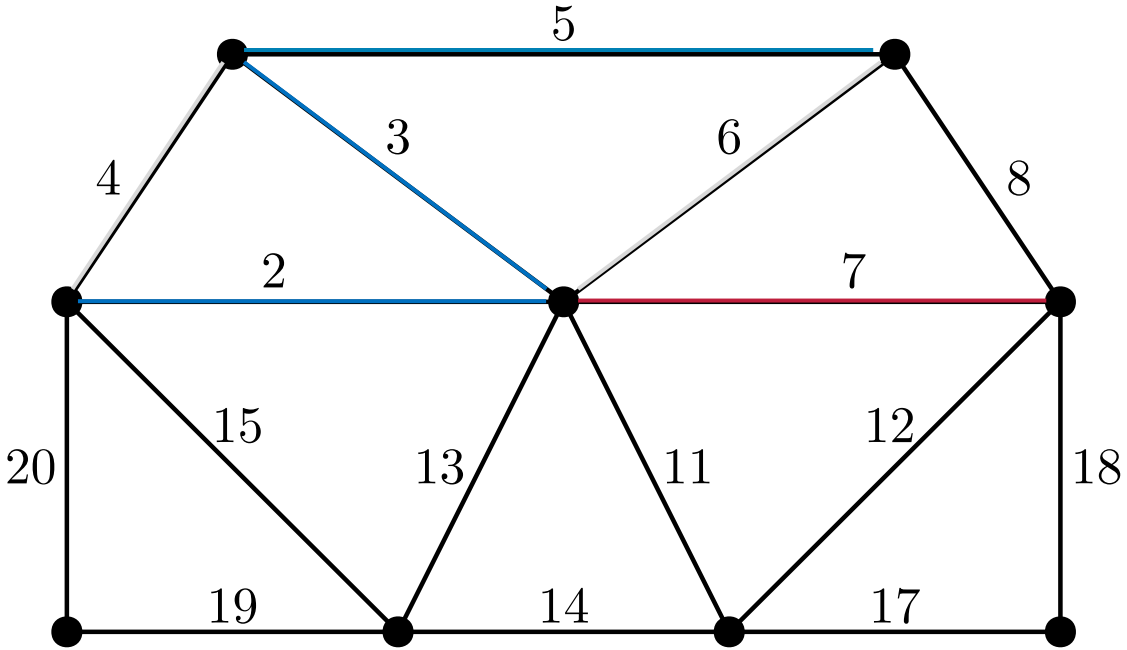
Ein Beispiel



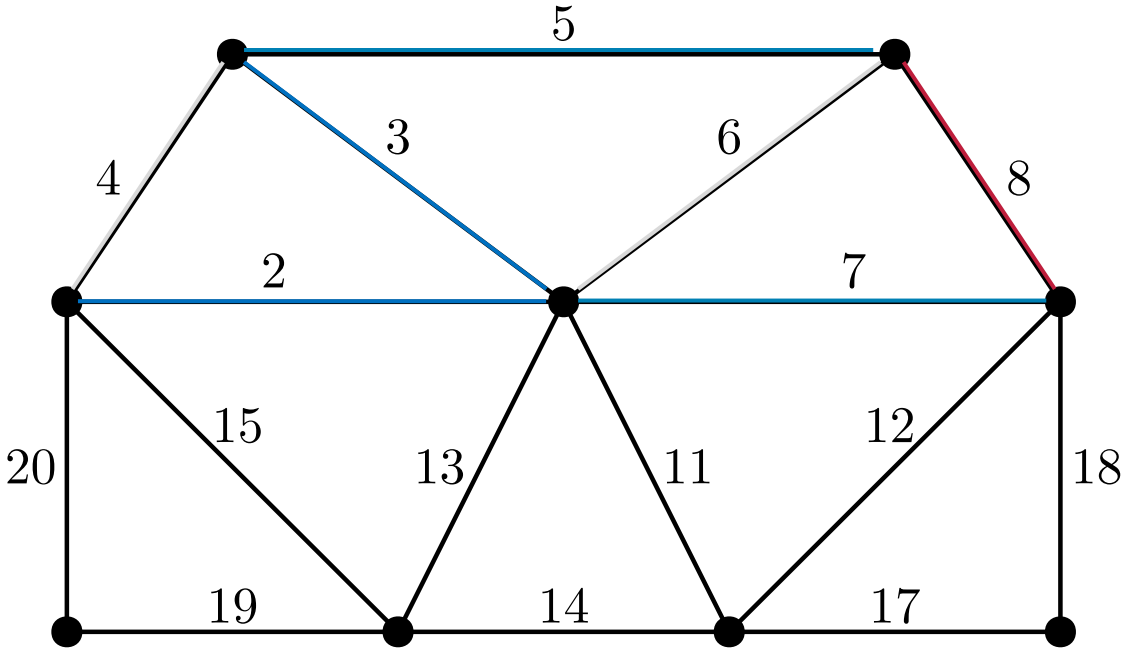
Ein Beispiel



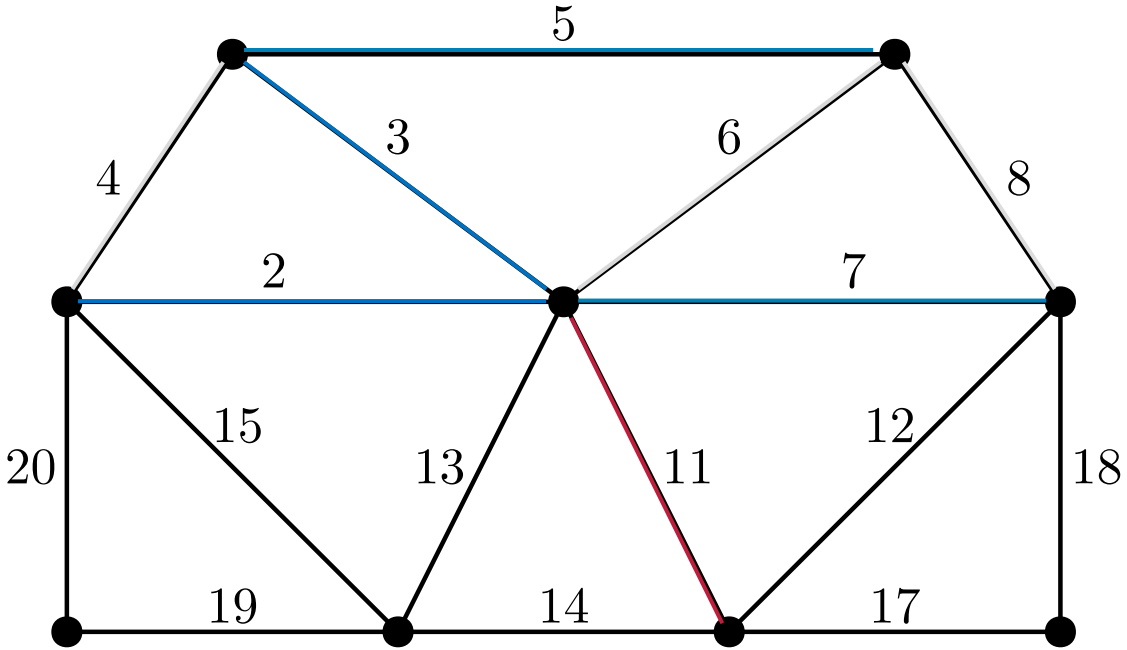
Ein Beispiel



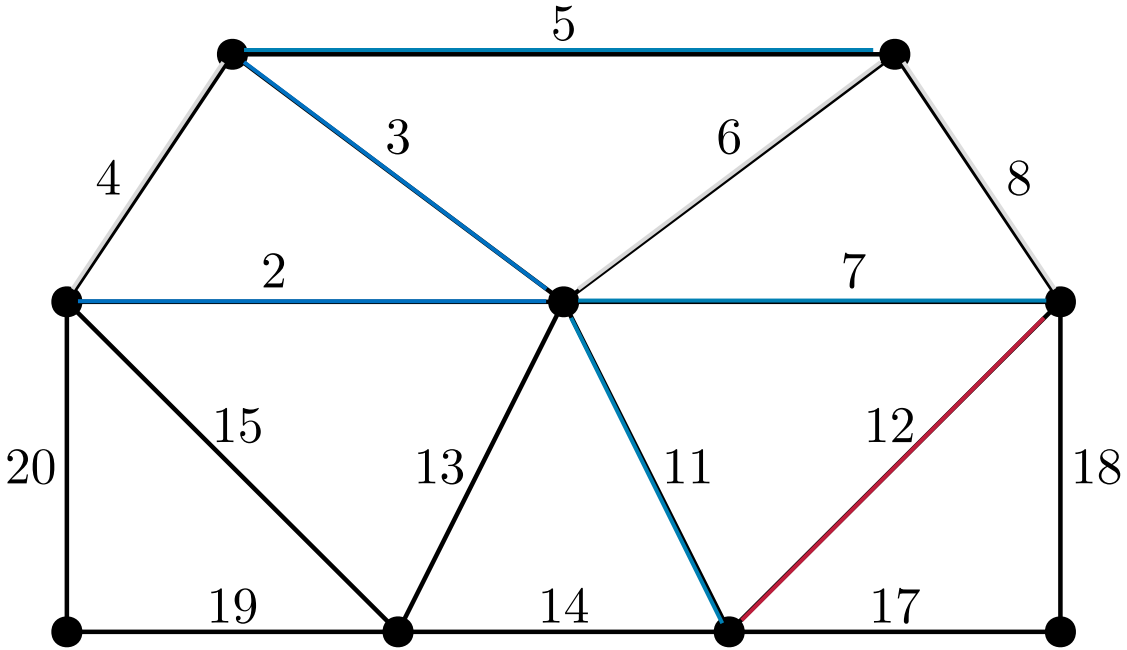
Ein Beispiel



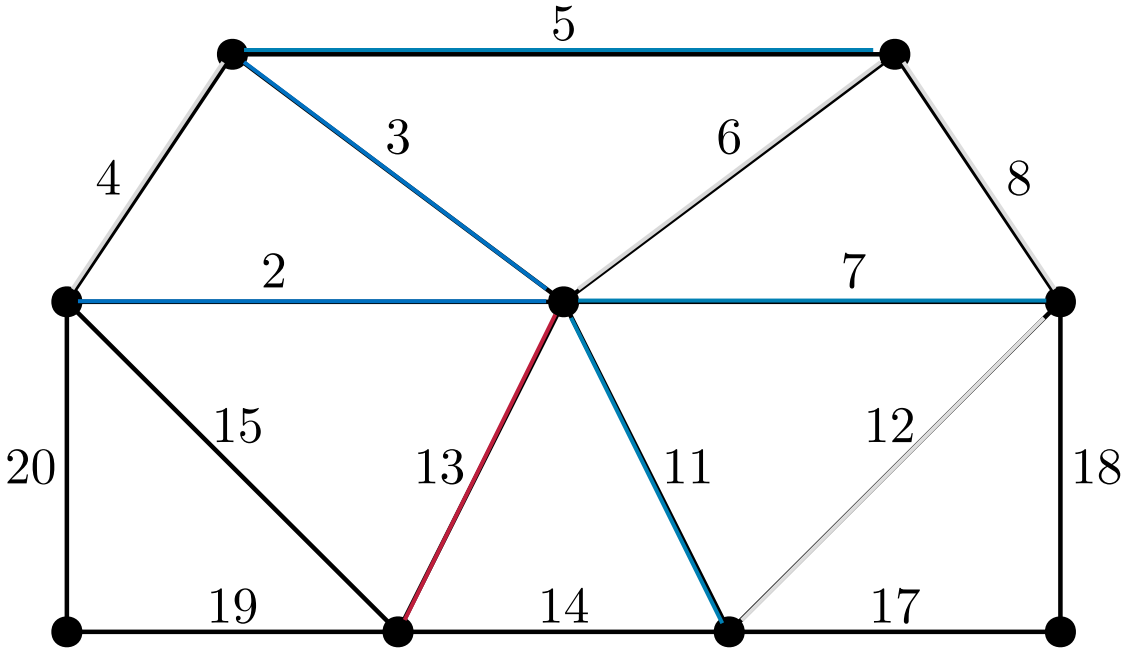
Ein Beispiel



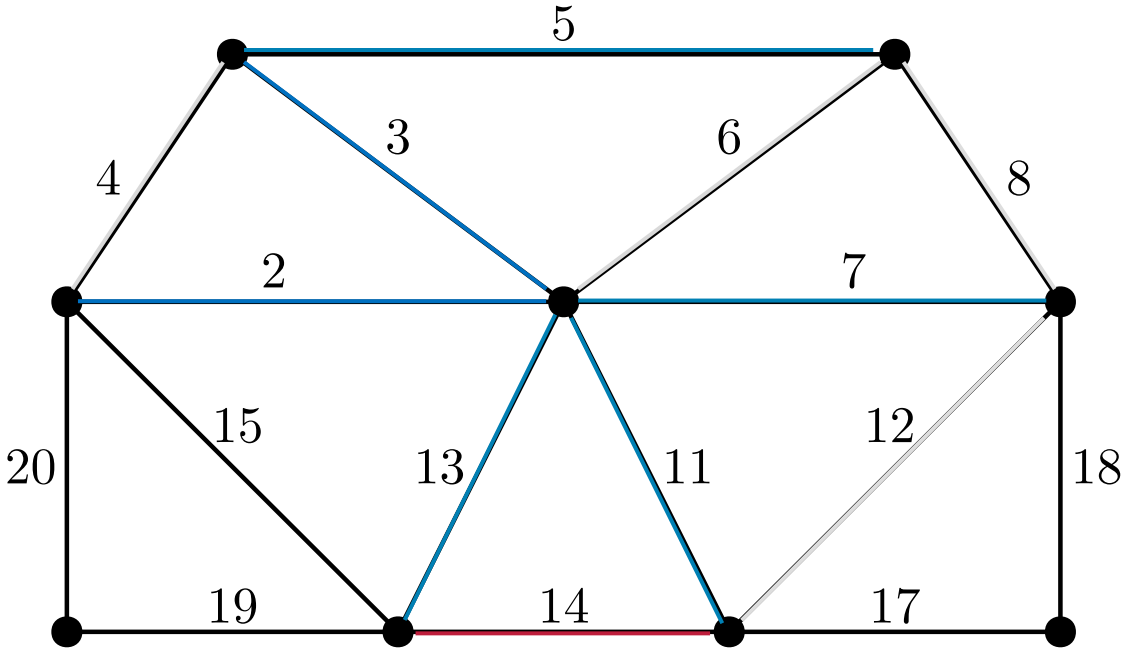
Ein Beispiel



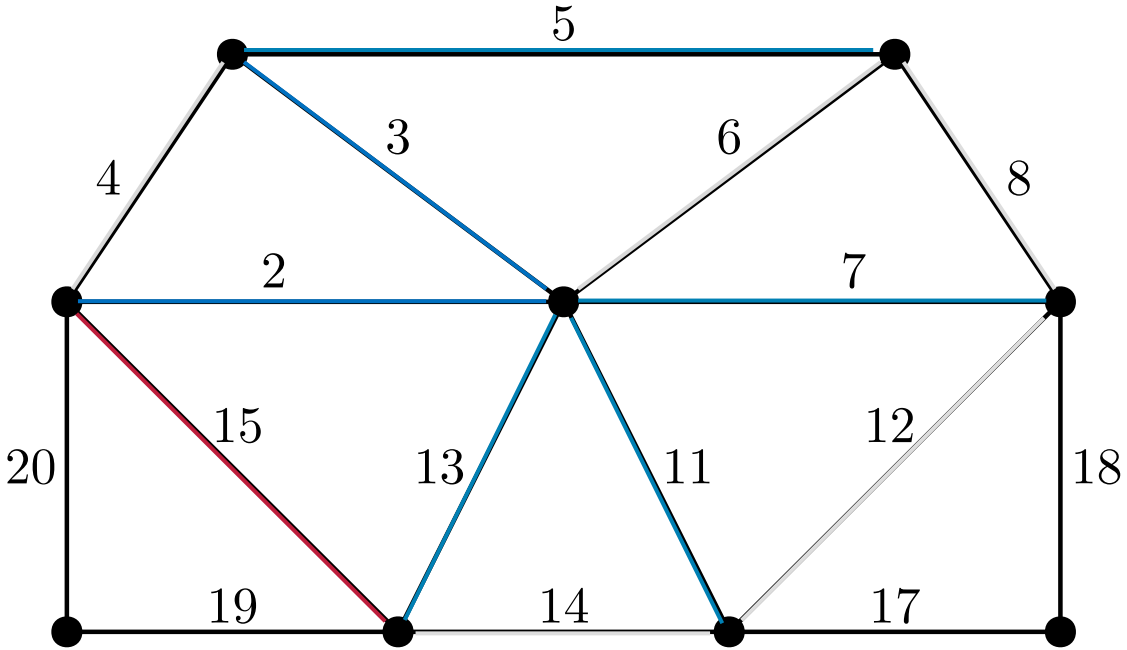
Ein Beispiel



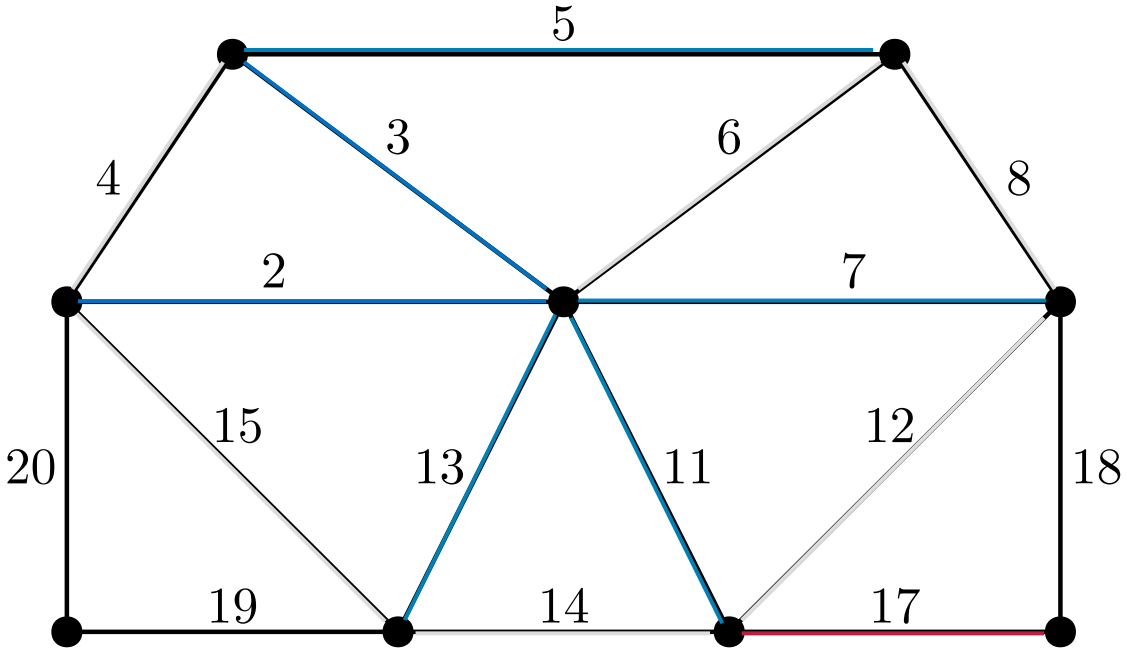
Ein Beispiel



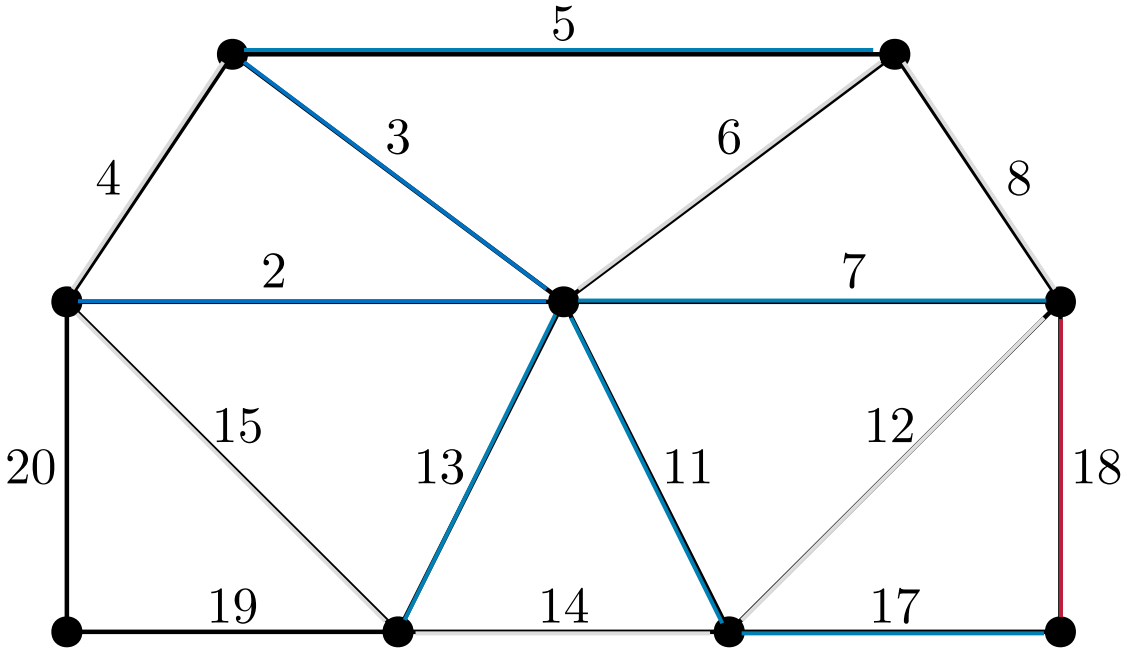
Ein Beispiel



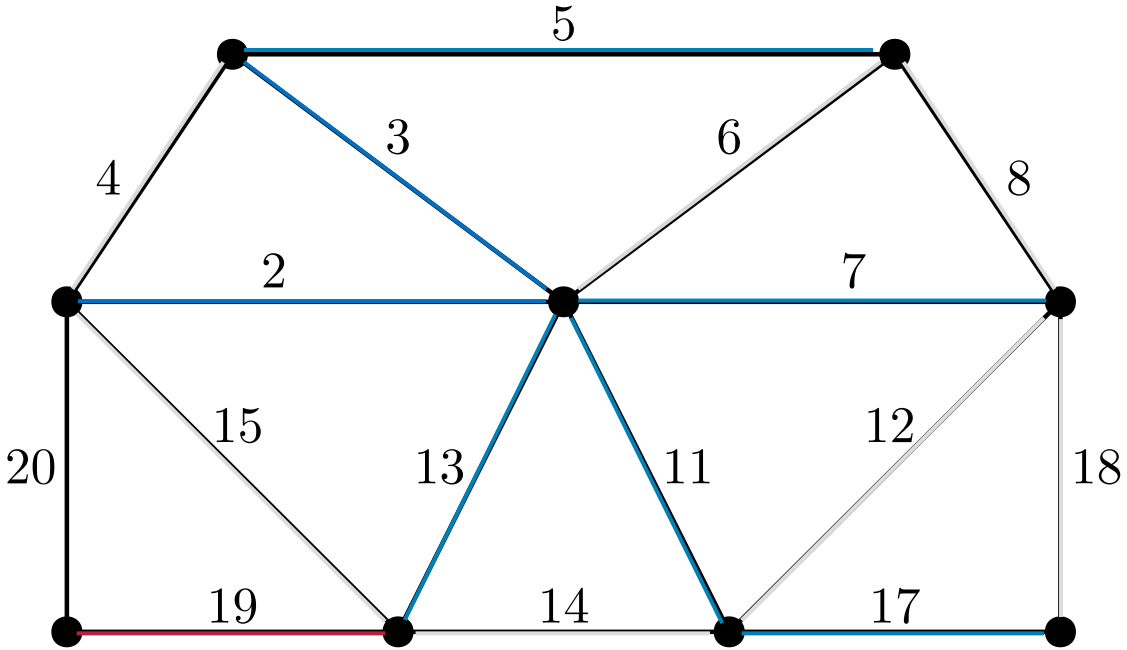
Ein Beispiel



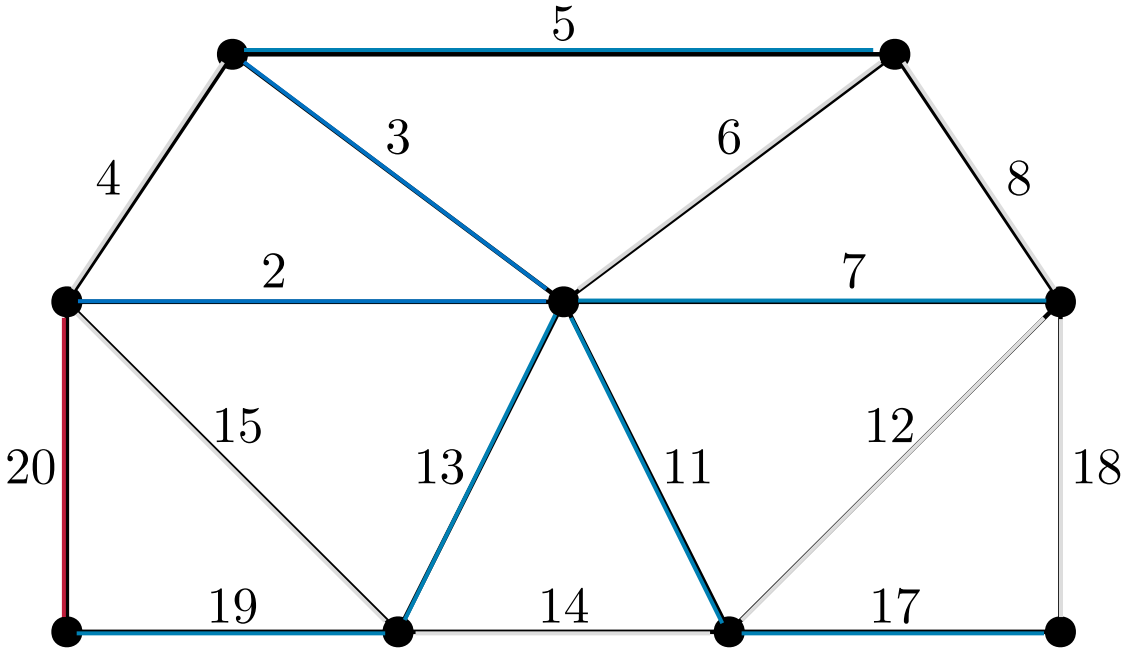
Ein Beispiel



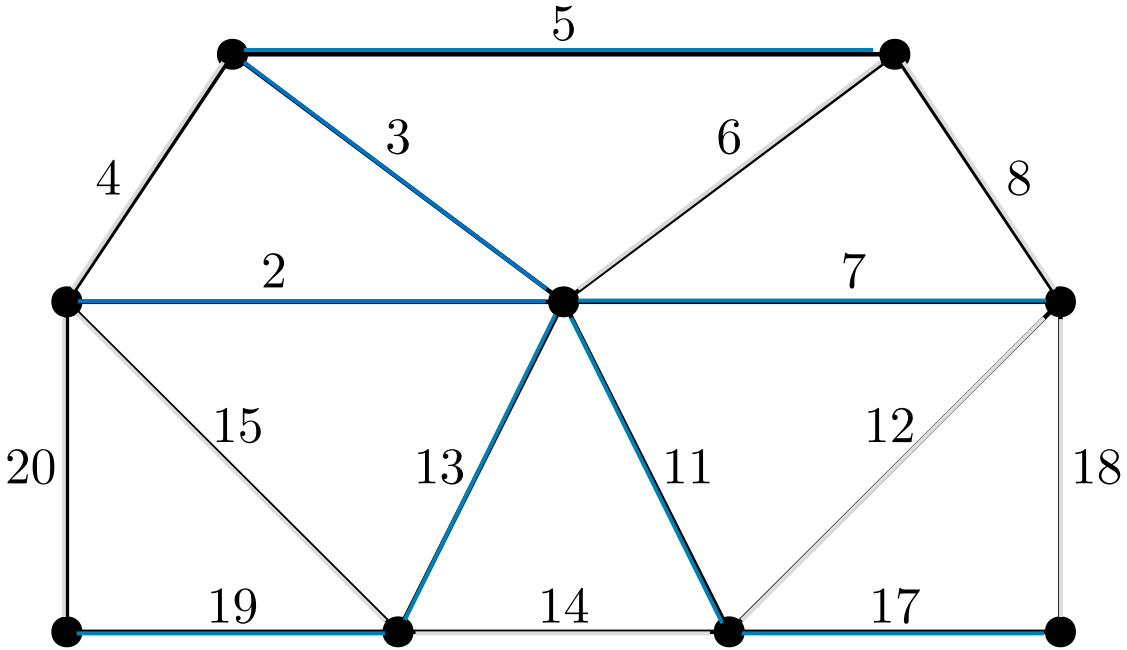
Ein Beispiel



Ein Beispiel



Ein Beispiel

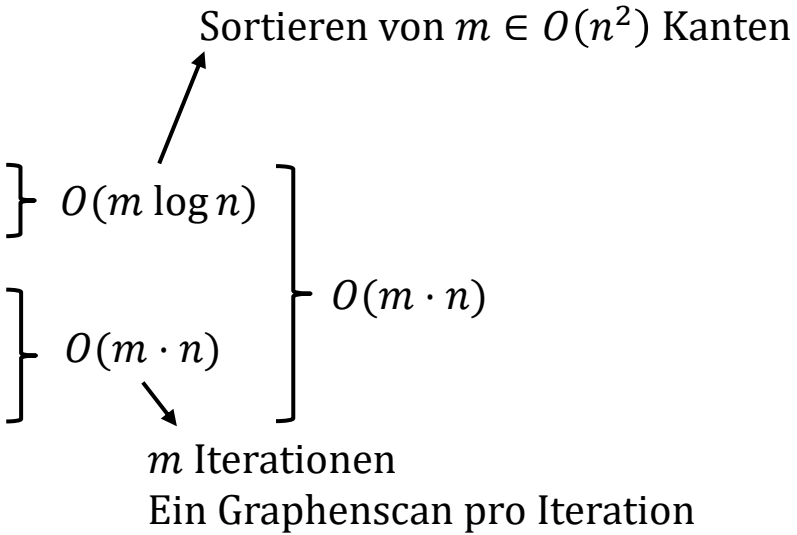


2.2.1 Algorithmus von Kruskal

Algorithmus von Kruskal

Algorithmus 2.7
Eingabe:
Zshgd. Graph $G = (V, E)$,
Kostenfunktion c: $E \to \mathbb{R}^+$
Ausgabe:
Spannbaum $T = (V, F)$ minimalen Gesamtgewichts

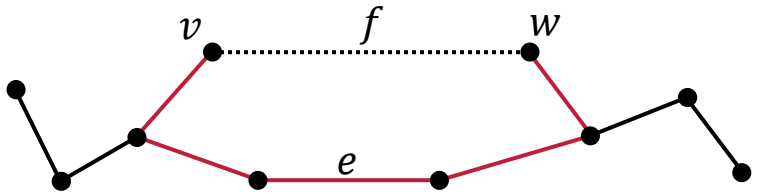
1. Function KRUSKAL(G, c)
2. Sortiere Kanten aufsteigend nach Gewicht, sodass $c(e_1) \leq c(e_2) \leq \cdots \leq c(e_m)$
3. $F := \emptyset$
4. For $i = 1$ to m do
5. if $(V, F \cup \{e_i\})$ kreisfrei then
6. $F := F \cup \{e_i\}$
7. Return $T := (V, F)$



Algorithmus von Kruskal

Satz 2.8
Kruskal's Algorithmus löst Problem 2.1 (MST) korrekt in $O(mn)$ Zeit.

Satz 2.9
Sei $G = (V, E)$ ein Graph, $c\colon E \to \mathbb{R}^+$ eine Kostenfunktion und $T = (V, F)$ ein aufspannender Baum von G. Dann sind folgende Aussagen äquivalent.
(a) T ist ein MST
(b) Für jede Kante $f = \{v, w\} \in E \setminus F$ gilt: Für jede Kante e auf dem vw-Pfad in T ist $c(e) \leq c(f)$



Algorithmus von Kruskal

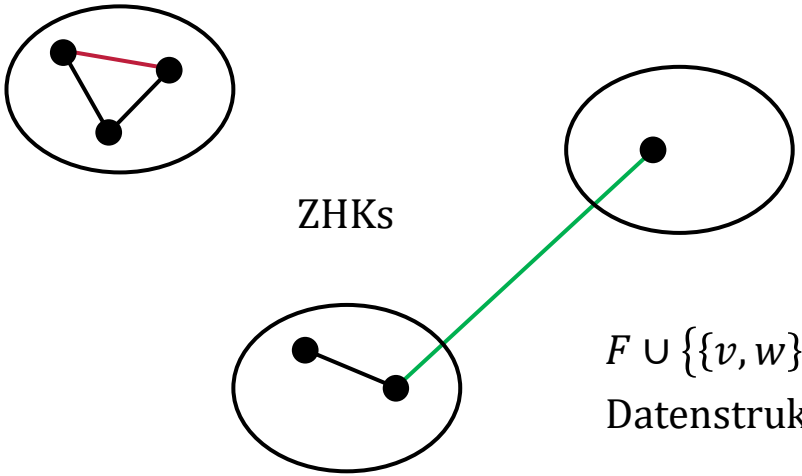
Satz 2.8
Kruskal's Algorithmus löst Problem 2.1 (MST) korrekt in $O(mn)$ Zeit.

Beweis:
Laufzeit bereits gesehen.

Korrektheit: Der vom Algorithmus konstruierte Baum T erfüllt Eigenschaft (b) aus Satz 2.9. T ist also ein MST.

Algorithmus von Kruskal

Lemma 2.10
Kruskal‘s Algorithmus lässt sich so implementieren, dass sich eine Laufzeit von $O(m \log n)$ ergibt.



$F \cup \{\{v, w\}\}$ kreisfrei $\Leftrightarrow$ v und w liegen in verschiedenen ZHKs

Datenstruktur zum Erkennen von gleichen ZHKs nötig!

Union-Find-Datenstruktur

Idee:

- Jede ZHK wird durch einen Knoten repräsentiert.
- Jeder Knoten in der ZHK zeigt (indirekt) auf den Repräsentanten.

Union-Find-Datenstruktur

Definition 2.11 (Arboreszenz)
Eine *Arboreszenz* A_r ist ein gerichteter Graph, in dem jeder Knoten v bis auf r (die Wurzel) einen eindeutigen Vorgänger $p(v)$ besitzt.
Die *Höhe* $h(A_r)$ der Arboreszenz entspricht der Länge eines längsten gerichteten Pfades in A_r.

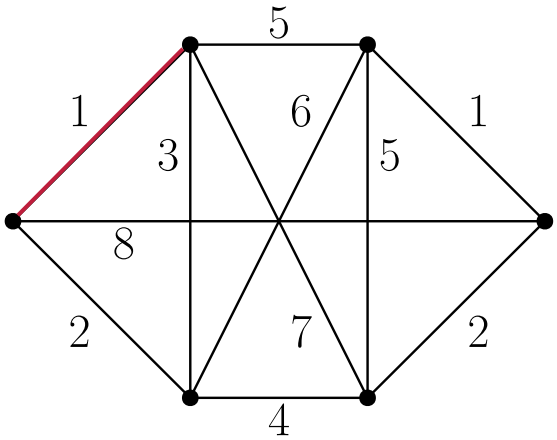
Initialisierung:
Jeder Knoten bildet eine eigene Arboreszenz.

Union (A_v, A_w):
O.B.d.A. sei $h(A_v) \geq h(A_w)$
Verweise von Wurzel(A_w) auf Wurzel(A_v)

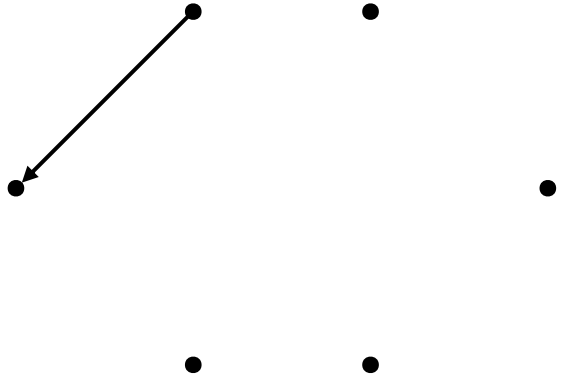
Find (v):
Folge den Kanten startend in v, um Knoten r ohne Vorgänger mit $v \in A_r$ zu finden.

Ein Beispiel

Graph

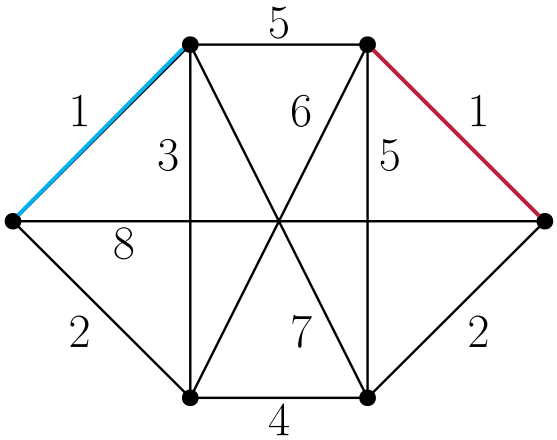


Union-Find-Datenstruktur

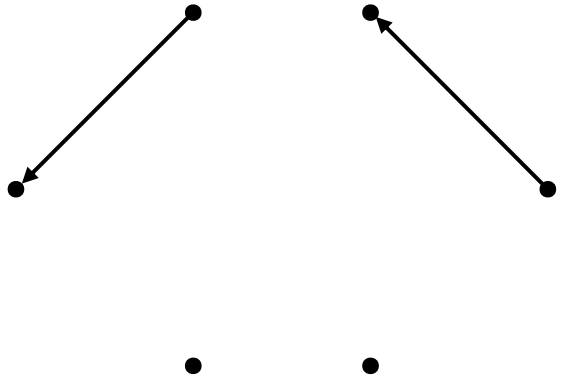


Ein Beispiel

Graph

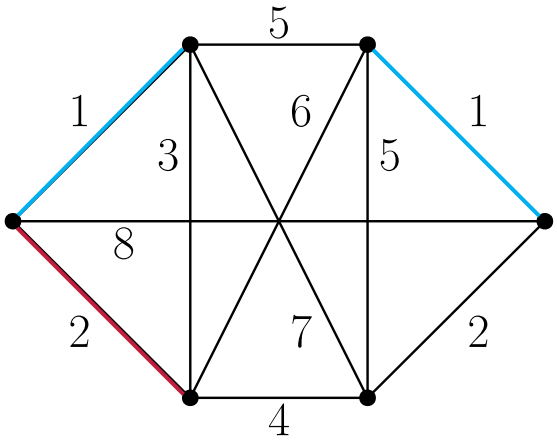


Union-Find-Datenstruktur

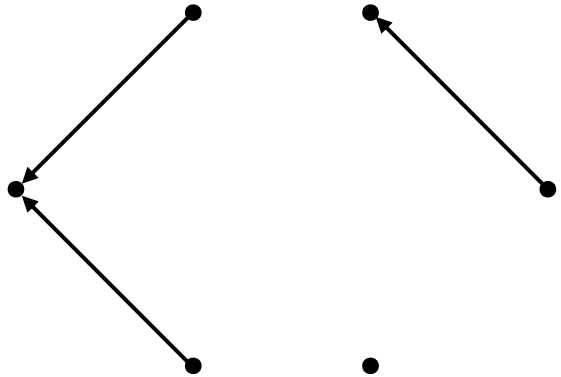


Ein Beispiel

Graph

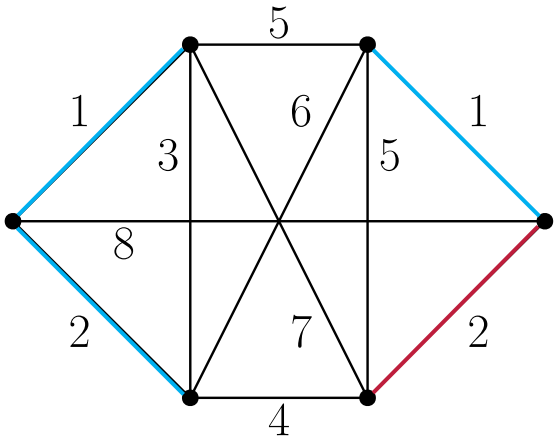


Union-Find-Datenstruktur

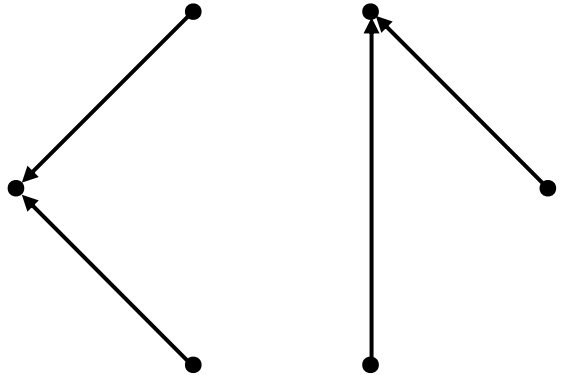


Ein Beispiel

Graph

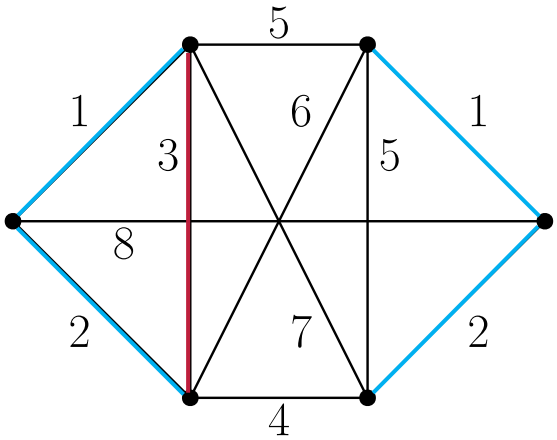


Union-Find-Datenstruktur

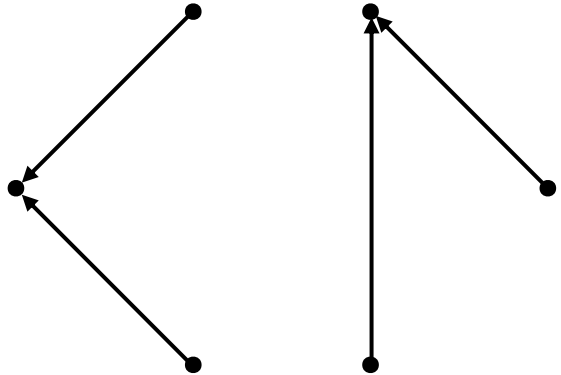


Ein Beispiel

Graph

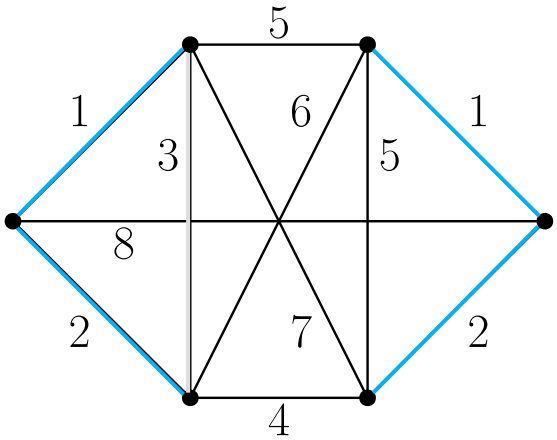


Union-Find-Datenstruktur

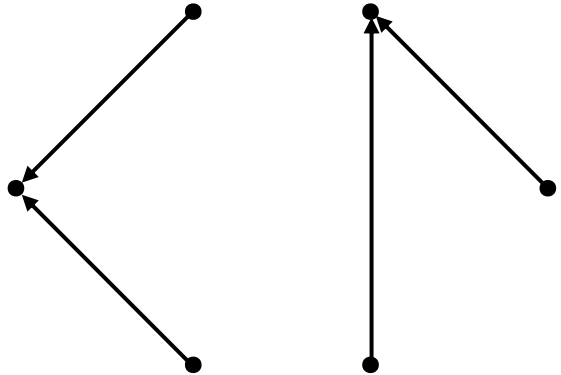


Ein Beispiel

Graph

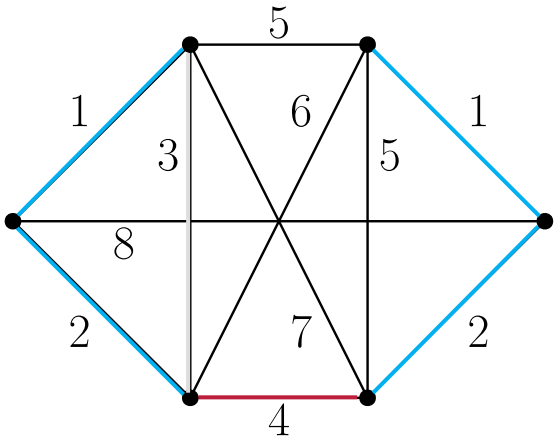


Union-Find-Datenstruktur

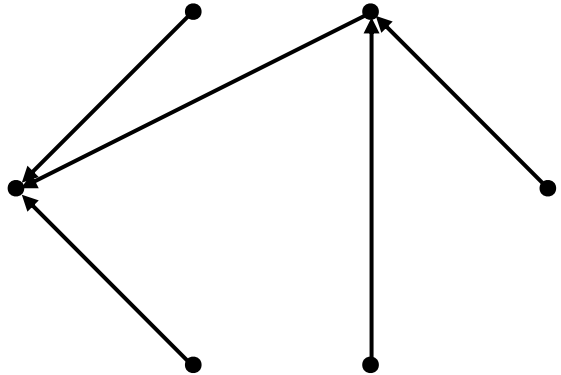


Ein Beispiel

Graph

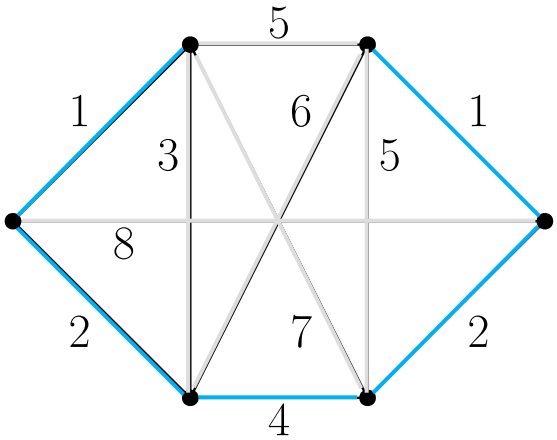


Union-Find-Datenstruktur

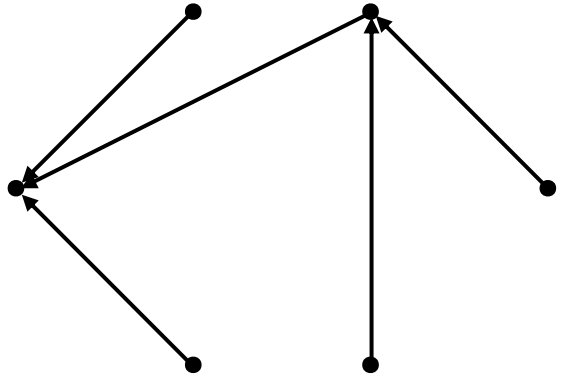


Ein Beispiel

Graph



Union-Find-Datenstruktur



Union-Find-Datenstruktur

Lemma 2.10
Kruskal‘s Algorithmus lässt sich so implementieren, dass sich eine Laufzeit von $O(m \log n)$ ergibt.

Lemma 2.12
Jede r-Arboreszenz A_r in der Union-Find-Datenstruktur mit Höhe $h(A_r)$ besitzt mindestens $2^{h(A_r)}$ Knoten.

Mit Lemma 2.12 folgt, dass $h(A_r) \in O(\log n)$ gilt. Damit kostet der Test auf Kreisfreiheit nur noch $O(\log n)$ Zeit. Damit ist Lemma 2.10 bewiesen.