Technische Universität Braunschweig

Netzwerkalgorithmen – Vorlesung #3

Arne Schmidt

Netzwerkalgorithmen **Sommersemester 2021**

Dr. Arne Schmidt
Christian Rieck

Abgabe: 16. Mai 2021
Rückgabe: 21. Mai 2021

Übungsblatt 1

Bitte schickt eure Lösungen in einer einzigen PDF Datei (Name der Datei und Betreff: „blatt[nr]_[name]_[matrikelnr]“) per Mail an euren jeweiligen Tutor. Bitte beachtet auch den Hausaufgaben-Merkzettel[1]!

Hausaufgabe 1: **(4+3 Punkte)**

Sei $G = (V, E)$ ein Baum mit n Knoten und sei $n \geq 2$.

a) Zeige: G besitzt $n - 1$ Kanten.

b) Zeige: G besitzt mindestens zwei Blätter.

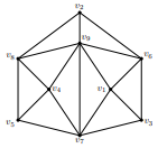
Hausaufgabe 2: **(5+5 Punkte)**

Sei $G = (V, E)$ der Graph der in Abbildung 1 dargestellt ist. Die Kante zwischen den Knoten v_i und v_j $(i < j)$ ist mit $e_{i,j}$ und das Gewicht mit $c(e_{i,j})$ bezeichnet.

a) Bestimme einen MST in G mit dem Algorithmus von Kruskal.

b) Bestimme einen MST in G mit dem Algorithmus von Prim und dem Startknoten v_6.

Gib die Reihenfolge der eingefügten Kanten an und zeichne den MST. Kommen in einem Schritt des Algorithmus mehrere Kanten infrage, wähle die Kante in lexikographischer Ordnung aus (also z.B. $e_{a,b}$ vor $e_{c,d}$, wenn $a < c$ bzw. $a = c$ und $b < d$).



$c(e_{1,3}) = 1$ $c(e_{4,5}) = 2$
$c(e_{1,6}) = 3$ $c(e_{4,7}) = 6$
$c(e_{1,7}) = 6$ $c(e_{4,8}) = 3$
$c(e_{1,9}) = 7$ $c(e_{4,9}) = 4$
$c(e_{2,6}) = 9$ $c(e_{5,7}) = 7$
$c(e_{2,8}) = 9$ $c(e_{5,8}) = 1$
$c(e_{2,9}) = 8$ $c(e_{6,8}) = 5$
$c(e_{3,6}) = 2$ $c(e_{7,9}) = 5$
$c(e_{3,7}) = 7$ $c(e_{8,9}) = 3$

Abbildung 1: Eine Zeichnung des Graphen G und die Gewichte der jeweiligen Kanten.

Hausaufgabe 3: **(3 Punkte)**

Sei $G = (V, E)$ ein Graph und T der von Kruskals Algorithmus konstruierte aufspannende Baum von G. Zeige, dass T die Optimalitätsbedingung (b) aus Satz 2.9 (der VL) erfüllt.

[1] https://www.ibr.cs.tu-bs.de/alg/Merkzettel/homework-booklet.pdf

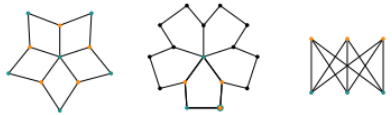
Präsenzaufgabe:

Ein Graph heißt *bipartit*, wenn die Knotenmenge des Graphen in zwei disjunkte Mengen V_1 und V_2 zerlegt werden kann, sodass jede Kante je einen Knoten aus V_1 und V_2 enthält, siehe Abbildung 2.

Sei $K_{m,n} = (V_1 \dot\cup V_2, E)$ der vollständig bipartite Graph mit $|V_1| = m$, $|V_2| = n$ und $E := \{\{u, v\} \mid u \in V_1 \text{ und } v \in V_2\}$.

a) Zeige, dass der $K_{2,n}$ genau $n \cdot 2^{n-1}$ aufspannende Bäume besitzt.

b) Zeige, dass der $K_{3,n}$ genau $n^2 \cdot 3^{n-1}$ aufspannende Bäume besitzt.



(a) (b) (c)

Abbildung 2: (a) Eine Darstellung eines bipartiten Graphen. Die Mengen V_1 und V_2 sind über die Knotenfarben gegeben. (b) Eine Darstellung eines Graphen der nicht bipartit ist. Die Knoten des markierten Kreises können nicht in zwei disjunkte Mengen unterteilt werden, sodass die Eigenschaft eines bipartiten Graphen nicht erfüllt ist. (c) Eine Darstellung des vollständig bipartiten Graphen $K_{3,3}$.

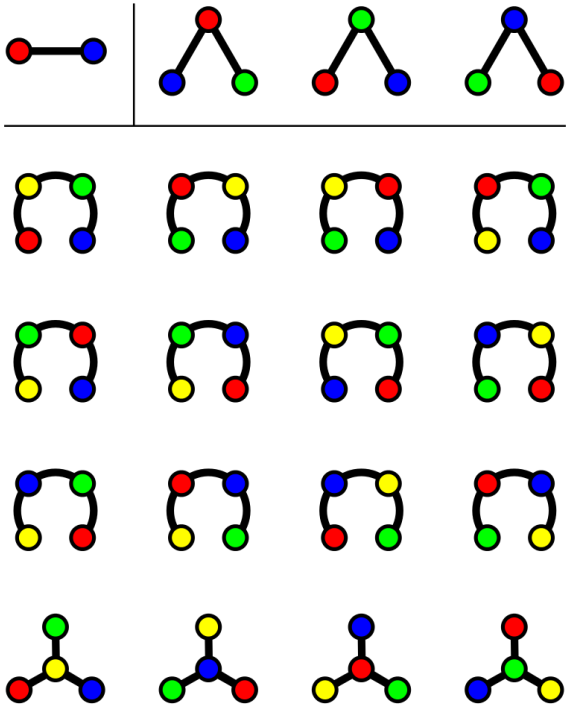
Präsenzaufgabe:

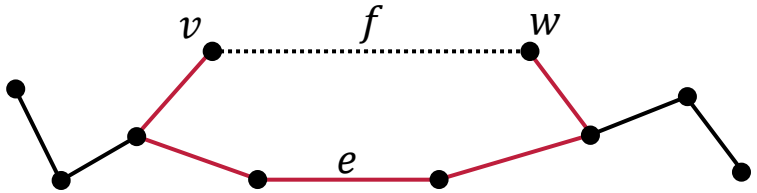
In einer Stadt benötigt jedes Haus einen Wasseranschluss. Es kostet c_i Euro, einen Brunnen an Haus i zu graben und $c_{i,j}$ Euro eine Leitung zwischen den Häusern i und j zu verlegen. Ein Haus bekommt Wasser wenn es einen Brunnen besitzt, oder es einen Pfad aus Leitungen gibt, der zu einem Brunnen führt.

Konstruiere einen Graphen G und zeige, dass ein Wassernetzwerk minimaler Kosten einem minimal aufspannenden Baum in G entspricht.

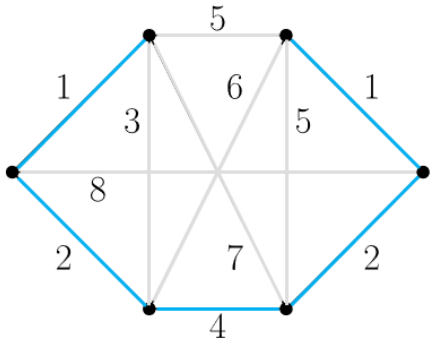
Wiederholung

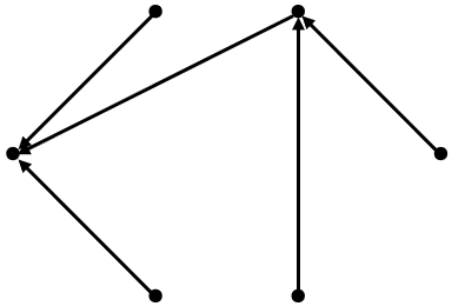
Wiederholung





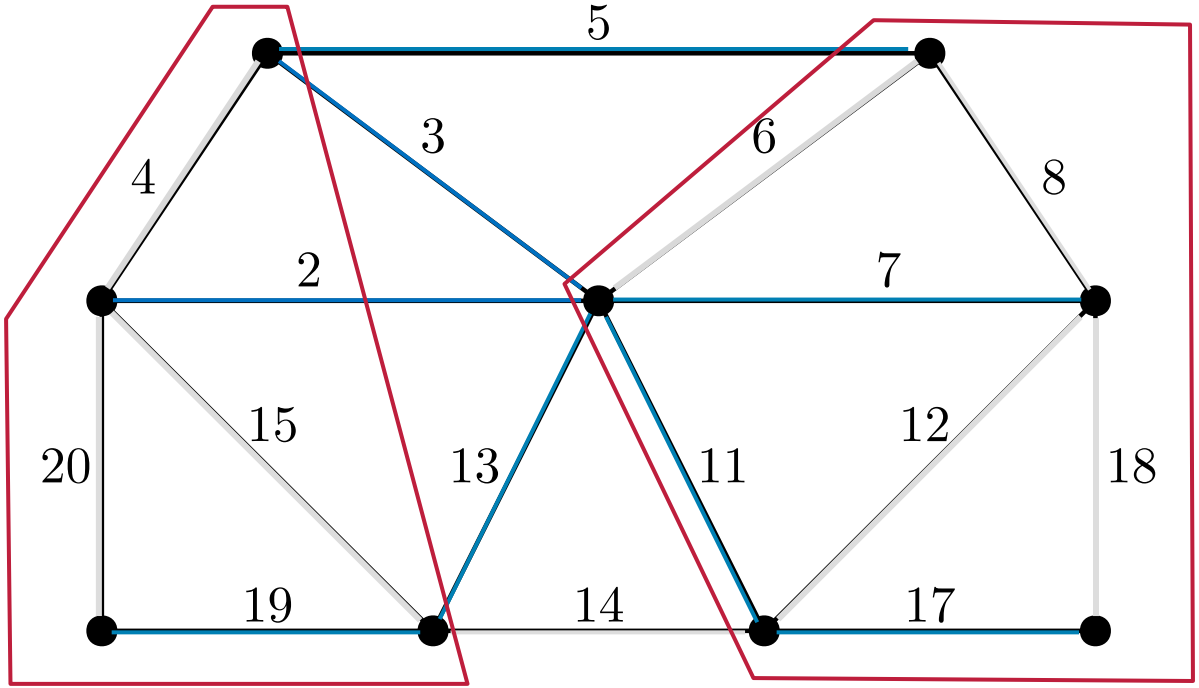
Lemma 2.10
Kruskal‘s Algorithmus lässt sich so implementieren, dass sich eine Laufzeit von $O(m \log n)$ ergibt.



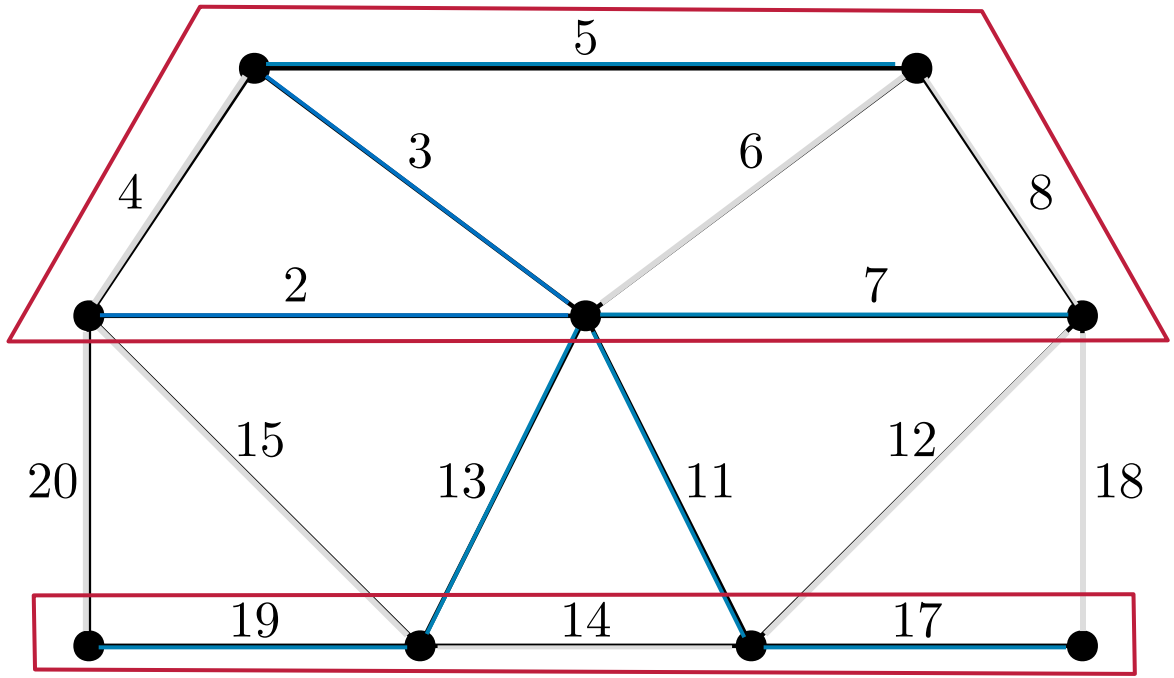


2.2.2 Algorithmus von Prim

Schnitte



Schnitte



Schnitte

Definition 2.13
Für einen Graphen $G = (V, E)$ und eine Menge $X \subseteq V$ ist der *Schnitt* $E(X, V \setminus X)$ definiert durch

$$E(X, V \setminus X) := \{e = \{u, v\} \mid u \in X, v \notin X\}.$$

Lemma 2.14
Ein Graph $G = (V, E)$ ist zshgd. $\Leftrightarrow$ Für alle Mengen $\emptyset \neq X \subsetneq V$ ist $E(X, V \setminus X) \neq \emptyset$.

Satz 2.15
Für einen Graph $G = (V, E)$ und einen aufspannenden Baum T in G sind folgende Aussagen äquivalent:
(a) T ist MST
(b) Für alle Mengen $\emptyset \neq X \subsetneq V$ ist die kleinste Kante aus $E(X, V \setminus X)$ in T enthalten.

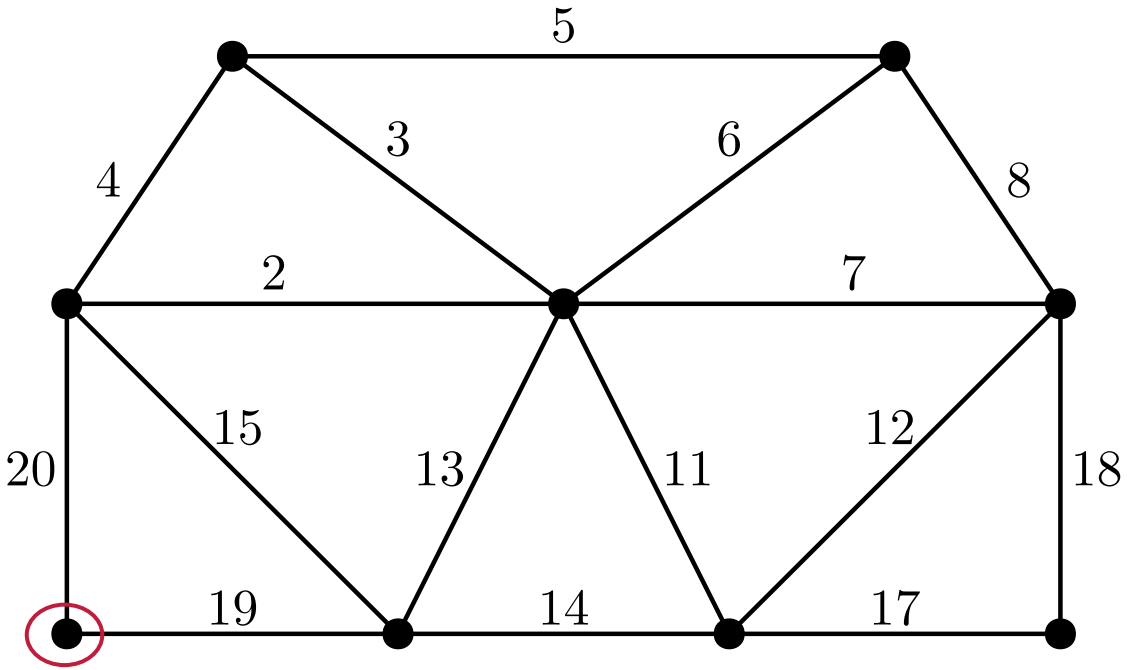
$\Rightarrow$ Algorithmus von Prim

Algorithmus von Prim

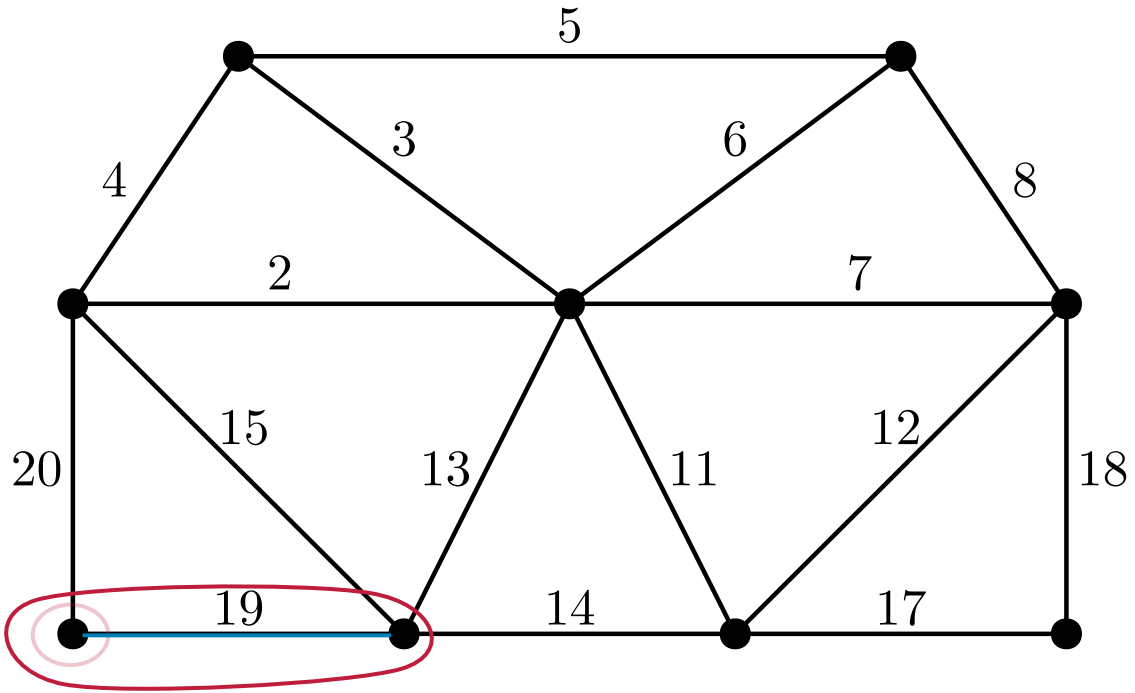
Algorithmus 2.16
Eingabe:
Zshgd. Graph $G = (V, E)$,
Kostenfunktion c: $E \to \mathbb{R}^+$
Ausgabe:
Spannbaum $T = (V, F)$ minimalen Gesamtgewichts

1. Function PRIM(G, c)
2. Wähle Knoten $v \in V$.
3. Setze $X := \{v\}$ und $F := \emptyset$.
4. While $X \neq V$ do
5. Wähle Kante $e \in E(X, V \setminus X)$ mit kleinstem Gewicht.
6. Setze $w := (V \setminus X) \cap e, X := X \cup \{w\}$ und $F := F \cup \{e\}$.
7. Return $T := (V, F)$

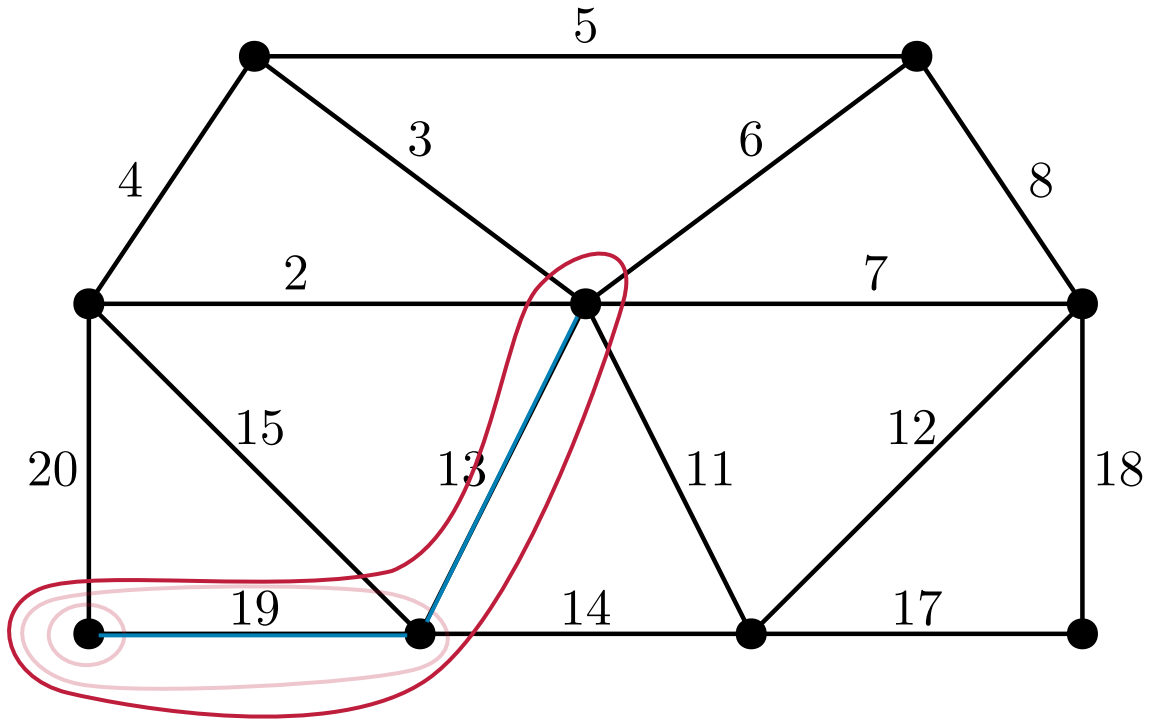
Algorithmus von Prim - Beispiel



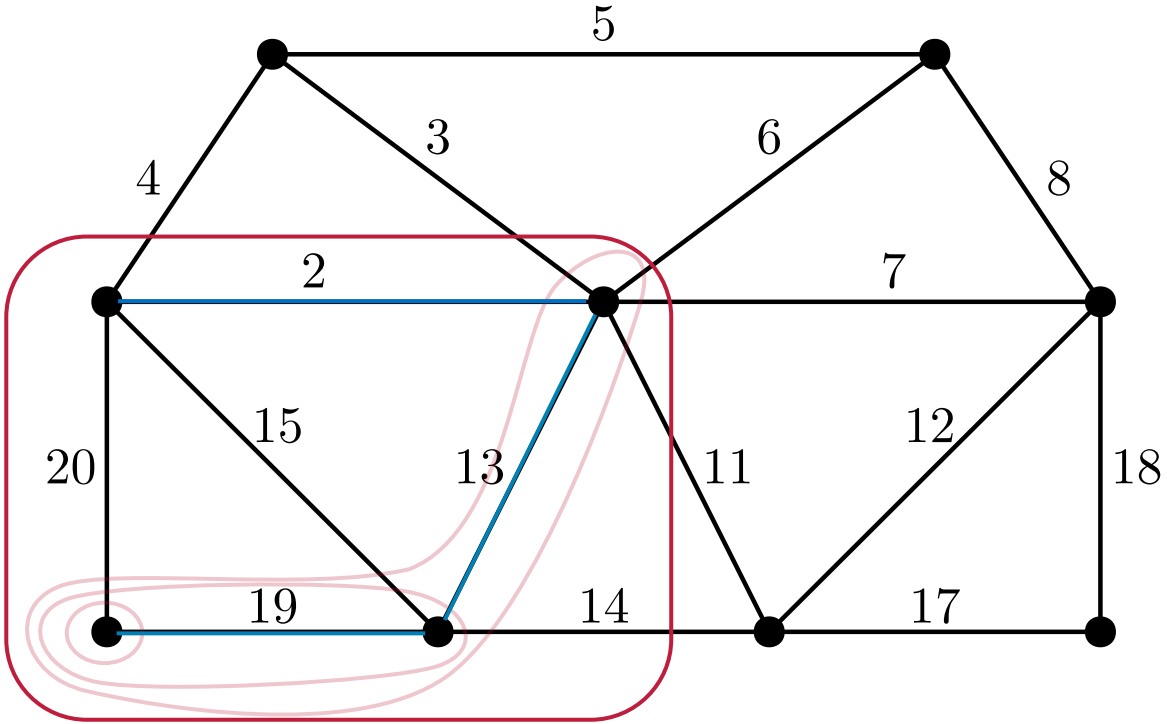
Algorithmus von Prim - Beispiel



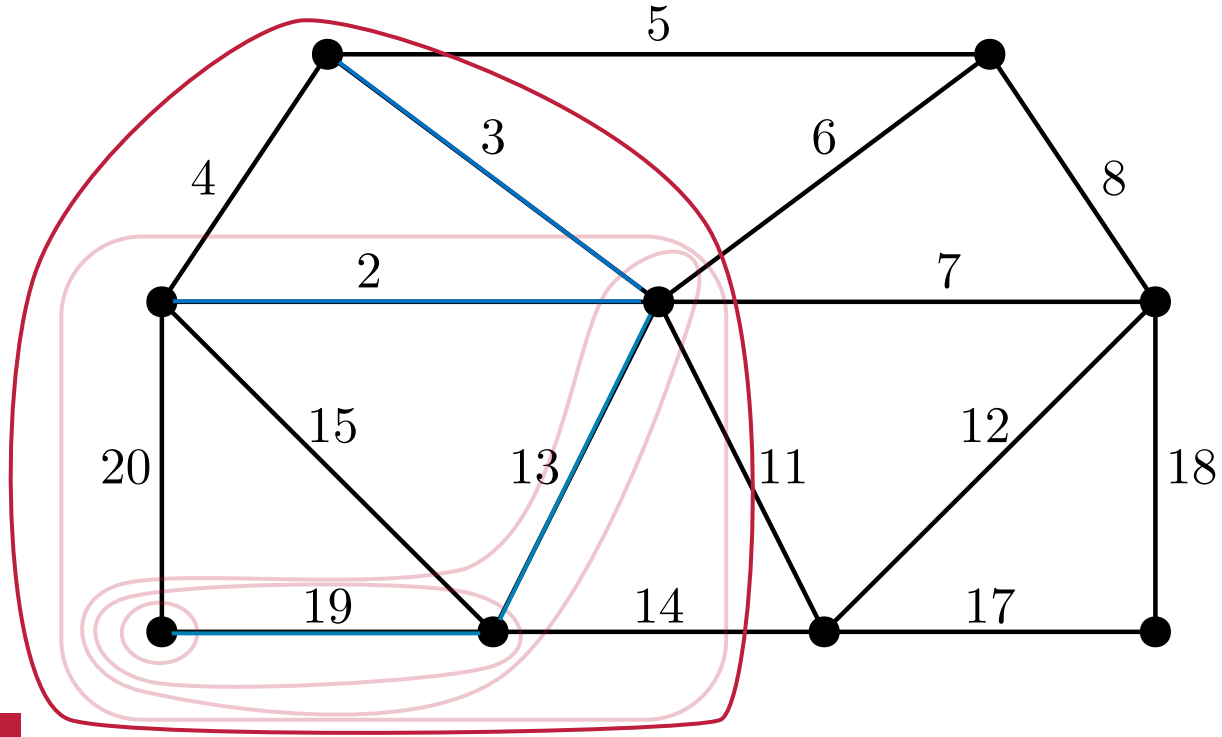
Algorithmus von Prim - Beispiel



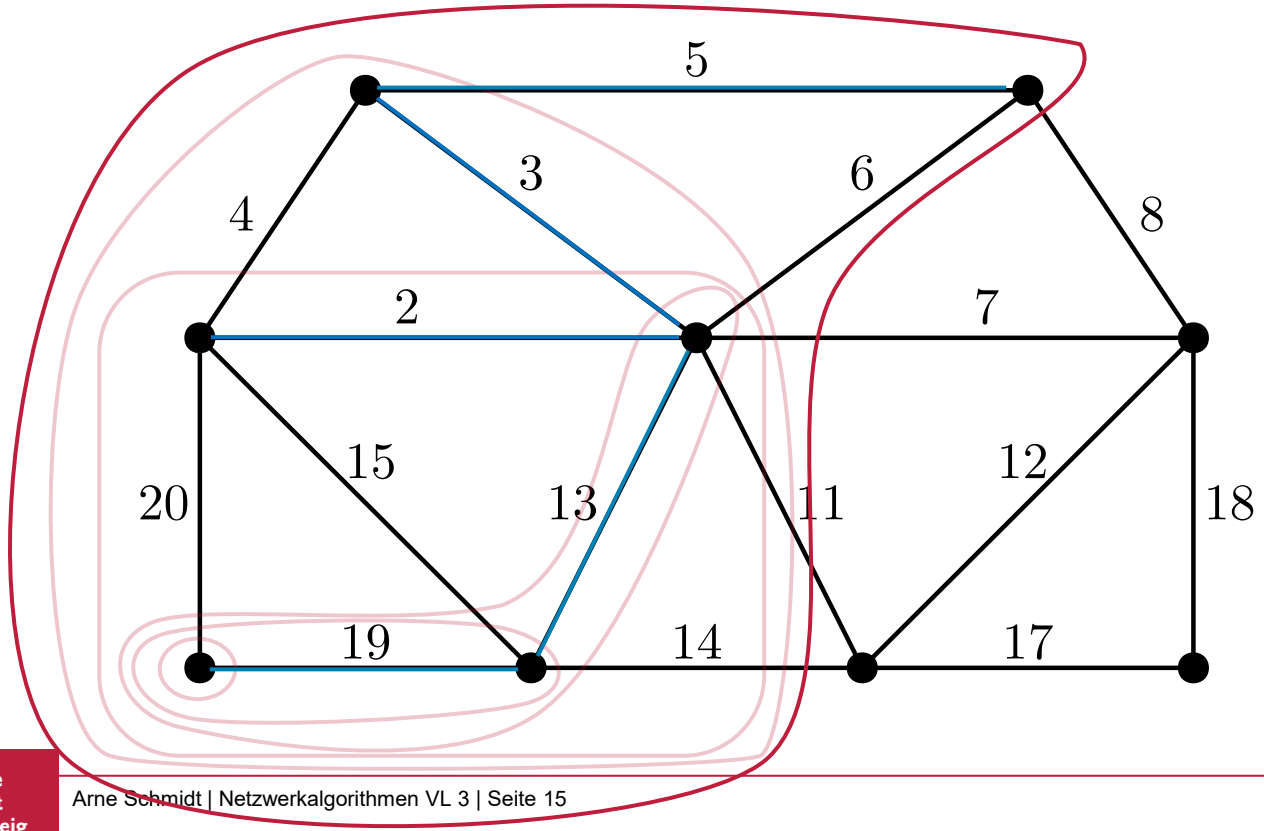
Algorithmus von Prim - Beispiel



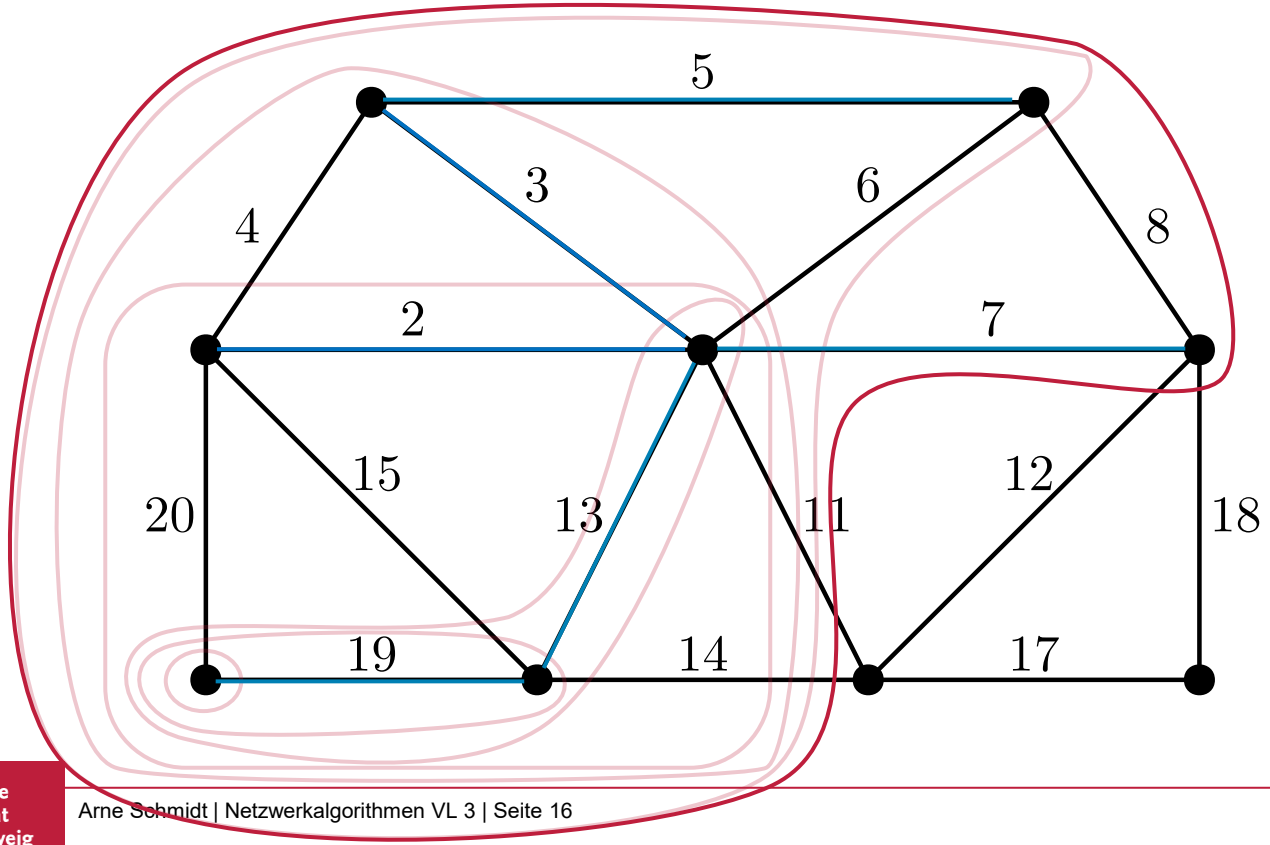
Algorithmus von Prim - Beispiel



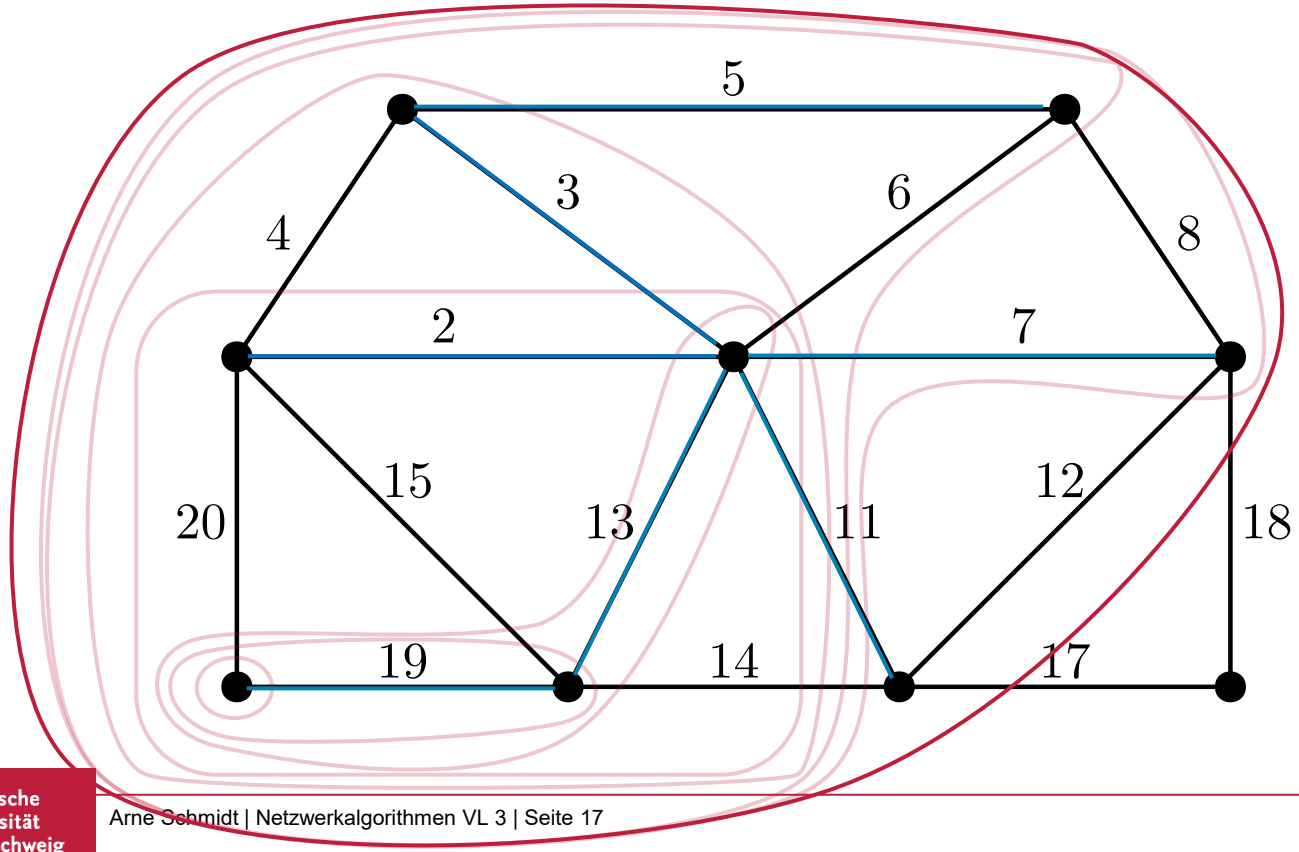
Algorithmus von Prim - Beispiel



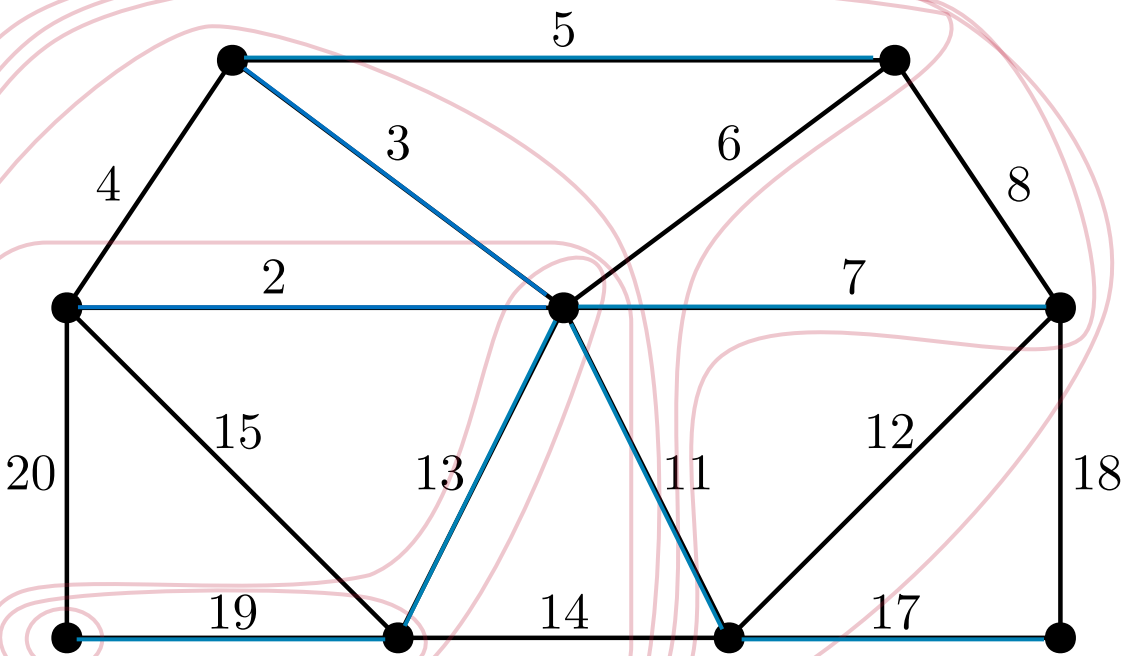
Algorithmus von Prim - Beispiel



Algorithmus von Prim - Beispiel



Algorithmus von Prim - Beispiel



Algorithmus von Prim

Satz 2.17
Prim‘s Algorithmus löst Problem 2.1 (MST) korrekt in Zeit $O(m + n \log n)$ Zeit.

Beweisskizze für Laufzeit:

Merke zu jedem nicht erreichten Knoten die kleinste Kante nach X. Das sind maximal n viele Kandidaten.
Suche in jeder Iteration besten Knoten heraus.
⇒ Fibonacci-Heaps: $\boldsymbol{O(\log n)}$ **Zeit pro Iteration**

Nach Hinzufügen des Knotens w zu X aktualisiere Kandidaten mit Kanten von w:
$\boldsymbol{O\big(\delta(w)\big)}$ **Zeit pro Iteration**

⇒ $O(m + n \log n)$ über alle Iterationen.

4.11 Andere Strukturen

4.11.1 Fibonacci-Heaps

Die Fibonacci-Heaps sind eine Heap-Struktur mit sehr schneller *amortisierter* Zugriffszeit (s. Tabelle 4.3) und werden als Prioritätenwarteschlange benutzt, d.h. jeder Knoten besitzt eine *Priorität* (auch Schlüssel), sodass Knoten mit hoher (bzw. niedriger) Priorität schnell entnommen werden können. Genauer besteht ein Fibonacci-Heap aus einer Liste von Bäumen mit den Eigenschaften von Min-Heaps. Jeder Knoten im Heap besteht aus einem Schlüssel und einem weiteren Bit, welches markiert, ob für diesen Knoten bereits ein Kind entfernt wurde. Zusätzlich werden, wie für die Liste von Bäumen, für die Kinder jedes Knotens eine Doppelt-Verkettete-Liste verwendet. Der Zeiger für den Startknoten des Fibonacci-Heaps zeigt immer auf die kleinste Wurzel aller Bäume.

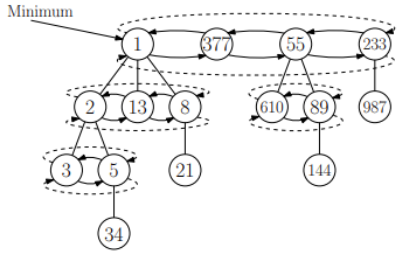


Abbildung 4.34: Ein Beispiel für einen Fibonacci-Heap.

2.3 Matroide

Unabhängigkeitssysteme und Matroide

Problem 2.18
Gegeben:
- Mengensystem $(E, \mathcal{I})$ mit $\mathcal{I} \subseteq 2^E$
- Kostenfunktion $c: E \to \mathbb{R}^+$

Gesucht:
Element $X \in \mathcal{I}$ mit $\sum_{e \in X} c(e)$ maximal/minimal

Definition 2.19
Ein Mengensystem $(E, \mathcal{I})$ mit $\mathcal{I} \subseteq 2^E$ ist ein *Unabhängigkeitssystem*, wenn
(1) $\emptyset \in \mathcal{I}$
(2) $I \in \mathcal{I}, I' \subset I \Rightarrow I' \in \mathcal{I}$

Elemente aus $\mathcal{I}$ heißen *unabhängig*.
Maximal unabhängige Mengen heißen *Basen*.

Ein Unabhängigkeitssystem heißt *Matroid*, wenn gilt:
(3) Wenn $I_1, I_2 \in \mathcal{I}$ und $|I_1| < |I_2|$, dann existiert $x \in I_2 \setminus I_1$, sodass $I_1 \cup \{x\} \in \mathcal{I}$

Matroide Beispiele

Beispiel 1:
Sei E_1 die Menge der Spaltenvektoren der Länge n über den Körper $\mathbb{K}$ und sei $\mathcal{I}_1 \coloneqq \{F \subseteq E_1 \mid \text{Vektoren in } F \text{ sind linear unabhängig}\}$.

Behauptung:
$(E_1, \mathcal{I}_1)$ ist ein Matroid.

Beispiel 2:
Gegeben sei ein Graph $G = (V, E_2)$ uns sei $\mathcal{I}_2 \coloneqq \{F \subseteq E_2 \mid F \text{ enthält keinen Kreis}\}$.

Behauptung:
$(E_2, \mathcal{I}_2)$ ist ein Matroid.

Matroide Beispiel 1

Sei E_1 die Menge der Spaltenvektoren der Länge n über den Körper $\mathbb{K}$ und sei $\mathcal{I}_1 \coloneqq \{F \subseteq E_1 \mid \text{Vektoren in } F \text{ sind linear unabhängig}\}$.

Zu zeigen: $(E_1, \mathcal{I}_1)$ ist ein Matroid

(1) $\emptyset \in \mathcal{I}_1$ (0 Vektoren sind linear unabhängig)

(2) Teilmengen unabhängiger Vektoren sind immer noch unabhängig.

(3) Zu Zeigen: $F_1, F_2 \in \mathcal{I}_1$ und $|F_1| < |F_2| \Rightarrow \exists x \in F_2 \setminus F_1$: $F_1 \cup \{x\} \in \mathcal{I}$

Da F_1 und F_2 linear unabhängig und $|F_1| < |F_2|$, gilt $\dim\big(\text{span}(F_1)\big) < \dim\big(\text{span}(F_2)\big)$.
Also existiert $x \in F_2$ mit $x \notin \text{span}(F_1)$.
Damit ist $F_1 \cup \{x\}$ linear unabhängig, also in $\mathcal{I}_1$

Greedy für gewichtete Matroide

Algorithmus 2.20

Eingabe:

Matroid $M = (E, \mathcal{I})$

Kostenfunktion c: $E \to \mathbb{R}^+$

Ausgabe:

Basis B mit $\sum_{e \in X} c(e)$ minimal.

1. Function MatroidGreedy(M, c)
2. Sortiere Elemente aufsteigend nach Gewicht, sodass $c(e_1) \leq c(e_2) \leq \cdots \leq c(e_m)$
3. Setze $B := \emptyset$
4. For $i = 1$ to m do
5. if $B \cup \{e_i\} \in \mathcal{I}$ then
6. $B := B \cup \{e_i\}$
7. Return B

Greedy für gewichtete Matroide

Satz 2.21
Algorithmus 2.20 bestimmt für ein Matroid M mit Kostenfunktion c eine Basis mit optimalem Gewicht.

Man kann sogar folgendes zeigen:

Satz 2.22
Sei $(E, \mathcal{I})$ ein Unabhängigkeitssystem. Algorithmus 2.20 bestimmt für $(E, \mathcal{I})$ mit Kostenfunktion c genau dann eine Basis optimalen Gewichts, wenn $(E, \mathcal{I})$ ein Matroid ist.

Algorithmus 2.20
Eingabe:
Matroid $M = (E, \mathcal{I})$
Kostenfunktion c: $E \to \mathbb{R}$
Ausgabe:
Basis B mit $\sum_{e \in X} c(e)$ minimal.

```
1.  Function MatroidGreedy(M, c)
2.      Sortiere Kanten aufsteigend nach Gewicht,
            sodass c(e_1) ≤ c(e_2) ≤ ⋯ ≤ c(e_m)
3.      Setze B := ∅
4.      For i = 1 to m do
5.          if B ∪ {e_i} ∈ I then
6.              B := B ∪ {e_i}
7.      Return B
```

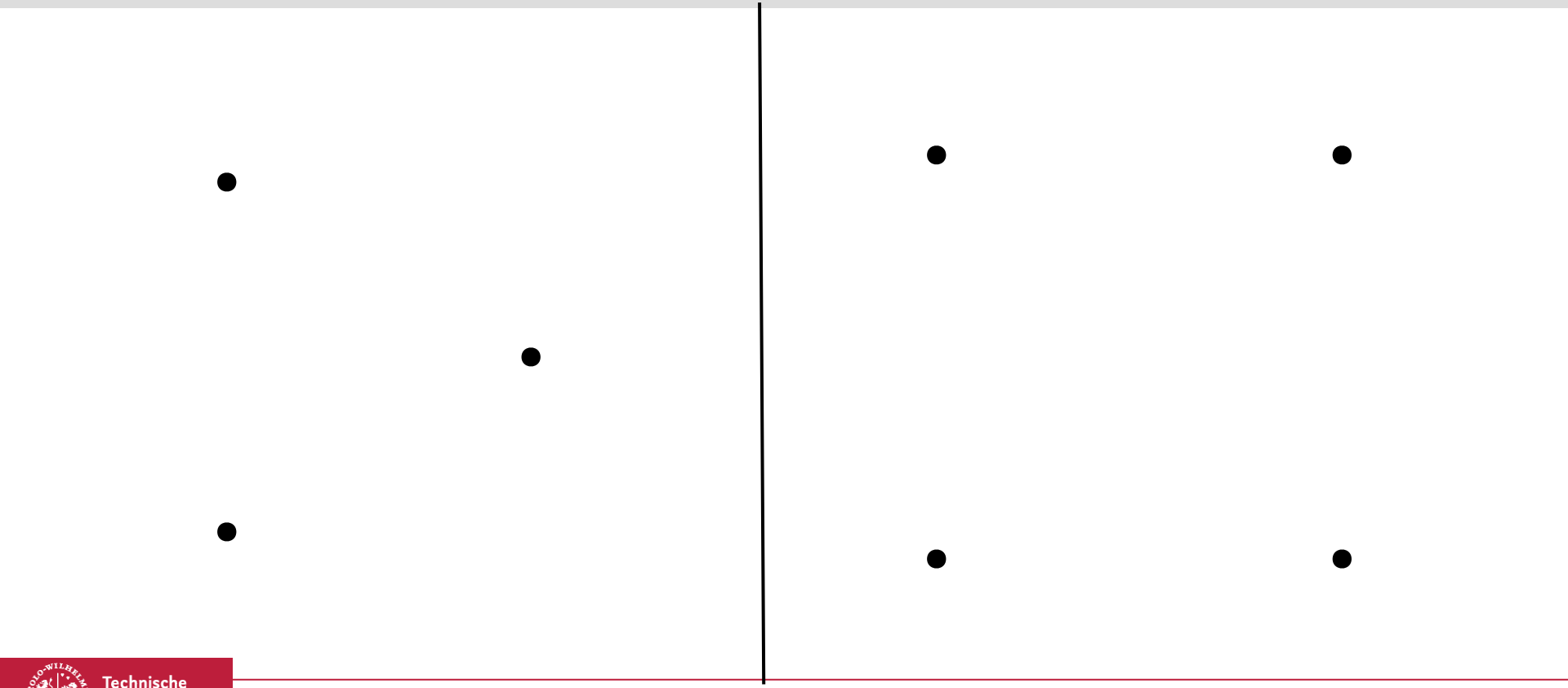
2.4 Rück- und Ausblick

Graphenklasse	Prim	Kruskal*
Dünne Graphen ($m \in O(n \log n)$)	$O(n \log n)$	$O(n \log n)$
Dichte Graphen ($m \in \Omega(n \log n)$)	$O(m)$	$O(m \log n)$
Kleine, ganzzahlige Gewichte	$O(m + n \log n)$	$O(m \log n)$
Geometrisch	$O(n^2)$	$O(n^2 \log n)$

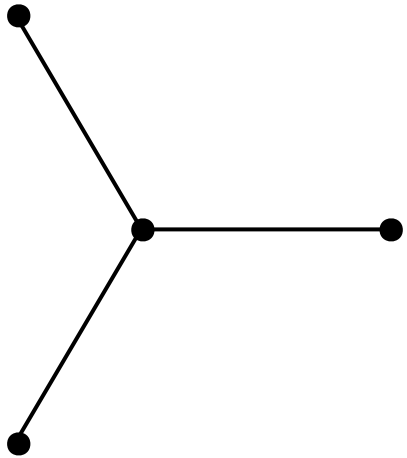
*: Sind Kanten vorsortiert, performt Kruskal besser! (vgl. große Übung)

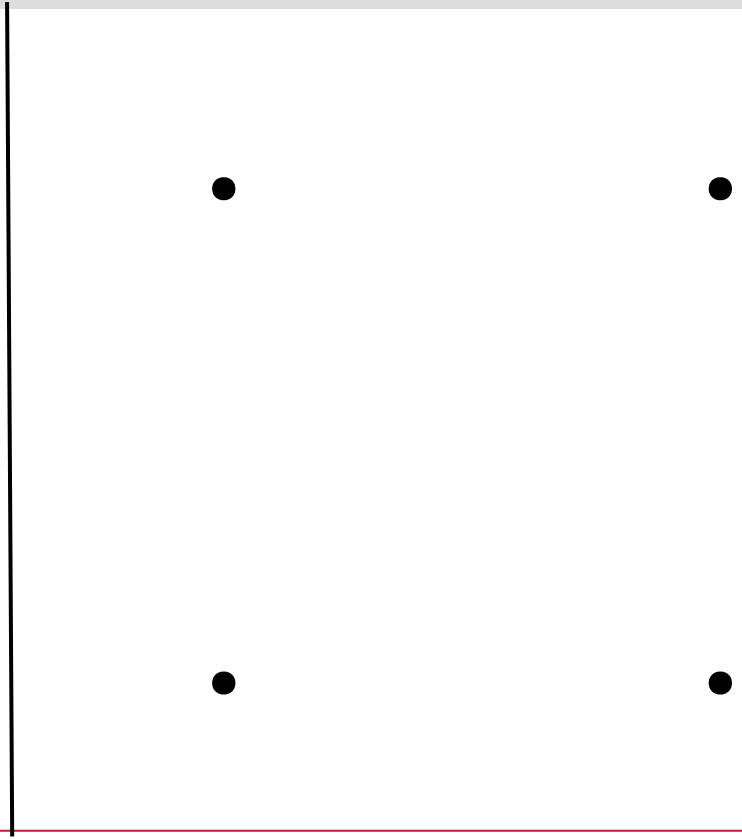
Was passiert, wenn Knoten in Terminals und Verteiler unterteilt sind?
Wichtiger Unterschied: Nur Terminals müssen verbunden werden!

Steinerbäume - Geometrisch



Steinerbäume - Geometrisch





https://youtu.be/BVbIRM01UTs

Quiz!

Mentimeter