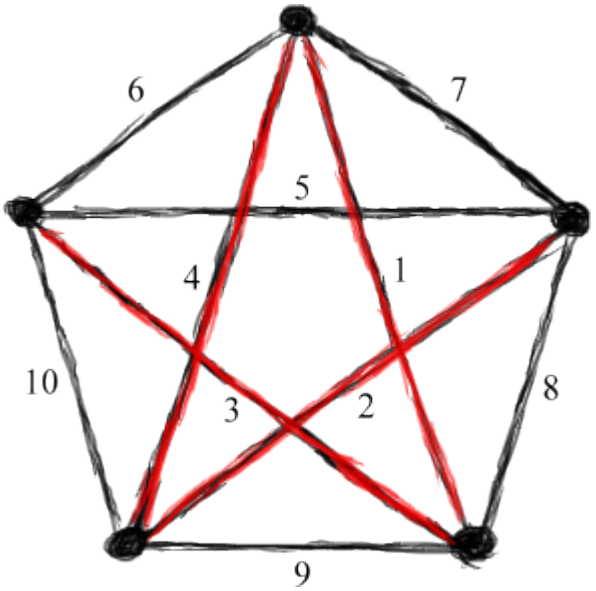
Minimal spannende Bäume



Fakultät für Informatik
Otto-von-Guericke-Universität

Inhalt

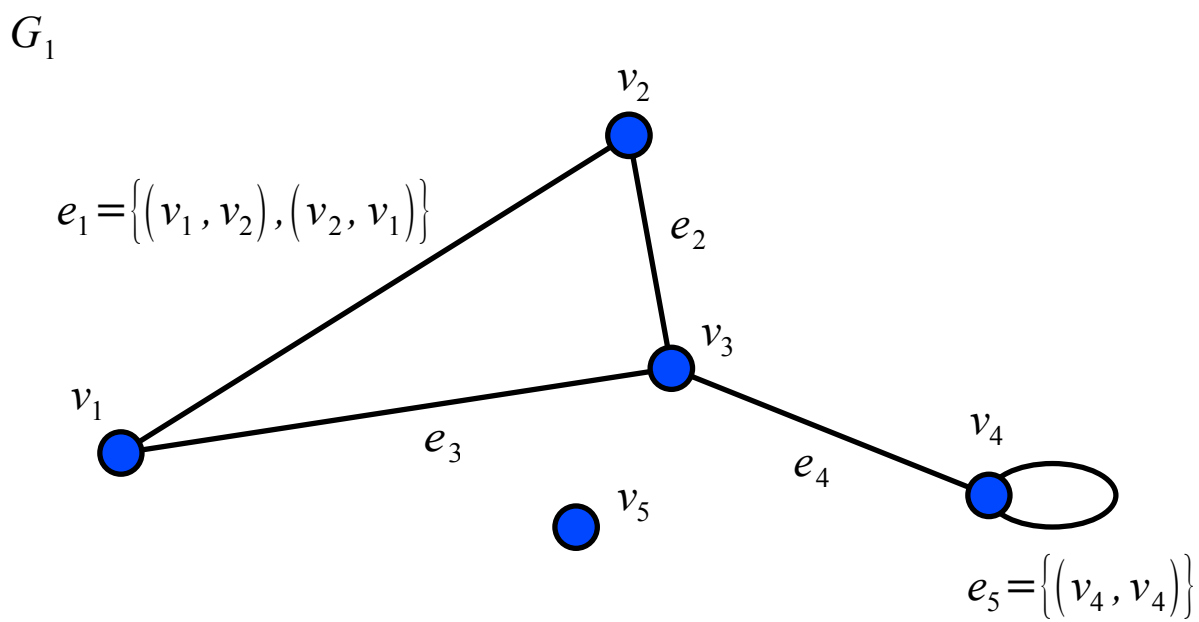
- Ungerichtete Graphen
  - Definition
  - Wege
  - Untergraphen
  - Kantengewichtete Graphen
- Bäume
  - Definition und Eigenschaften
  - Minimal spannende Bäume
    - Algorithmen zur Bestimmung minimal Spannender Bäume

Definition ungerichteter Graphen

Ein ungerichteter Graph G ist eine algebraische Struktur $G=(V,E)$, wobei V eine endliche nicht leere Menge und E mit $E \subseteq V \times V$ eine symmetrische Relation auf V ist $(\forall x, y \in V : x\,E\,y \Rightarrow y\,E\,x)$. Wir bezeichnen V als Menge von Knoten (vertices) und E als Menge von Kanten (edges).

Beispiel:

$G_1=(V_1,E_1)=(\{v_1,v_2,v_3,v_4,v_5\},$
$\{(v_1,v_2),(v_2,v_1),(v_1,v_3),(v_3,v_1),(v_2,v_3),(v_3,v_2),(v_3,v_4),(v_4,v_3),(v_4,v_4)\})$.



Definition von Wegen in einem Graphen

Ein Weg $w \in V^n$ zwischen zwei Knoten $x, y \in V$ ist ein n-Tupel $w = (x_1, x_2, \cdots, x_n)$ mit $x_1 = x$, $x_n = y$ und $n \geq 2, n \in \mathbb{N}$ genau dann, wenn $(x_i, x_{i+1}) \in E$ mit $1 \leq i \leq n-1, i \in \mathbb{N}$ gilt. Wir nennen x Anfangs- und y Endknoten des Weges w. $W(G)$ ist die Menge aller Wege von G.

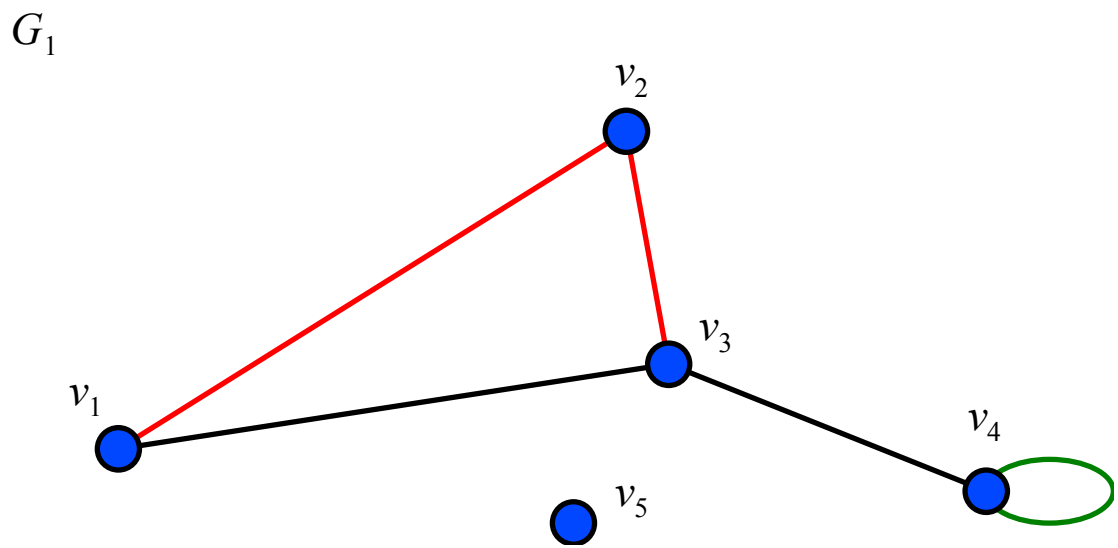
Wir bezeichnen einen Weg $(x_1, x_2, \cdots, x_n) \in W(G)$ als **Kreis**, wenn Anfangs- und Endknoten mit $x_1 = x_n$ gleich sind und $x_i \neq x_j$ mit $i \neq j$ und $i, j \in \{k \mid 1 \leq k \leq n-1, k \in \mathbb{N}\}$ gilt. Einen Graphen G, der keine Kreise enthält, nennen wir **kreisfrei**.

Darüber hinaus heißt ein Graph $G = (V, E)$ **zusammenhängend**, wenn zwischen je zwei unterschiedlichen Knoten x und y aus V ein Weg $(x, \cdots, y)$ aus $W(G)$ existiert, also

$$(\forall x, y \in V : x \neq y \Rightarrow \exists (x, \cdots, y) \in W(G))$$

gilt

Beispiel zu Wegen in einem Graphen



Weg: $(v_1, v_2, v_3) \in W(G_1)$

Kreis: $(v_4, v_4) \in W(G_1)$

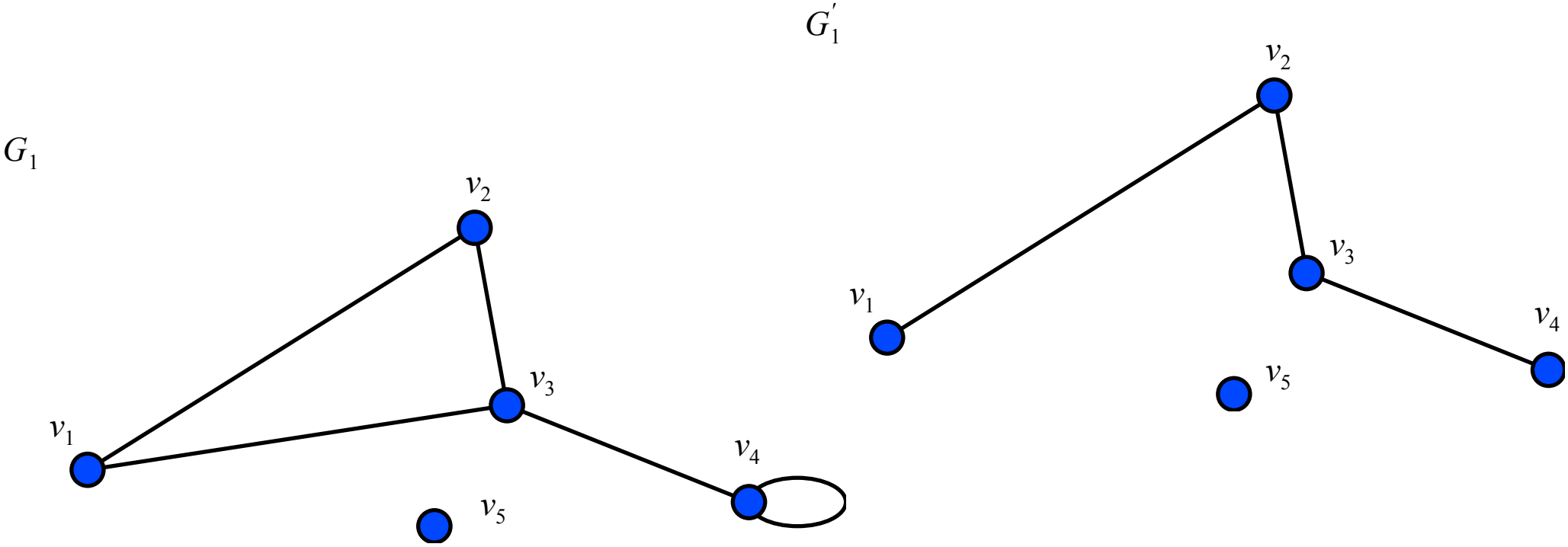
Nicht-zusammenhängend: $G_1 = (E_1, V_1)$

Zusammenhängend: $(V_1 \setminus \{v_5\}, E_1)$

Definition von Untergraphen von Graphen

Wir bezeichnen einen Graphen $G' = (V', E')$ als Untergraphen des Graphen $G = (V, E)$, wenn $V' \subseteq V$ und $E' \subseteq E$ gelten. Falls $V' = V$ gilt, dann heißt G' spannend.

Beispiel:



G'_1 ist ein spannender Untergraph von G_1

Definition von kantengewichteten Graphen

Ein kantengewichteter Graph $G=(V,E,w)$ *ist ein Graph* (V,E) *mit einer Abbildung* $w:E\rightarrow\mathbb{R}^+$, *wobei* $w((x,y))=w((y,x))$. *Wir nennen die Funktion w Gewicht (weight).*

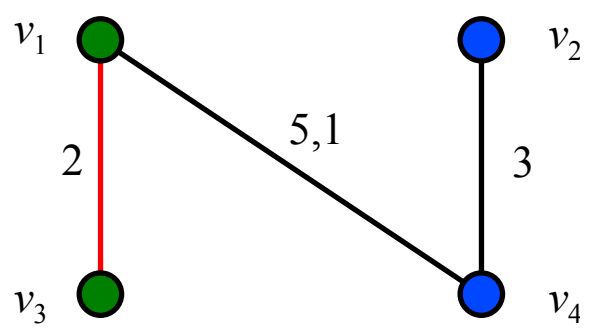
Gewicht von G: $w_\Sigma(G)=\sum_{(x,y)\in E,\, x\neq y}\frac{w((x,y))}{2}+\sum_{(x,x)\in E} w((x,x))$

Beispiel:

$G_3=(\{v_1,v_2,v_3,v_4\},\{(v_1,v_3),(v_3,v_1),(v_1,v_4),(v_4,v_1),(v_2,v_4),(v_4,v_2)\},w)$

$w(v_1,v_3)=w(v_3,v_1)=2,\ w(v_1,v_4)=w(v_4,v_1)=5{,}1,\ w(v_2,v_4)=w(v_4,v_2)=3$

G_3



$w_\Sigma(G_3)=10{,}1$

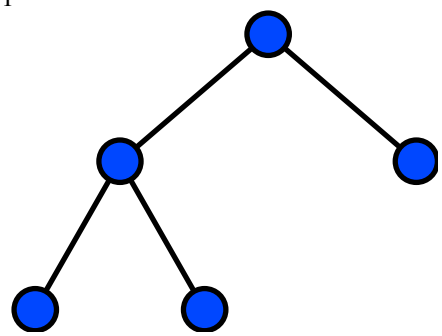
Definition von Bäumen und spannenden Bäumen

Ein Baum (tree) T ist ein zusammenhängender und kreisfreier Graph (V, E).

Ein spannender Baum S ist ein Baum, der ein spannender Untergraph eines Graphen ist.

Beispiel:

T_1



Satz zur Existenz von spannenden Bäumen

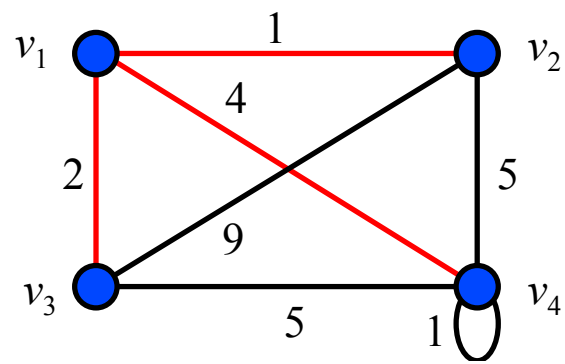
Es existiert dann und nur dann ein spannender Baum in einem Graphen G, wenn G zusammenhängend ist.

Definition von minimal spannenden Bäumen

Ein minimal spannender Baum (Minimum Spanning Tree – MST) M eines zusammenhängenden kantengewichteten Graphen $G=(V,E,w)$ *ist ein spannender Baum* (V,E') *vom Graphen* (V,E)*, falls kein anderer spannender Baum* $(V,\overline{E})$ *von* (V,E) *derart existiert, dass* $w_\Sigma((V,\overline{E},\overline{w}))<w_\Sigma((V,E',w'))$ *mit* $\forall\, e\in\overline{E}:\overline{w}(e)=w(e)$ *und* $\forall\, e\in E':w'(e)=w(e)$ *gilt.*

Beispiel:

G_5, M_1



Ein Graph G_5 mit minimal spannenden Baum M_1. Die Kanten von M_1 sind rot markiert.

Satz zur Existenz von minimal spannenden Bäumen

Es existiert dann und nur dann ein minimal spannender Baum in einem kantengewichteten Graphen G, wenn G zusammenhängend ist .

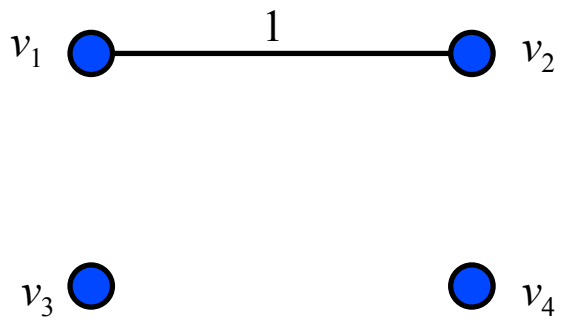
Algorithmus von Kruskal

Eingabe: zusammenhängender kantengewichteter Graph $G=(V,E,w)$
Ausgabe: minimal spannender Baum $M=(V,E')$ *von* G

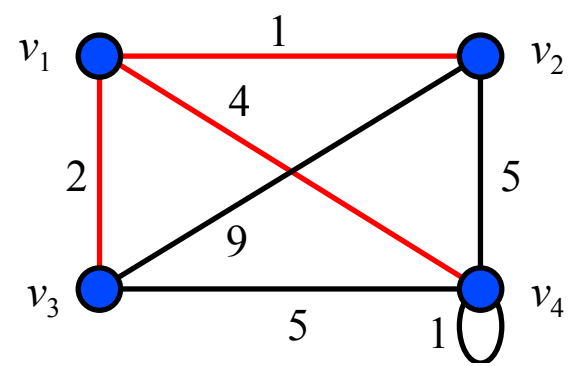
B 1: $E' := \emptyset;\ \overline{E} := E;$
B 2: *solange* $\overline{E} \neq \emptyset$
{
B 3: $K := \{(x,y),(y,x)\}$ *mit* $(x,y) \in \{e \in \overline{E} \mid \neg \exists e' \in \overline{E} : w(e') < w(e)\};$
B 4: $\overline{E} := \overline{E} \setminus K;$
B 5: *wenn* $(V, E' \cup K)$ *kreisfrei, dann*
{
B 6: $E' := E' \cup K;$
}
}

Beispiel zum Algorithmus von Kruskal

1.



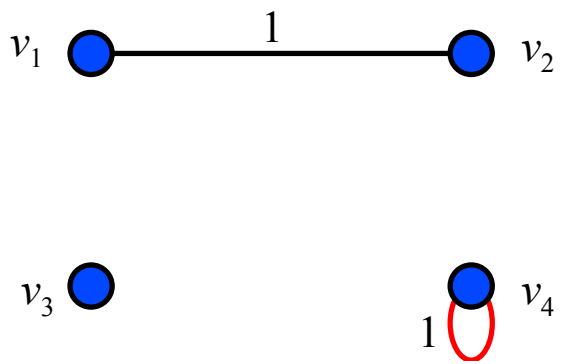
G_5, M_1



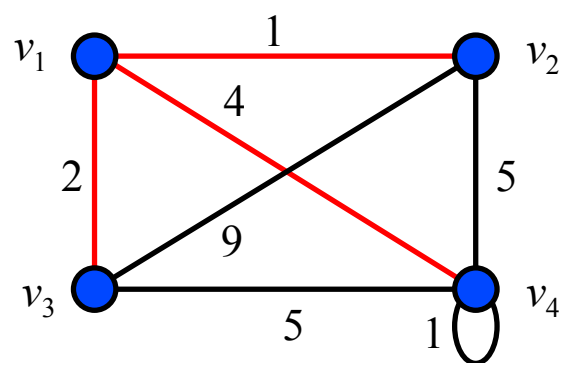
$K = \{(v_1, v_2), (v_2, v_1)\}$ $\overline{E} = E \setminus \{(v_1, v_2), (v_2, v_1)\}$ $E' = \{(v_1, v_2), (v_2, v_1)\}$

Beispiel zum Algorithmus von Kruskal (Fortsetzung)

2.



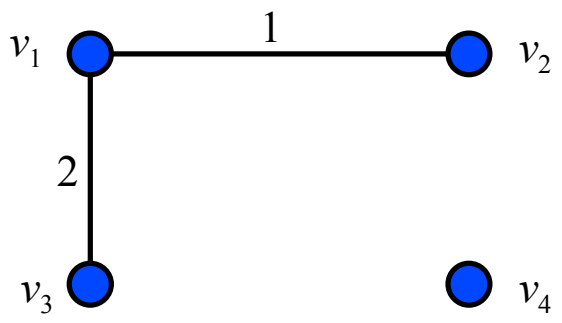
G_5, M_1



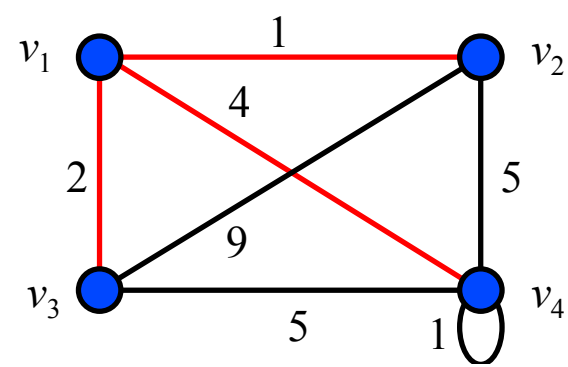
$K = \{(v_4, v_4)\}$ $\quad$ $\overline{E} = (E \setminus \{(v_1, v_2), (v_2, v_1)\}) \setminus \{(v_4, v_4)\}$ $\quad$ $E' = \{(v_1, v_2), (v_2, v_1)\}$

Beispiel zum Algorithmus von Kruskal (Fortsetzung)

3.



G_5, M_1



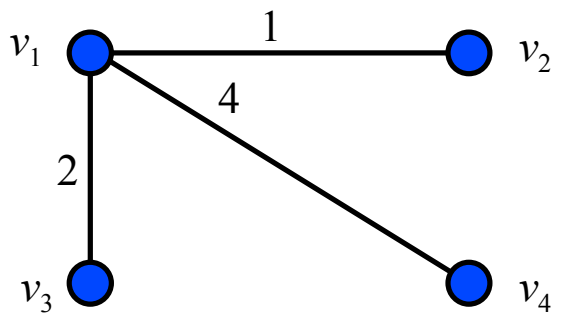
$$K = \{(v_1, v_3), (v_3, v_1)\}$$

$$\overline{E} = \left(\left(E \setminus \{(v_1, v_2), (v_2, v_1)\}\right) \setminus \{(v_4, v_4)\}\right) \setminus \{(v_1, v_3), (v_3, v_1)\}$$

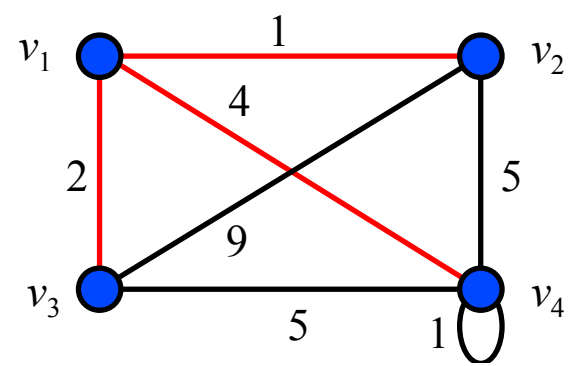
$$E' = \{(v_1, v_2), (v_2, v_1), (v_1, v_3), (v_3, v_1)\}$$

Beispiel zum Algorithmus von Kruskal (Fortsetzung)

4.



G_5, M_1



$$K = \{(v_1, v_4), (v_4, v_1)\}$$

$$\overline{E} = \left(\left(\left(E \setminus \{(v_1, v_2), (v_2, v_1)\}\right) \setminus \{(v_4, v_4)\}\right) \setminus \{(v_1, v_3), (v_3, v_1)\}\right) \setminus \{(v_1, v_4), (v_4, v_1)\}$$

$$E' = \{(v_1, v_2), (v_2, v_1), (v_1, v_3), (v_3, v_1), (v_1, v_4), (v_4, v_1)\}$$

Eigenschaften von Kruskals Algorithmus

Der Algorithmus von Kruskal ist korrekt.

Der Algorithmus von Kruskal lässt sich so implementieren, dass sein Aufwand $O(|E| \cdot log|V|)$ ist.

Algorithmus von Prim

Algorithmus von Prim:
Eingabe: zusammenhängender kantengewichteter Graph $G=(V,E,w)$
Ausgabe: minimal spannender Baum $M=(V,E')$ *von* G

B 1: $V':=\{v\}$ *mit* $v\in V$; $E':=\emptyset$;

B 2: *solange* $V'\neq V$

{

$$(x,y)\in\{(p,q)\in E \mid p\in V'\wedge q\in V\setminus V'\wedge \neg\exists(p',q')\in E: p'\in V'\wedge q'\in V\setminus V'\wedge w((p',q'))<w((p,q))\}$$

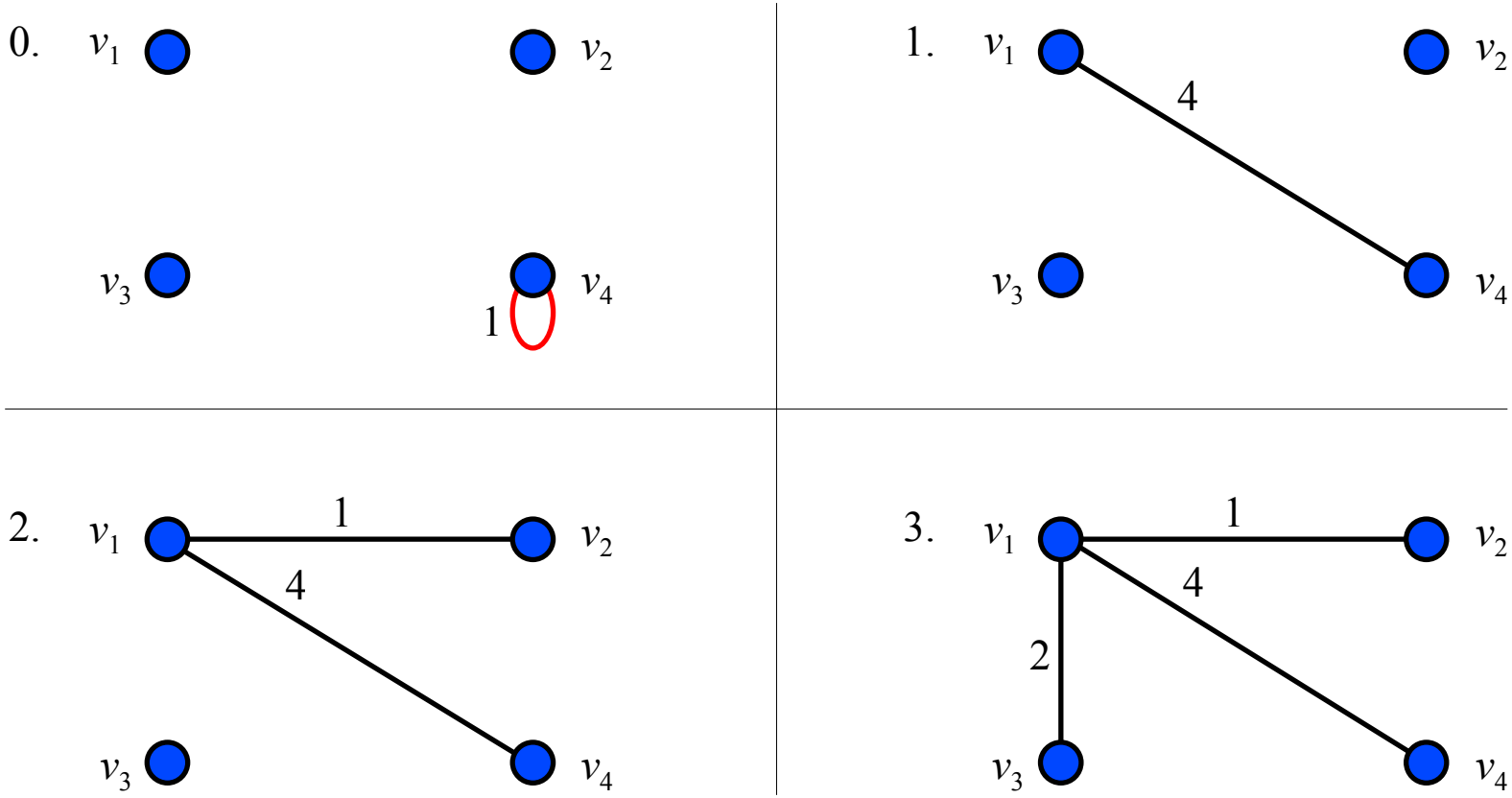
B 3: $E':=E'\cup\{(x,y),(y,x)\}$;

B 4: $V':=V'\cup\{y\}$;

}

Beispiel zum Algorithmus von Prim

$E' = \{(v_4, v_1), (v_1, v_4)\}$ $V' = \{v_4, v_1\}$



$E' = \{(v_4, v_1), (v_1, v_4), (v_1, v_2), (v_2, v_1)\}$
$V' = \{v_4, v_1, v_2\}$

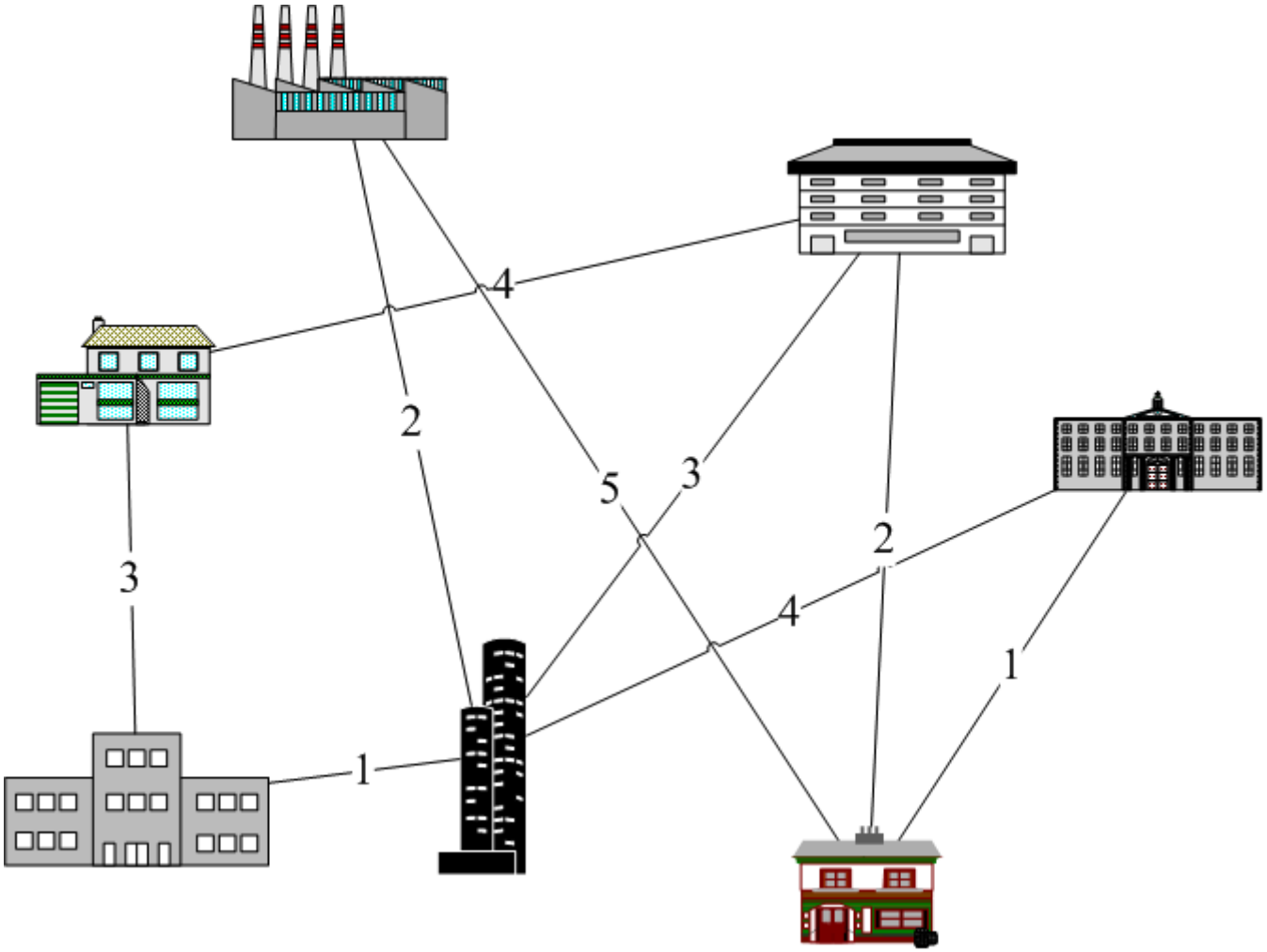
$E' = \{(v_4, v_1), (v_1, v_4), (v_1, v_2), (v_2, v_1), (v_1, v_3), (v_3, v_1)\}$
$V' = \{v_4, v_1, v_2, v_3\} = V$

Eigenschaften von Algorithmus Prim

Der Algorithmus von Prim ist korrekt.

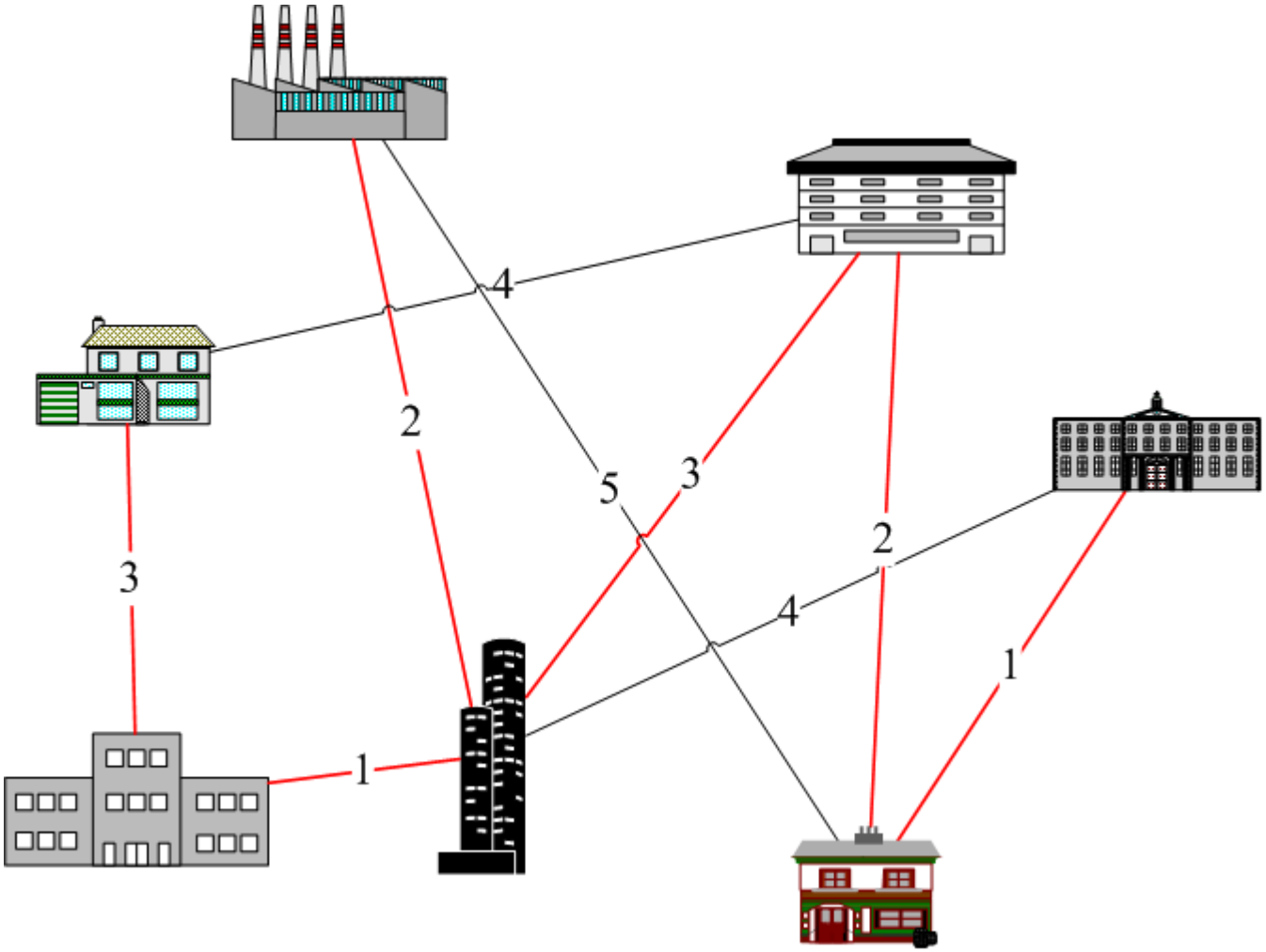
Der Algorithmus von Prim kann mit Hilfe eines Fibonacci-Heaps in $O\left(|E|+|V|\cdot log|V|\right)$ realisiert werden.

Praktische Anwendung der Algorithmen: Verlegen von Netzwerkkabeln



Gesucht sind minimale Kosten für das Netzwerk.

Praktische Anwendung der Algorithmen: Verlegen von Netzwerkkabeln (Fort.)



Ein minimal spannende Baum ist rot markiert.

Literatur

http://www-e.uni-magdeburg.de/harbich/study.htm#graphs_mst