Algorithmen und Datenstrukturen (Th. Ottmann und P. Widmayer)

Folien: Minimaler Spannbaum

Autor: Sven Schuierer

Institut für Informatik

Georges-Köhler-Allee

Albert-Ludwigs-Universität Freiburg

1 Minimal Spannende Bäume

Gegeben:

1. Graph $G = (V, E)$ (zusammenhängend, endlich)
2. Gewichtungsfunktion $g : E \longrightarrow \mathbb{R}_+$ (Gewicht oder Länge einer Kante)

Spannender Baum T für G:

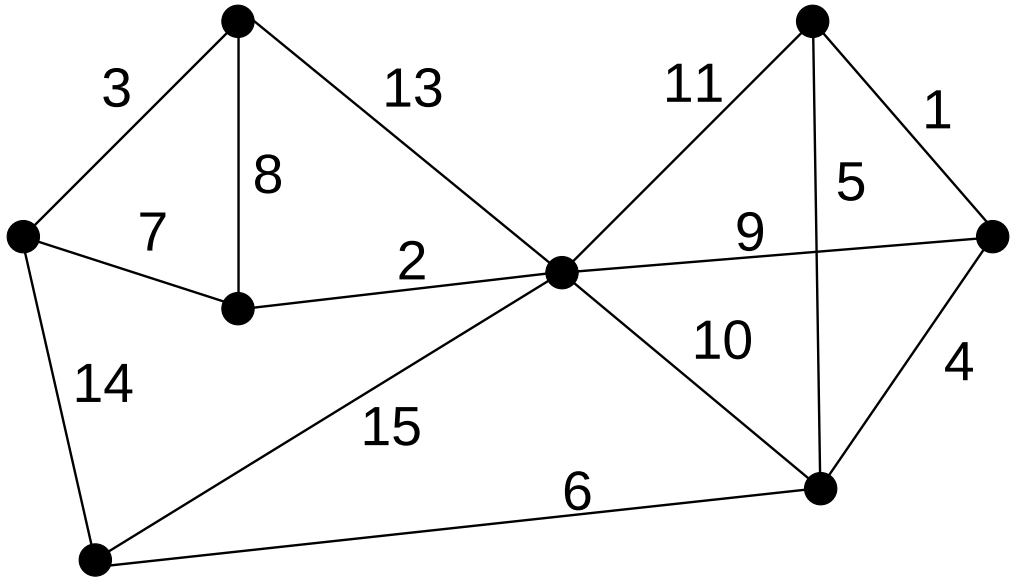
Zusammenhängender, azyklischer Teilgraph von G, der alle Knoten miteinander verbindet

Kosten von T: Summe der Gewichte aller Kanten in T

Aufgabe:

Berechne einen spannenden Baum mit minimalen Kosten (minimum spanning tree oder mst)

Beispiel



Prim

MST1 (Verfahren von Prim)

Input: Ein Graph $G = (V, E)$ mit Kantengewichtsfunktion g;

Output: Ein minimal spannender Baum $mst(G)$ von G;

1 wähle einen beliebigen Knoten $s \in V$

2 setze $U := \{s\}$ und $T := \emptyset$

3 while $V \setminus U \neq \emptyset$ do

4 wähle kürzeste Kante (u, v), so daß $u \in U$ und $v \in V \setminus U$

5 $T := T \cup \{(u, v)\}$

6 $U := U \cup \{v\}$

7 return T

Kruskal

2. Idee:

1. Betrachte zu Beginn Kollektion von einelementigen Knotenmengen

2. Vergrößere Knotenmengen sukzessive durch Vereinigung von je zwei Mengen, falls die Mengen durch eine Kante minimalen Gewichts miteinander verbunden werden

3. Das Verfahren endet, wenn die Kollektion nur noch aus einer einzigen Menge (der Knotenmenge V) besteht

Kruskal

MST2

Input: Ein Graph $G = (V, E)$ mit g;

Output: $mst(G)$;

1 $\mathcal{F} := \{\{v\} \mid v \in V\}$

2 while $|\mathcal{F}| > 1$ do

3 wähle kürzeste Kante (u, v), so daß $T_u, T_v \in \mathcal{F}$, $T_u \neq T_v$, und $u \in T_u$, $v \in T_v$
/* $U = T_u$ */

4 $T_{uv} := T_u \cup T_v \cup \{(u, v)\}$

5 $\mathcal{F} := (\mathcal{F} \setminus \{T_u, T_v\}) \cup \{T_{uv}\}$

6 return Baum in $\mathcal{F}$

Effiziente Implementation:

1. Durchlaufe die Kanten in der Reihenfolge nach zunehmenden Gewichten
2. Prüfe für jede Kante, ob sie zum mst hinzugenommen werden kann oder nicht

Implementation Kruskal

Kruskal

Input: Ein Graph $G = (V, E)$ mit g;

Output: $mst(G)$;

1 $T = \emptyset$

2 bilde die Priority Queue Q aller Kanten in E mit Prioritätsordnung $g(e)$

3 for all $v \in V$ do $v.make\text{-}set()$

4 while $|T| < n - 1$ do

5 $(u, v) = Q.delete\text{-}min()$

6 $U = u.find\text{-}set(); V = v.find\text{-}set()$

7 if $U \neq V$

8 then $U.union(V)$

9 $T = T \cup \{(u, v)\}$

10 return T