Minimale Spannbäume

- Minimale Spannbäume
- Darstellung von gewichteten Graphen
- Eigenschaften Minimaler Spannbäume
- Prims Algorithmus
- Kruskals Algorithmus

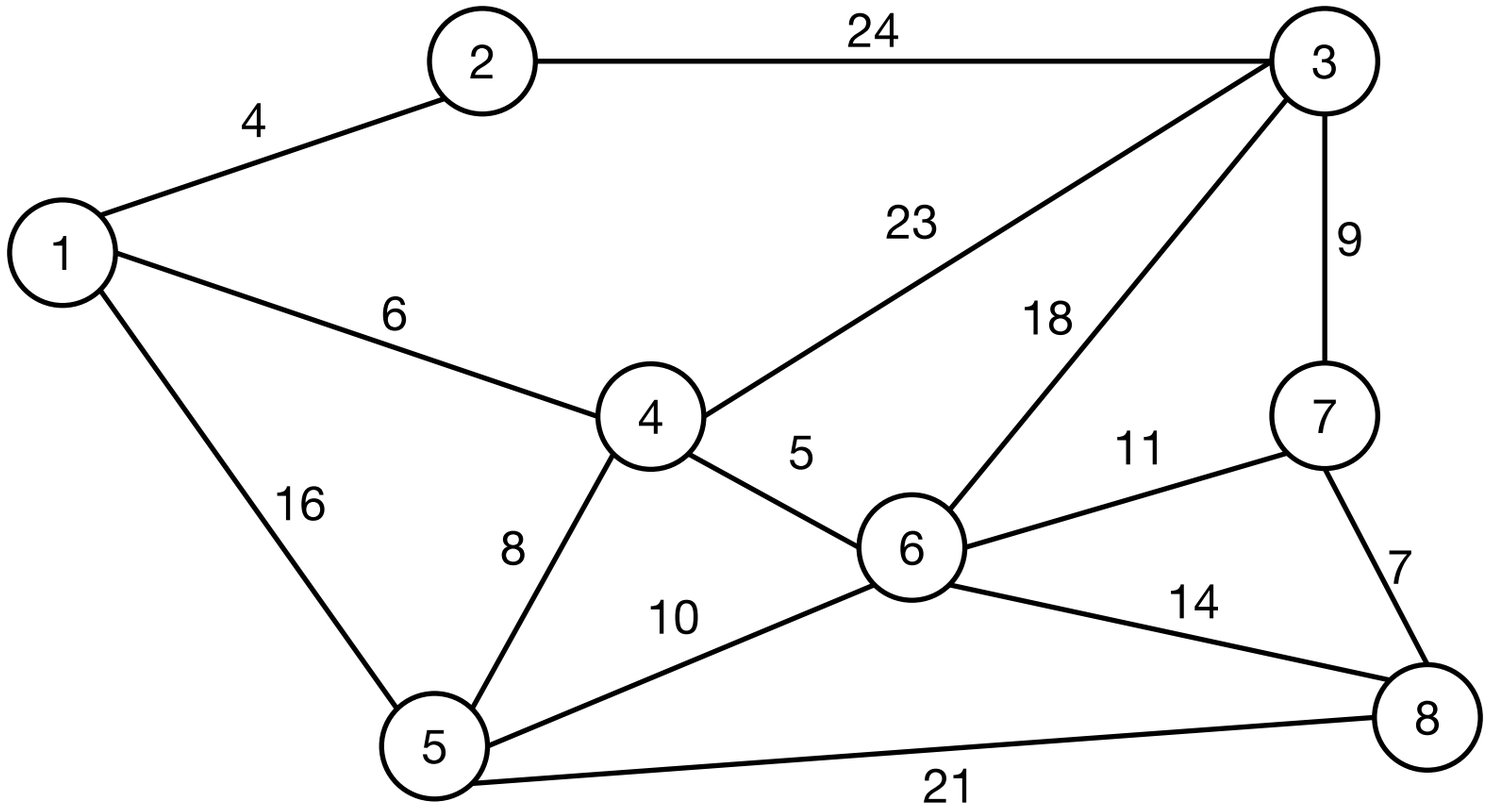
Holger Dell
Folien adaptiert von Philip Bille

Minimale Spannbäume

- Minimale Spannbäume
- Darstellung von gewichteten Graphen
- Eigenschaften Minimaler Spannbäume
- Prims Algorithmus
- Kruskals Algorithmus

Minimale Spannbäume (*minimum spanning trees*)

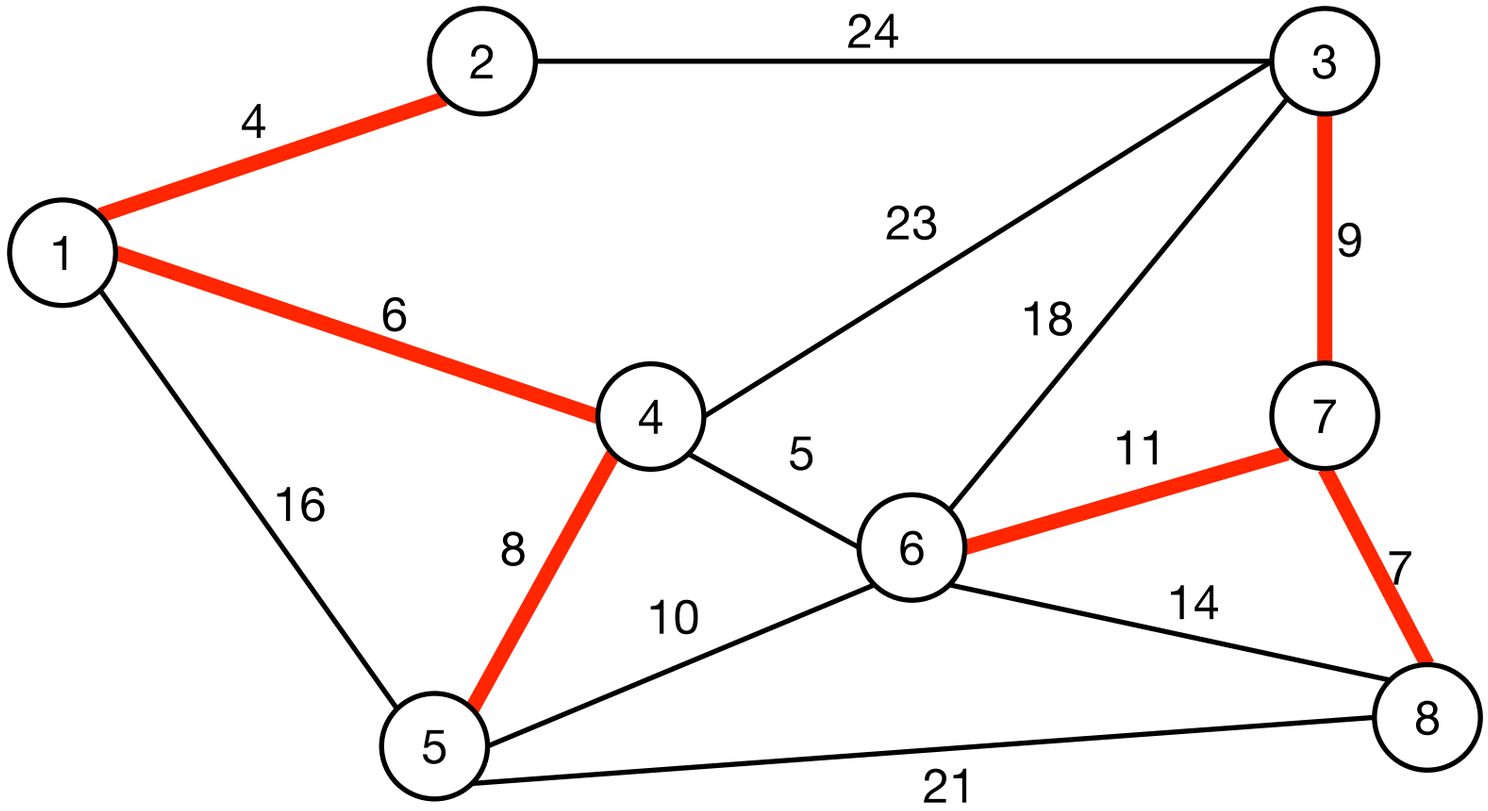
- Gewichtete Graphen. Gewicht w(e) auf jeder Kante e in G.
- Spannbaum. Zusammenhängender und azyklischer Teilgraph T von G, der alle Knoten von G enthält.
- Minimaler Spannbaum (MST). Spannbaum von minimalem Gesamtgewicht.



Graph G

Minimale Spannbäume (*minimum spanning trees*)

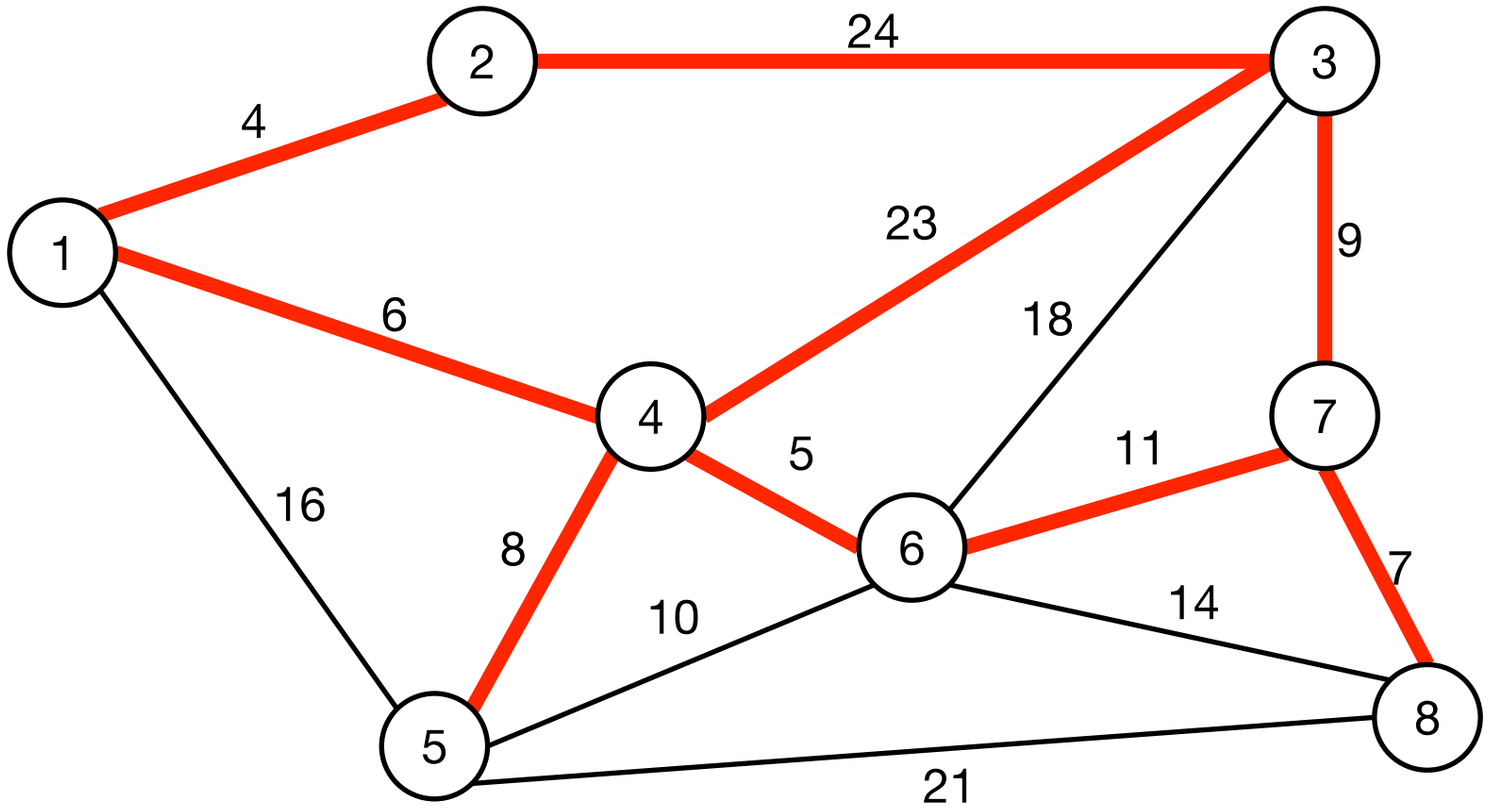
- Gewichtete Graphen. Gewicht w(e) auf jeder Kante e in G.
- Spannbaum. Zusammenhängender und azyklischer Teilgraph T von G, der alle Knoten von G enthält.
- Minimaler Spannbaum (MST). Spannbaum von minimalem Gesamtgewicht.



nicht zusammenhängend

Minimale Spannbäume (*minimum spanning trees*)

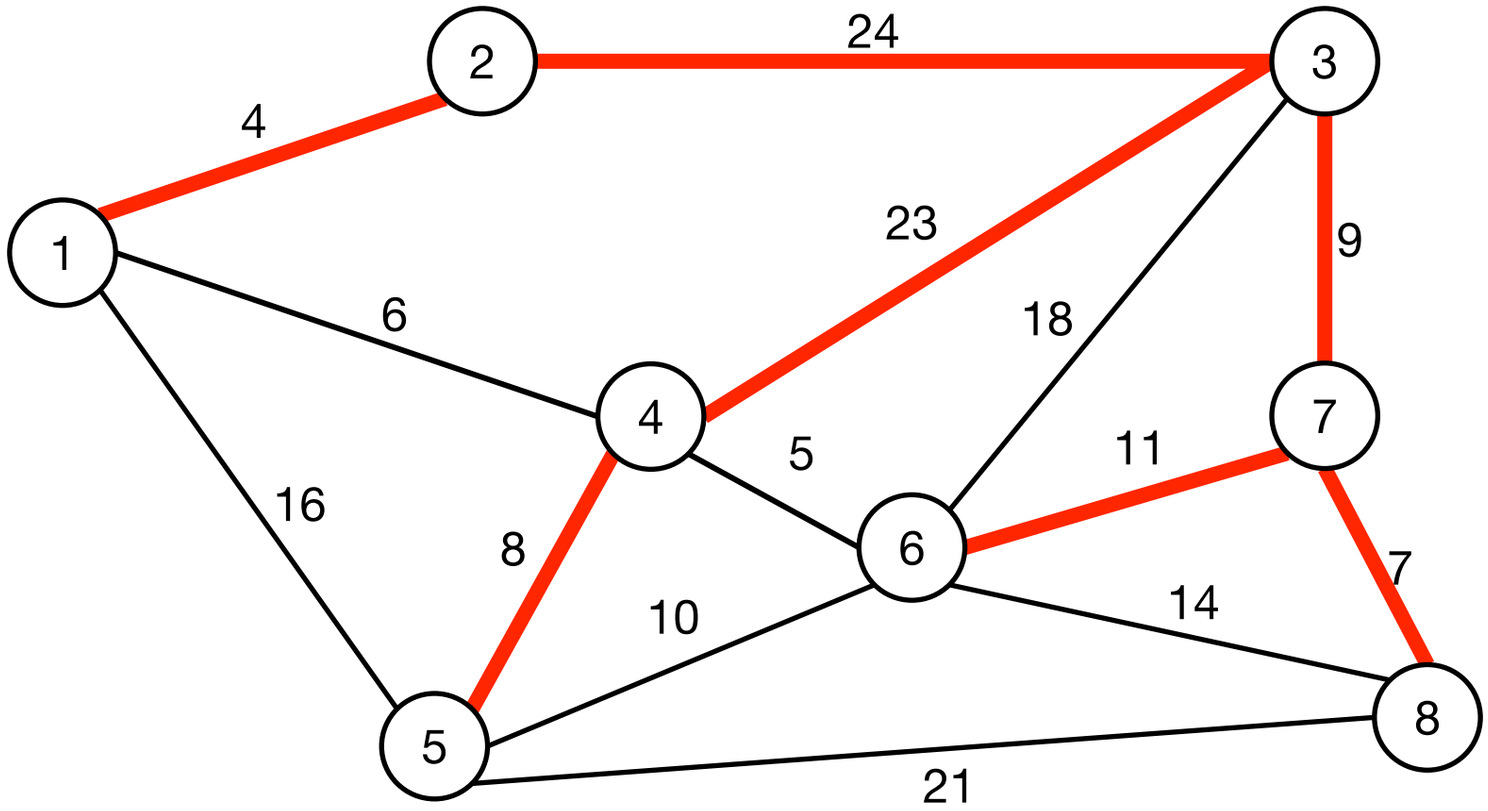
- Gewichtete Graphen. Gewicht w(e) auf jeder Kante e in G.
- Spannbaum. Zusammenhängender und azyklischer Teilgraph T von G, der alle Knoten von G enthält.
- Minimaler Spannbaum (MST). Spannbaum von minimalem Gesamtgewicht.



zusammenhängend und nicht azyklisch

Minimale Spannbäume (*minimum spanning trees*)

- Gewichtete Graphen. Gewicht w(e) auf jeder Kante e in G.
- Spannbaum. Zusammenhängender und azyklischer Teilgraph T von G, der alle Knoten von G enthält.
- Minimaler Spannbaum (MST). Spannbaum von minimalem Gesamtgewicht.

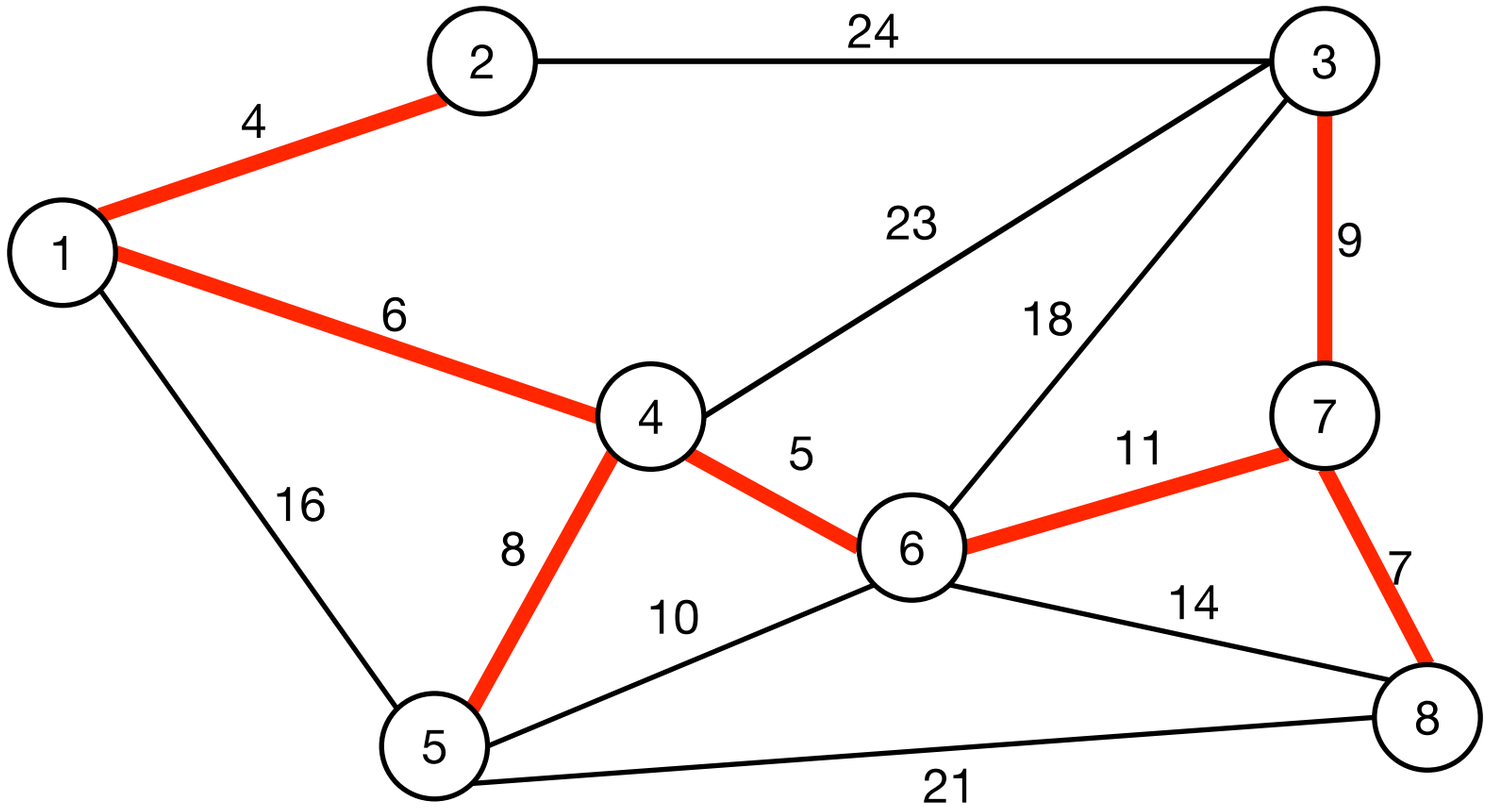


zusammenhängend und azyklisch = Spannbaum

Gesamtgewicht: 4+24+23+8+9+11+7 = 86

Minimale Spannbäume (*minimum spanning trees*)

- Gewichtete Graphen. Gewicht w(e) auf jeder Kante e in G.
- Spannbaum. Zusammenhängender und azyklischer Teilgraph T von G, der alle Knoten von G enthält.
- Minimaler Spannbaum (MST). Spannbaum von minimalem Gesamtgewicht.



Minimaler Spannbaum

Gesamtgewicht: 4+6+8+5+11+9+7 = 50

Anwendungen

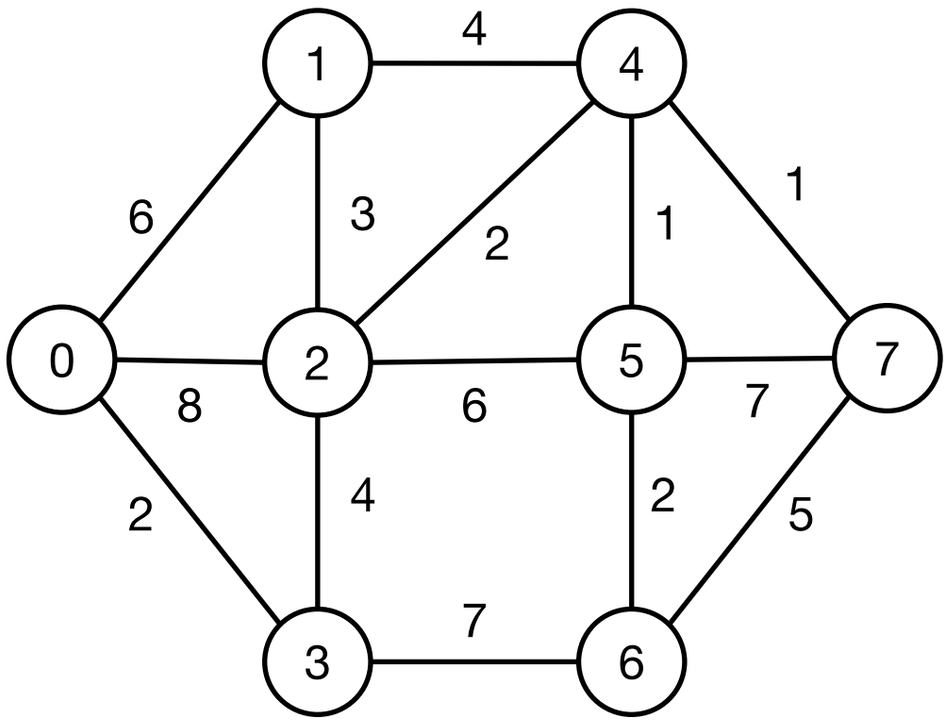
- Entwurf von Netzwerken.
  - Computer, Straßen, Telefon, Strom, Schaltkreis, Kabelfernsehen, Wasser, Glasfaser, …
- Approximationsalgorithmen.
  - Problem des Handlungsreisenden, Steinerbäume.
- Andere Anwendungen.
  - Meteorologie, Kosmologie, Biomedizinische Analyse, Codierung, Bildverarbeitung, ...

Minimale Spannbäume

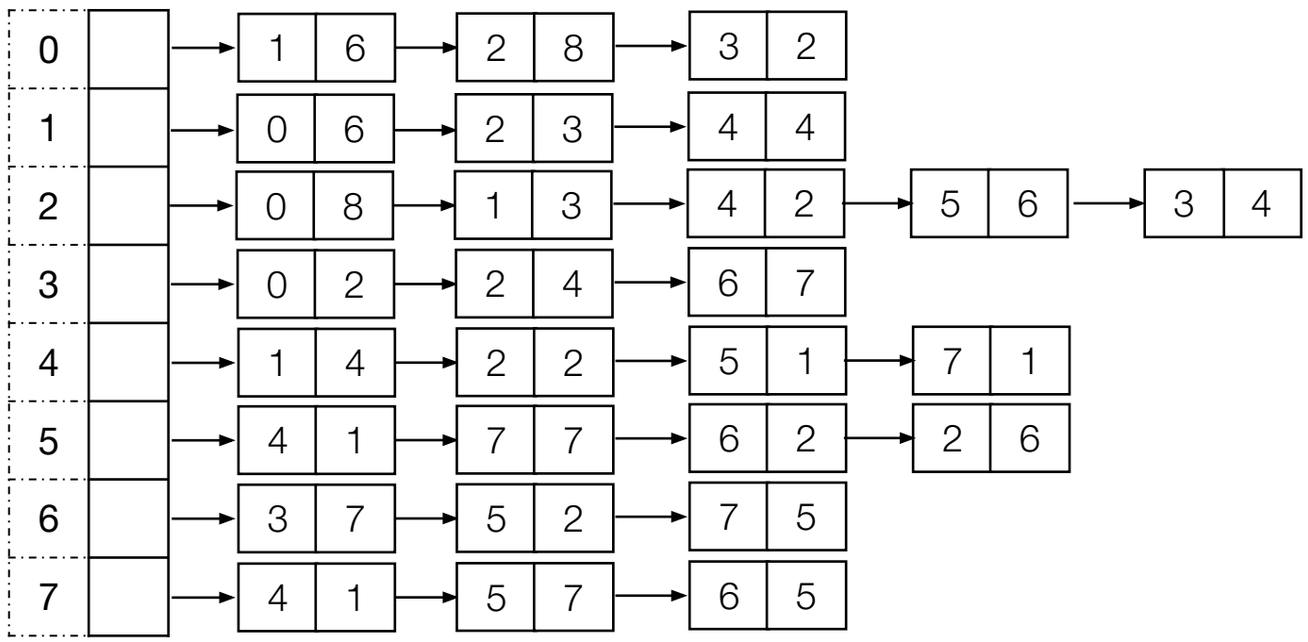
- Minimale Spannbäume
- Darstellung von gewichteten Graphen
- Eigenschaften Minimaler Spannbäume
- Prims Algorithmus
- Kruskals Algorithmus

Darstellung von gewichteten Graphen

- Adjazenzmatrix und Adjazenzliste.
- Ähnlich wie für gerichtete Graphen.



	0	1	2	3	4	5	6	7
0	0	6	8	2	0	0	0	0
1	6	0	3	0	4	0	0	0
2	8	3	0	4	2	6	0	0
3	2	0	4	0	0	0	7	0
4	0	4	2	0	0	1	0	1
5	0	0	6	0	1	0	2	7
6	0	0	0	7	0	2	0	5
7	0	0	0	0	1	7	5	0



Minimale Spannbäume

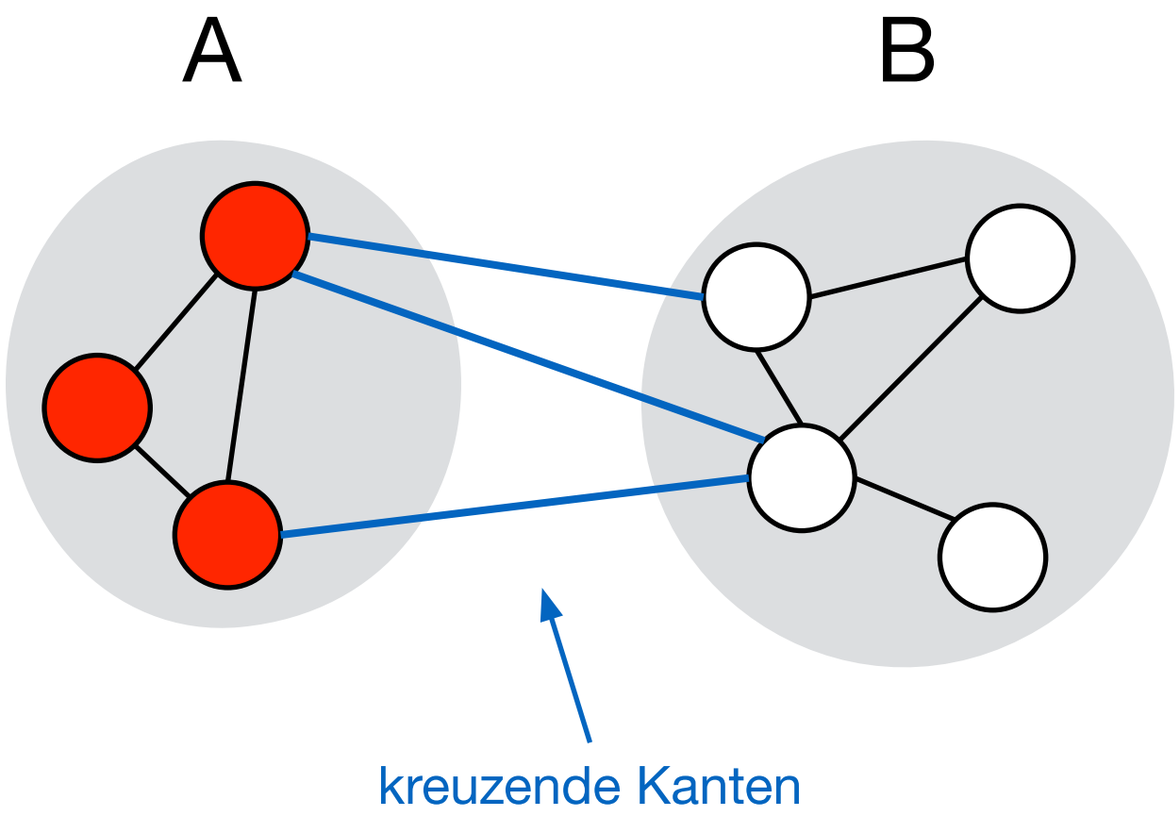
- Minimale Spannbäume
- Darstellung von gewichteten Graphen
- Eigenschaften Minimaler Spannbäume
- Prims Algorithmus
- Kruskals Algorithmus

Eigenschaften Minimaler Spannbäume

- vereinfachende Annahme:
  - Alle Kantengewichte sind unterschiedlich.
  - G ist zusammenhängend.
- $\Longrightarrow$ MST existiert und ist eindeutig.

Schnitte und Kreuzende Kanten (Definition)

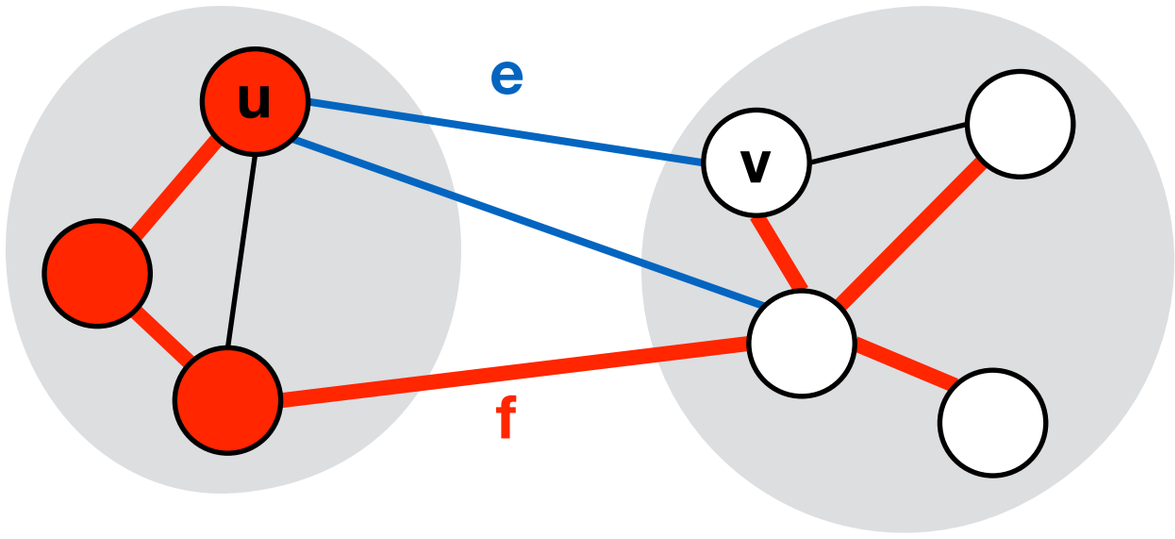
- Ein **Schnitt** (A, B) ist eine Partition der Knoten in zwei nicht-leere Knotenmengen.
- Eine **kreuzende** Kante ist eine Kante, die einen Endpunkt in A und einen in B hat.



Schnitteigenschaft (*cut property*)

- Schnitteigenschaft. Für jeden Schnitt ist die leichteste kreuzende Kante im MST.

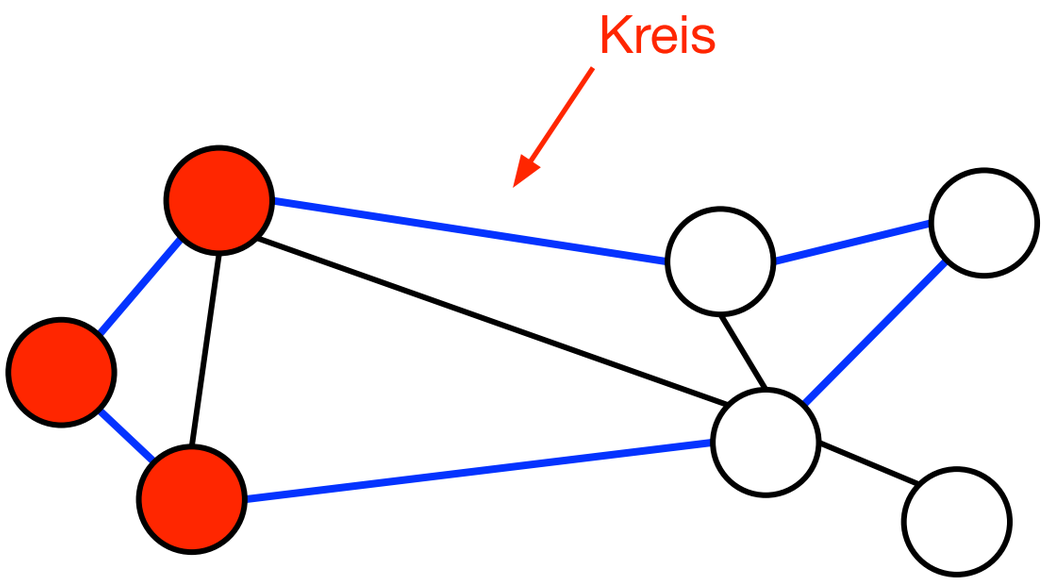
- Beweis durch Widerspruch.
  - Sei T ein MST, sei (A, B) ein Schnitt, und sei e die leichteste kreuzende Kante.
  - Angenommen e = {u, v} ist nicht in T enthalten.
  - Dann gibt es in T einen (u,v)-Weg, der eine andere kreuzende Kante f benutzt:

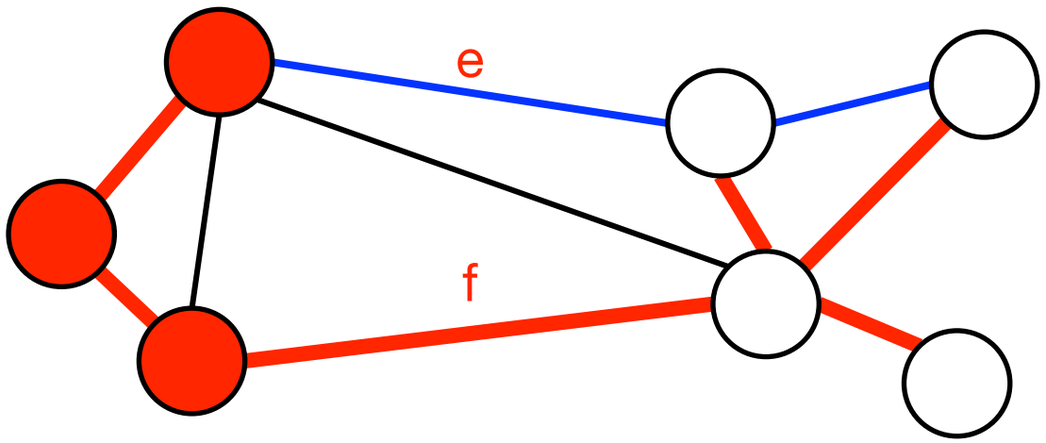


- Erzeuge einen neuen Teilgraph T’: Lösche f aus T und füge e ein.
- Der so erzeugte Teilgraph T’ ist ein Spannbaum mit kleinerem Gewicht.

Kreiseigenschaft

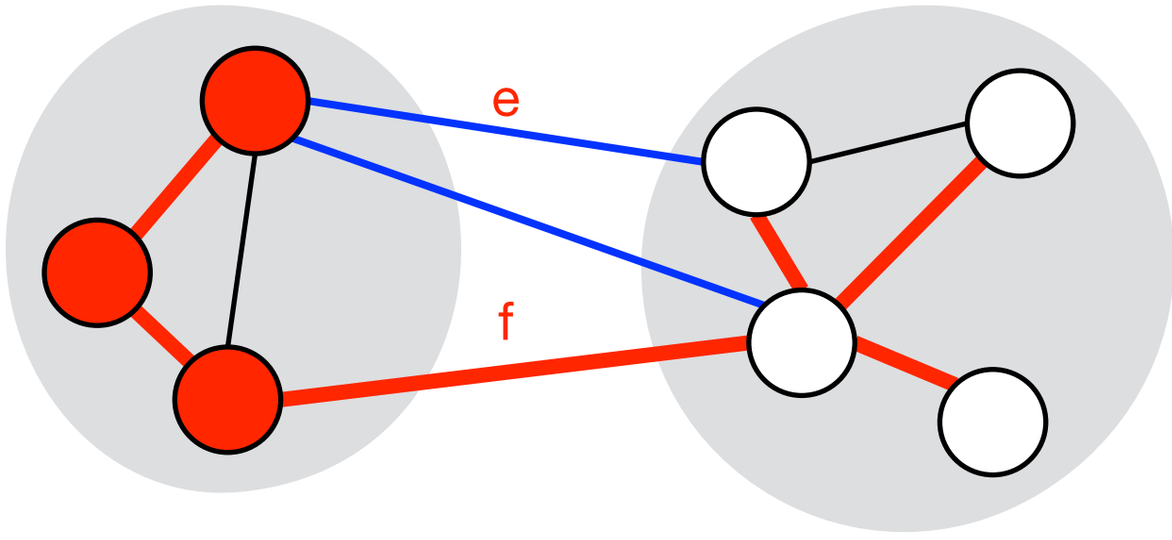
- Kreiseigenschaft. Für jeden Kreis ist die schwerste Kante nicht im MST.
- Beweis.
  - Angenommen die schwerste Kante f eines Kreises wäre im MST T.
  - Ersetze in T die Kante f durch eine leichtere Kante e im Kreis.
  - Der so erzeugte Teilgraph T’ ist ein Spannbaum mit kleinerem Gewicht. □

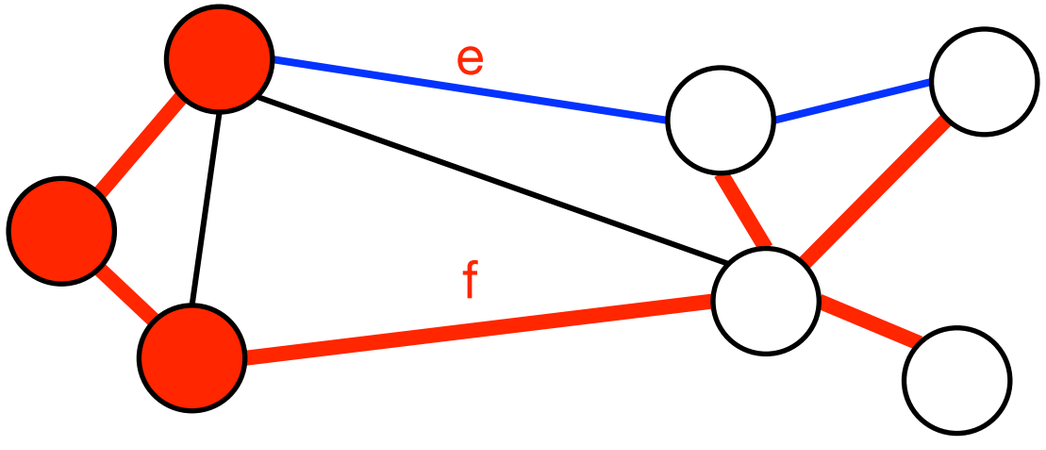




Eigenschaften Minimaler Spannbäume

- Schnitteigenschaft. Für jeden Schnitt ist die leichteste kreuzende Kante im MST.
- Kreiseigenschaft. Für jeden Kreis ist die schwerste Kante **nicht** im MST.



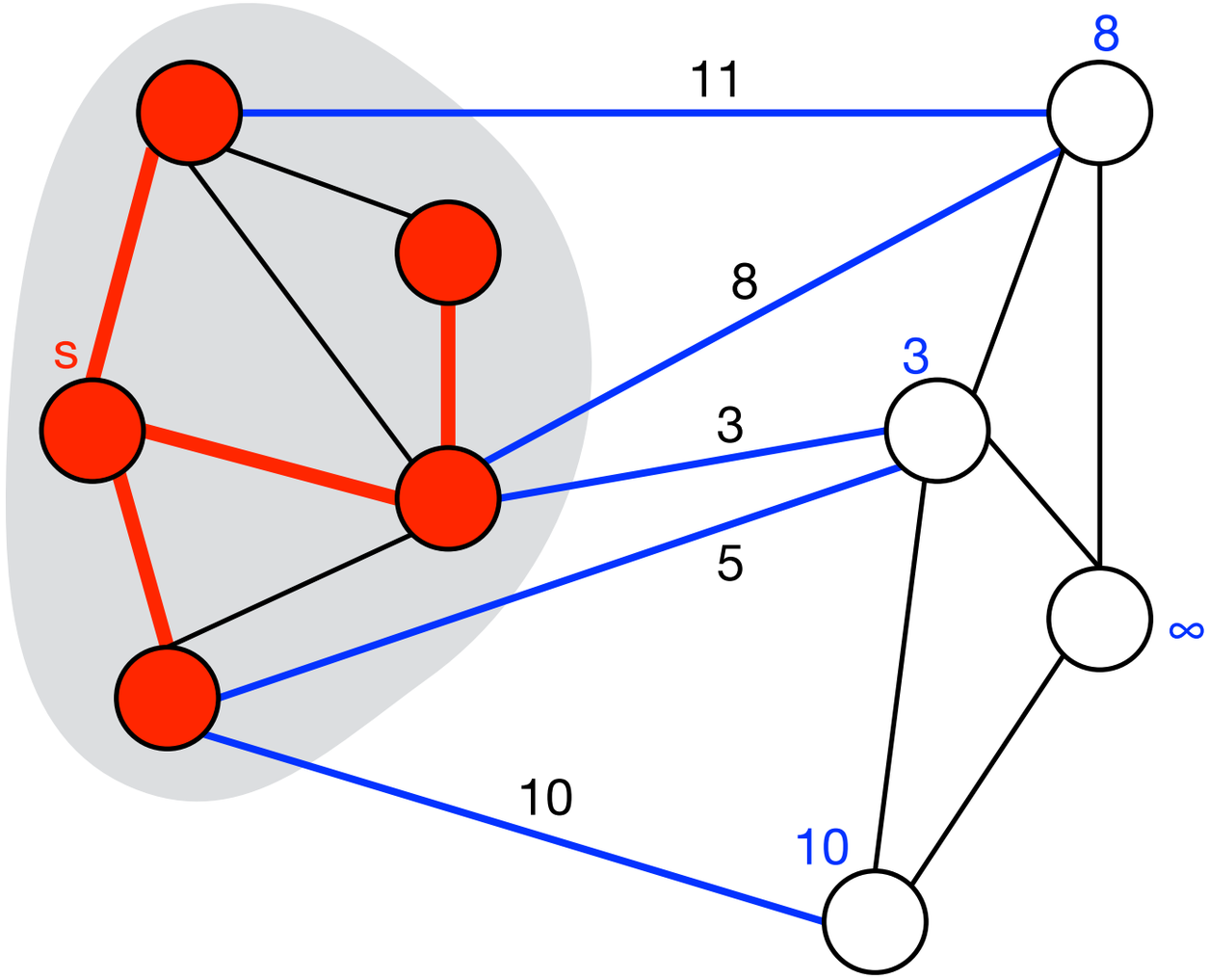


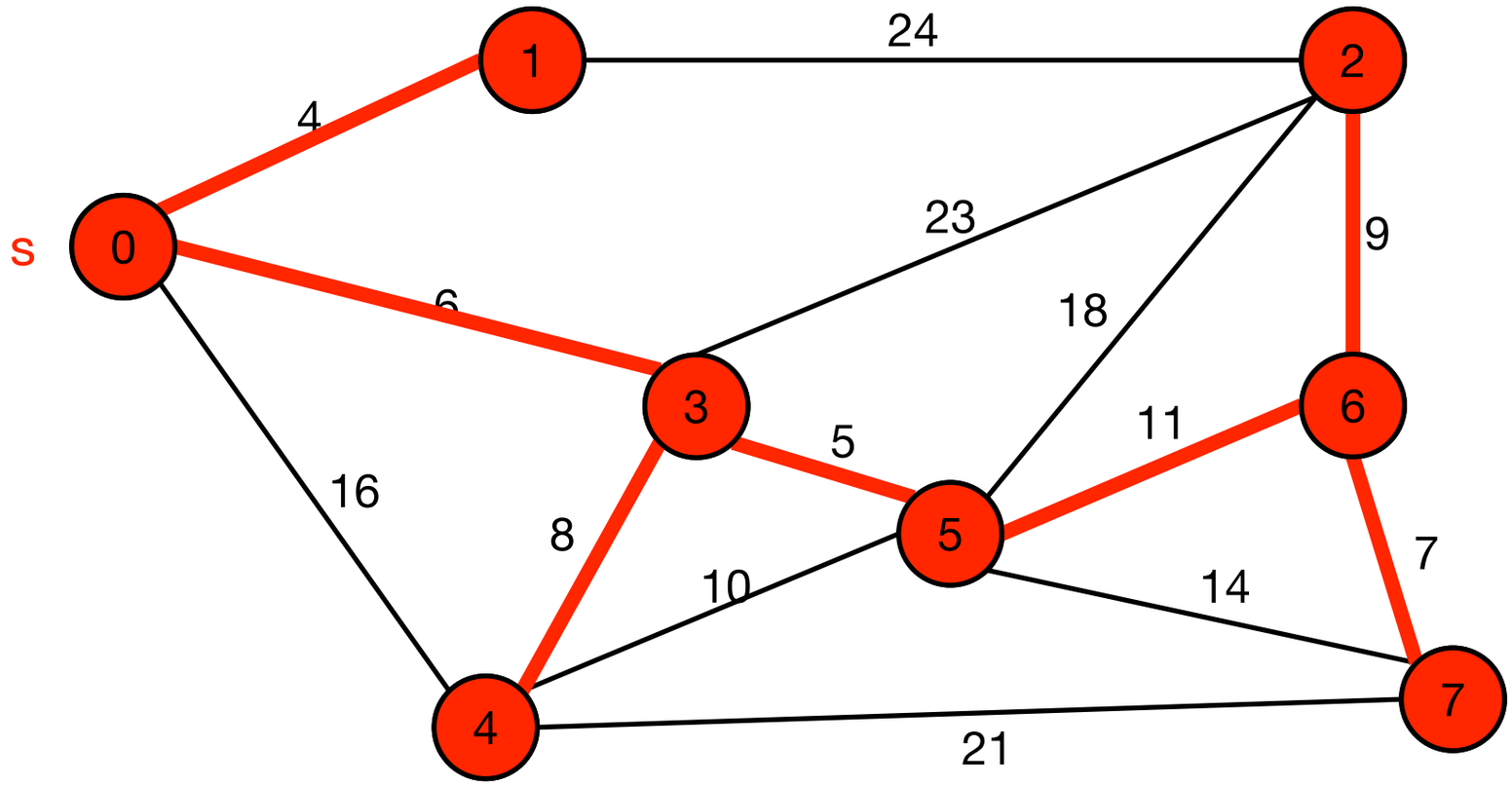
Minimale Spannbäume

- Minimale Spannbäume
- Darstellung von gewichteten Graphen
- Eigenschaften Minimaler Spannbäume
- Prims Algorithmus
- Kruskals Algorithmus

Prims Algorithmus

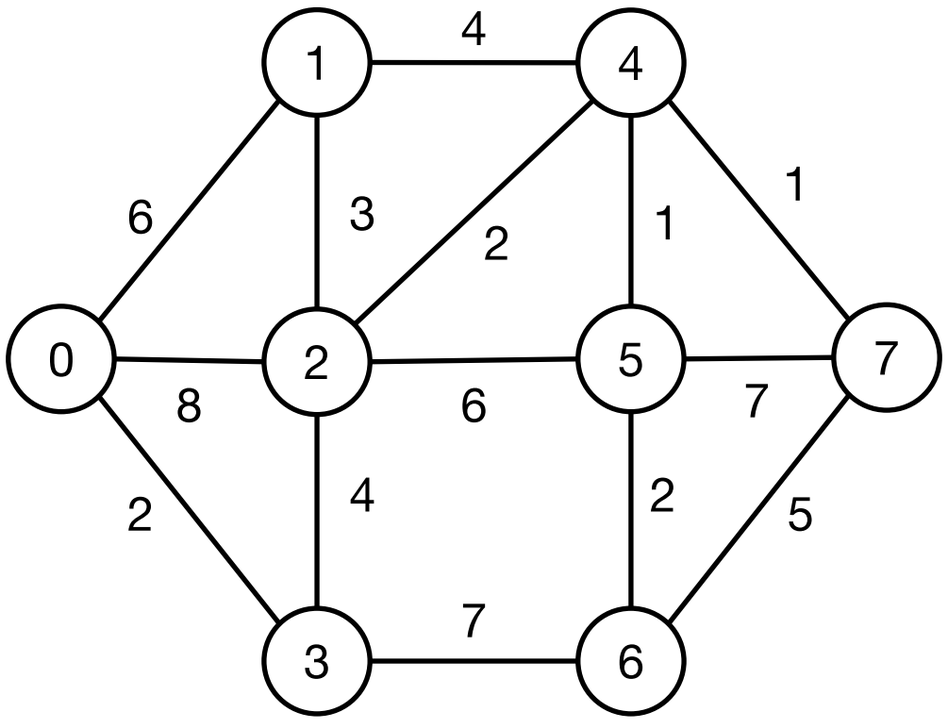
- Wachse einen Baum T von einem Knoten s.
- In jedem Schritt, füge **leichteste** Kante hinzu, die genau einen Endpunkt in T hat.
- Fertig, wenn n-1 Kanten hinzugefügt wurden.




s
0
1
2
3
4
5
6
7
24
4
23
9
6
18
16
11
5
8
10
14
7
21

Prims Algorithmus

- Wachse einen Baum T von einem Knoten s.
- In jedem Schritt, füge **leichteste** Kante hinzu, die genau einen Endpunkt in T hat.
- Fertig, wenn n-1 Kanten hinzugefügt wurden.
- Übung. Zeige die Ausführung von Prims Algorithmus auf dem folgenden Graph, wenn s=0 der Startknoten ist.

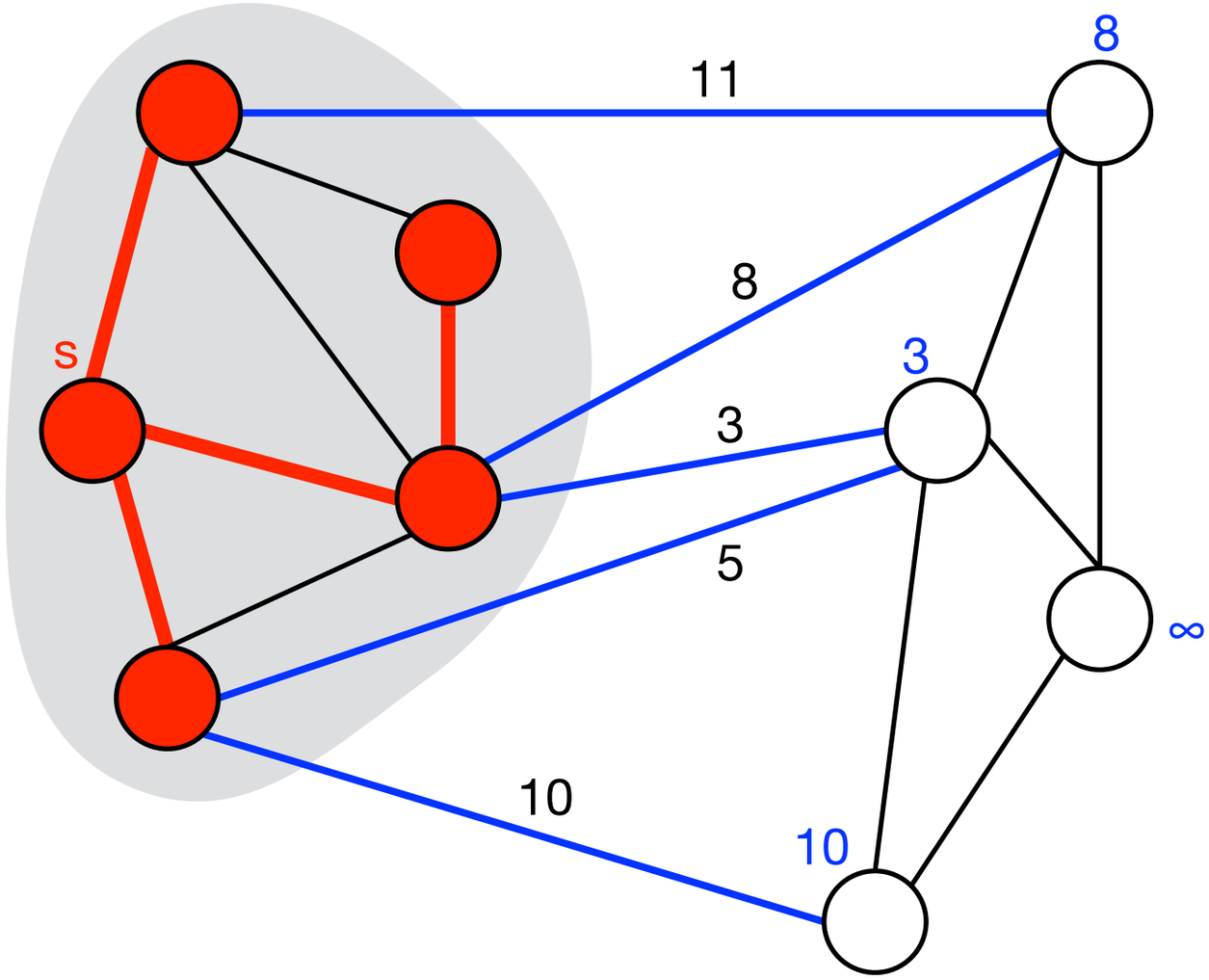


Prims Algorithmus

- Lemma. Prims Algorithmus berechnet den MST.
- Beweis.
  - Betrachte Schnitt zwischen T und den anderen Knoten.
  - Wir fügen die **leichteste** kreuzende Kante zu T hinzu.
  - Schnitteigenschaft $\Longrightarrow$ Kante ist im MST $\Longrightarrow$ T ist MST nach n-1 Schritten.

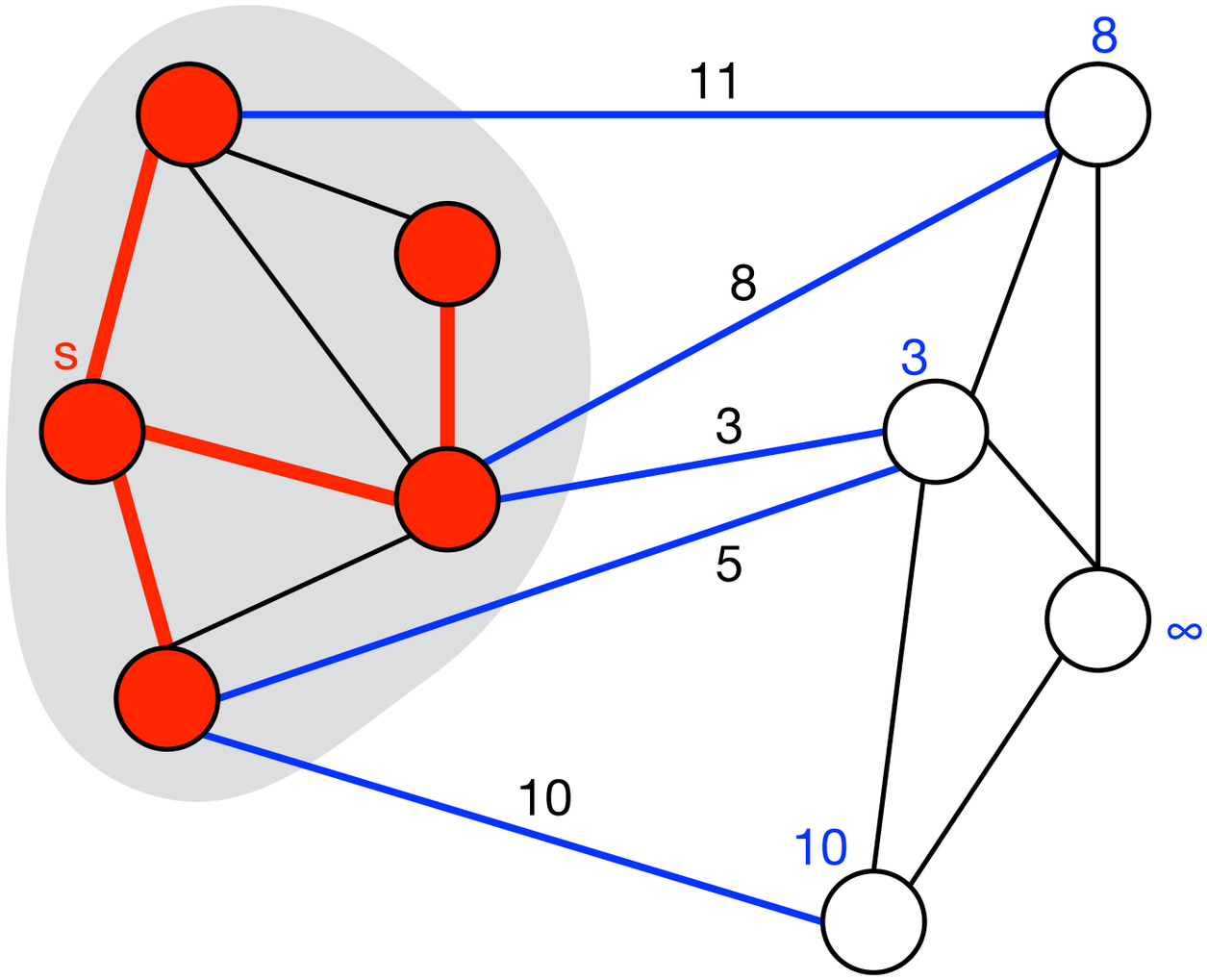
Prims Algorithmus

- Implementierung. Wie implementieren wir Prims Algorithmus?
- Schwierigkeit. Finde die leichteste kreuzende Kante.



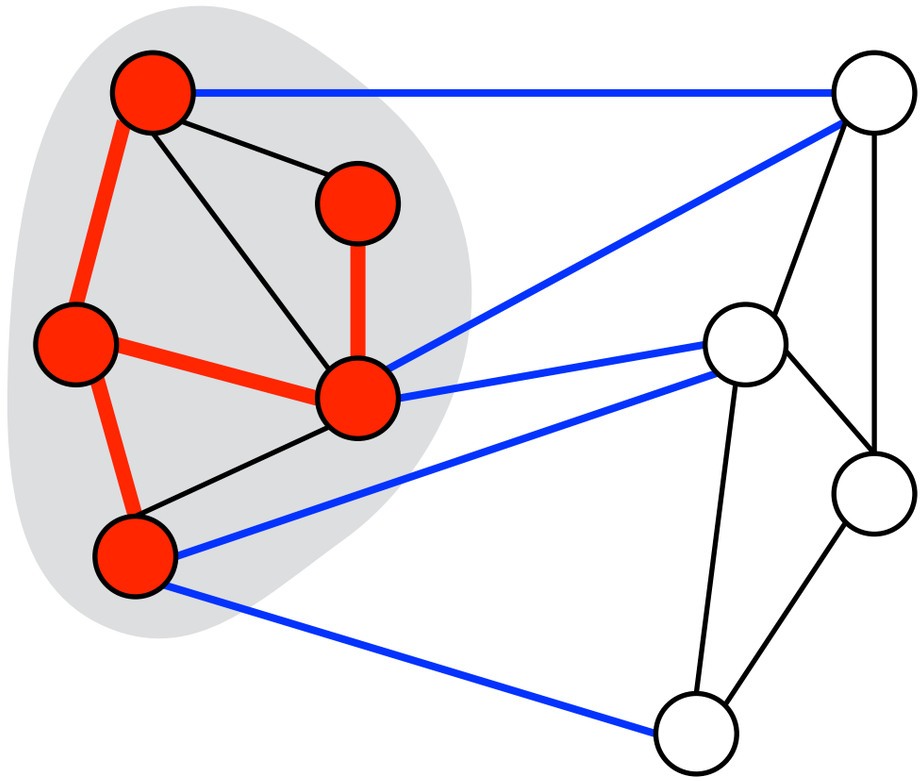
Prims Algorithmus

- Implementierung. Verwalte Knoten außerhalb von T in Prioritätswarteschlange.
  - Schlüssel von Knoten v = Gewicht der leichtesten kreuzenden Kante (oder ∞, falls keine kreuzende Kante existiert).
  - In jedem Schritt:
    - Finde leichteste Kante = EXTRACT-MIN
    - Aktualisiere Gewicht der Nachbarn des neuen Knoten mit DECREASE-KEY.



Prims Algorithmus

```
PRIM(G, s)
   für alle Knoten v∈V
      v.key = ∞
      v.π = null
      INSERT(P,v)
   DECREASE-KEY(P,s,0)
   while (P ≠ ∅)
      u = EXTRACT-MIN(P)
      für alle Nachbarn v von u
         if v ∈ P and w(u,v) < v.key
            DECREASE-KEY(P,v,w(u,v))
            v.π = u
```



- Zeit.
  - n EXTRACT-MIN
  - n INSERT
  - O(m) DECREASE-KEY
- Gesamtzeit mit Min-Heap. O(n log n + n log n + m log n) = O(m log n)

Prims Algorithmus

- Prioritätswarteschlange und Prims Algorithmus. Komplexität von Prims Algorithmus hängt von der Implementierung der Prioritätswarteschlange ab.
  - n INSERT
  - n EXTRACT-MIN
  - O(m) DECREASE-KEY

Prioritäts-warteschlange	INSERT	EXTRACT-MIN	DECREASE-KEY	Total
Feld	O(1)	O(n)	O(1)	$O(n^2)$
Binärer Heap	O(log n)	O(log n)	O(log n)	O(m log n)
Fibonacci Heap	$O(1)^\dagger$	$O(\log n)^\dagger$	$O(1)^\dagger$	O(m + n log n)

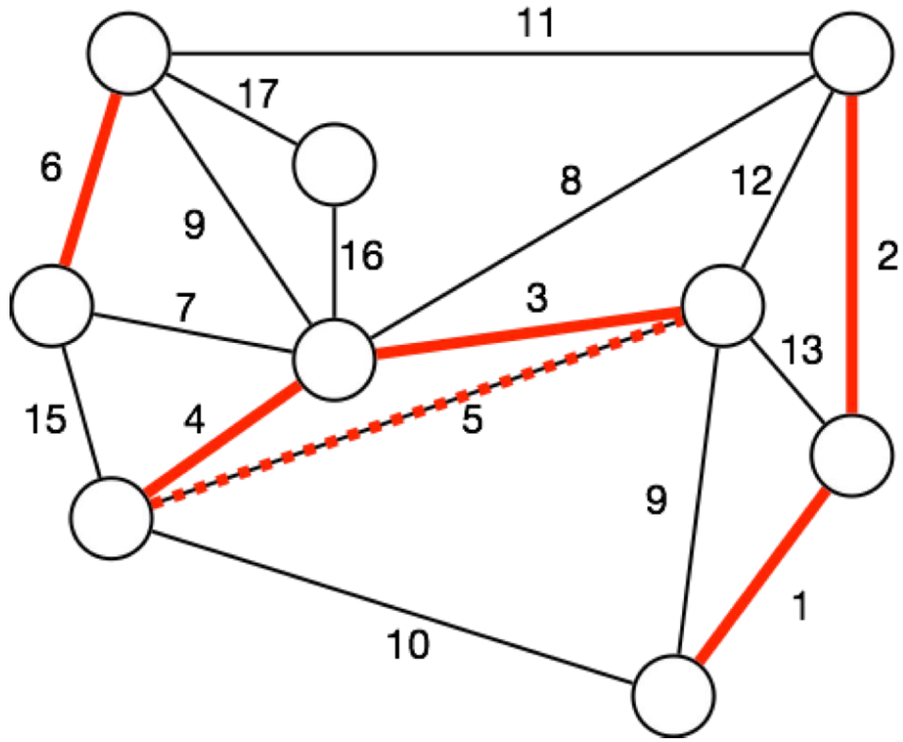
- Gier. Prims Algorithmus ist ein gieriger Algorithmus (*greedy algorithm*).
  - Trifft in jedem Schritt lokal optimale Entscheidungen, die zu einer global optimalen Lösung führen.

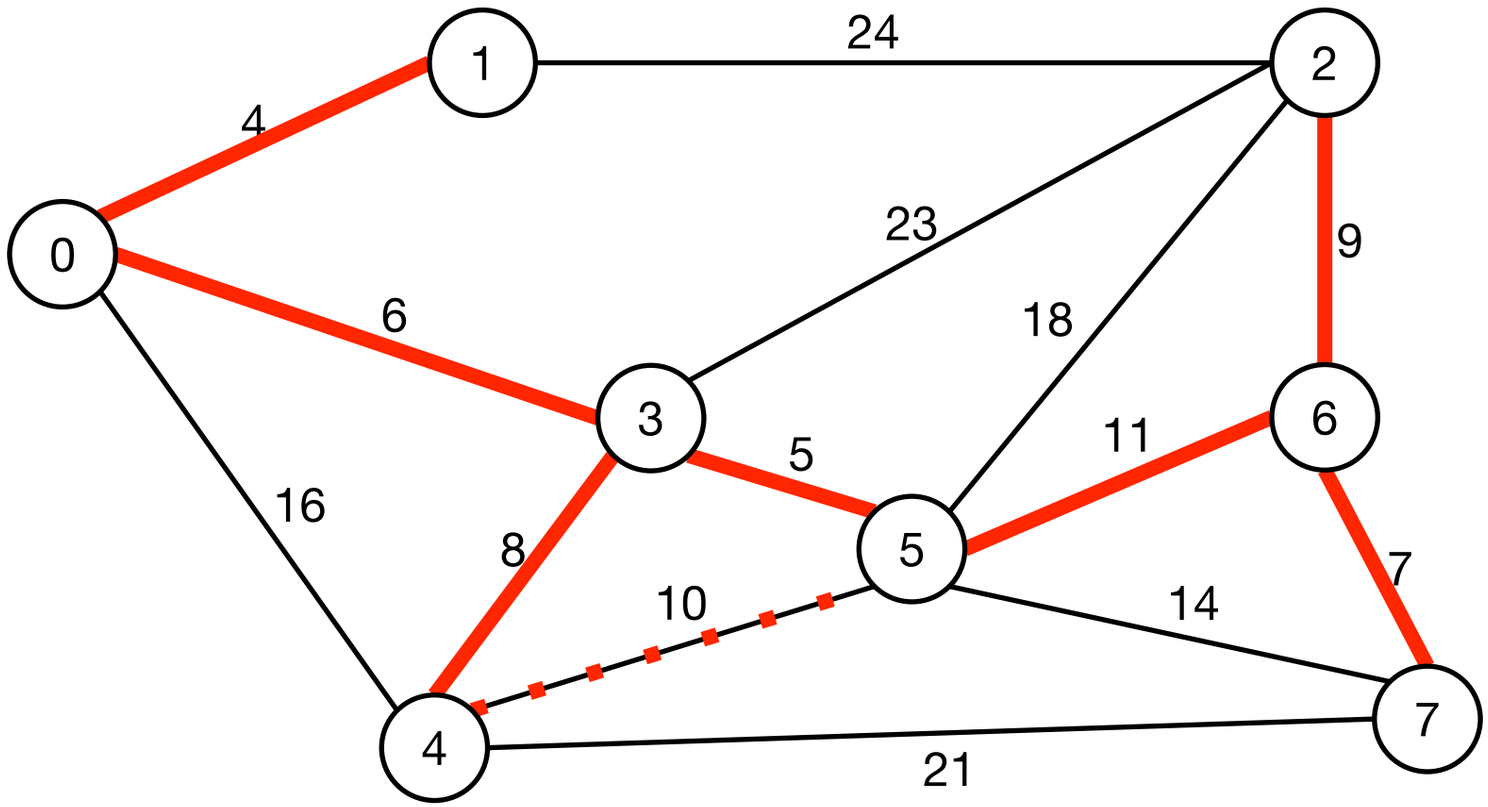
Minimale Spannbäume

- Minimale Spannbäume
- Darstellung von gewichteten Graphen
- Eigenschaften Minimaler Spannbäume
- Prims Algorithmus
- Kruskals Algorithmus

Kruskals Algorithmus

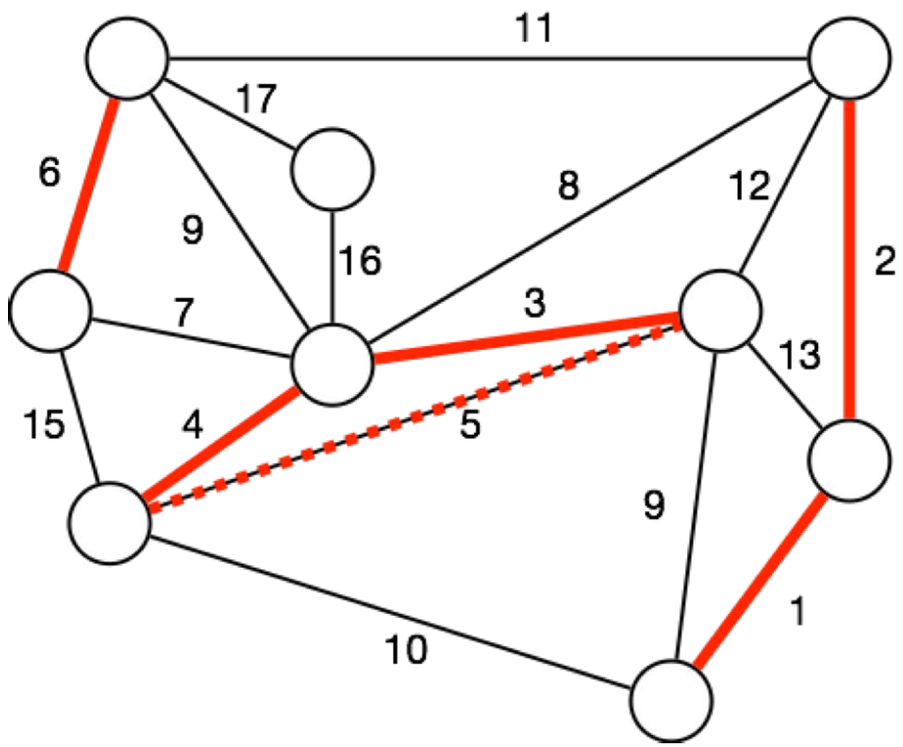
- Durchlaufe Kanten von der leichtesten bis zu schwersten.
- In jedem Schritt, füge Kante e zu T hinzu, wenn e keinen Kreis erzeugt.
- Fertig, wenn n-1 Kanten hinzugefügt wurden.




0
1
2
3
4
5
6
7
4
24
23
9
6
18
16
5
11
8
10
14
7
21

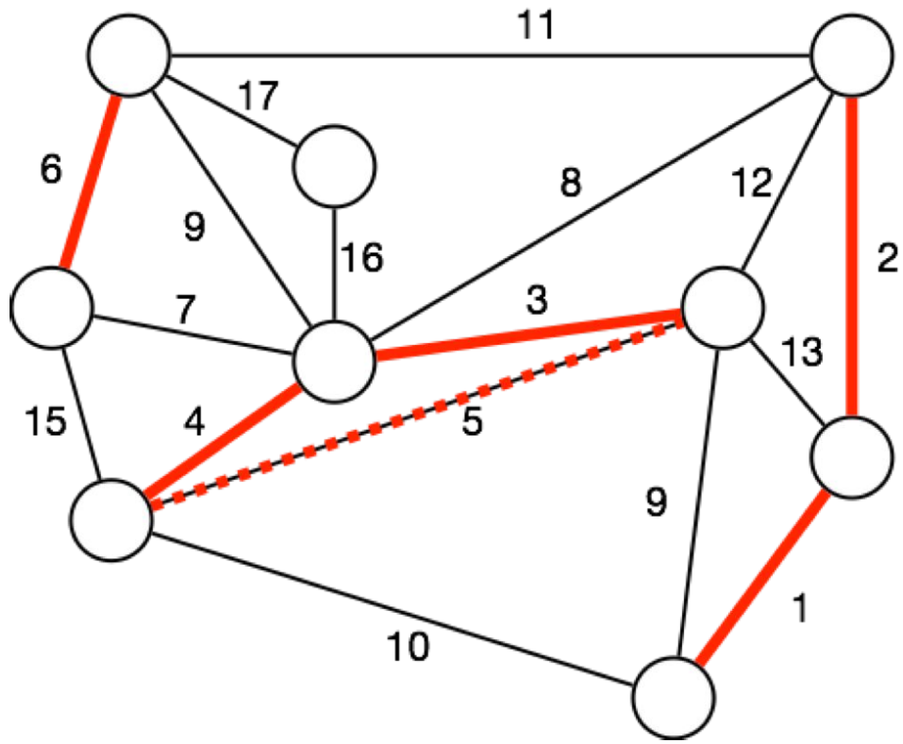
Kruskals Algorithmus

- Lemma. Kruskals Algorithmus berechnet den MST.
- Beweis.
  - Algorithms betrachtet Kanten von leicht zu schwer. Bei Kante e = (u,v):
  - Fall 1. e erzeugt einen Kreis und wird nicht zu T hinzugefügt.
    - e muss die schwerste Kante auf dem Kreis sein.
    - Kreiseigenschaft $\Longrightarrow$ e ist nicht Teils des MST.
  - Fall 2. e erzeugt keinen Kreis und wird zu T hinzugefügt.
    - e muss die leichteste kreuzende Kante des Schnitts sein.
    - Schnitteigenschaft $\Longrightarrow$ e ist in MST.
- $\Longrightarrow$ T=MST sobald n-1 Kanten hinzugefügt.



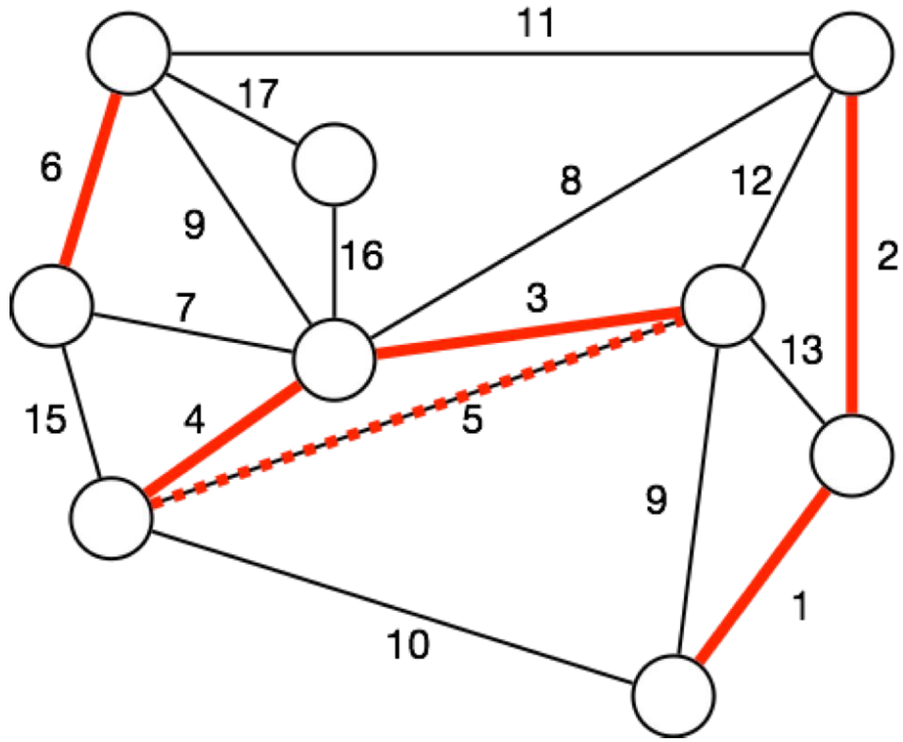
Kruskals Algorithmus

- Implementierung. Wie können wir Kruskals Algorithmus implementieren?
- Herausforderung. Testen, ob eine Kante e einen Kreis erzeugt.



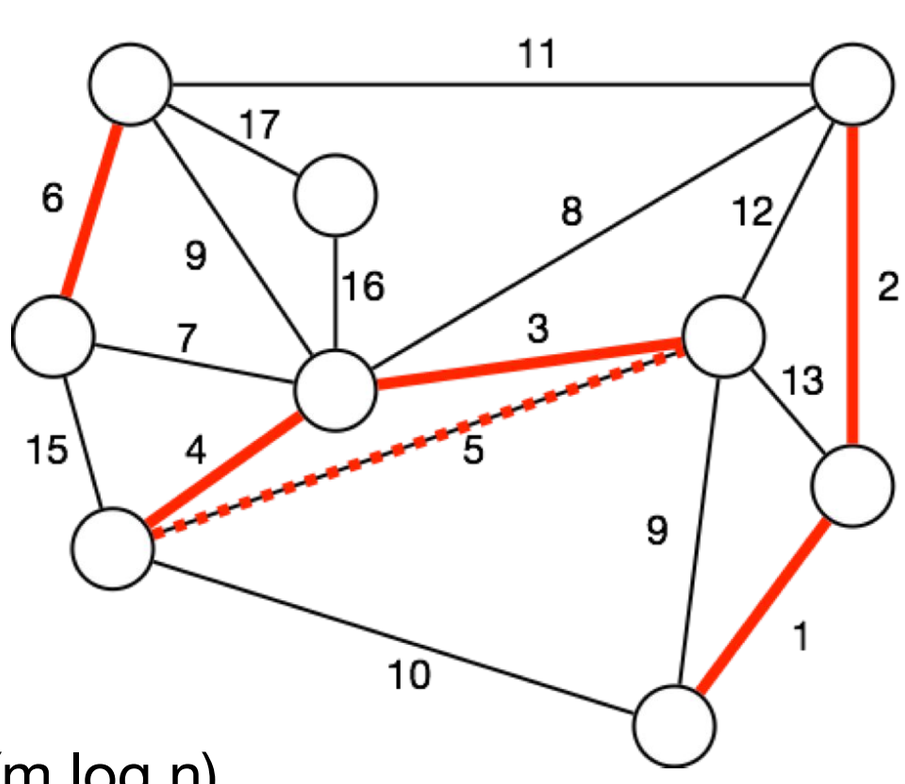
Kruskals Algorithmus

- Implementierung.
  Verwalte Kanten in Datenstruktur für dynamischen Zusammenhang.
- In jedem Schritt:
  - Teste, ob eine Kante einen Kreis erzeugt (= ZUSAMMENHÄNGEND).
  - Füge eine neue Kante ein (= INSERT).



Kruskals Algorithmus

```
KRUSKAL(G)
   Sortiere die Kanten
   INIT(n)
   für alle Kanten (u,v) in sortierter Reihenfolge
      if (!ZUSAMMENHÄNGEND(u,v))
         INSERT(u,v)
   return alle eingefügten Kanten
```



- Zeit.
  - Sortieren von m Kanten.
  - 1 INIT
  - m ZUSAMMENHÄNGEND
  - n INSERT
- Gesamtzeit. O(m log m + n + m log n + n log n) = O(m log n).
- Gier. Kruskals Algorithmus ist ebenfalls ein gieriger Algorithmus.

Minimale Spannbäume

- Was ist der beste Algorithmus, um MSTs zu finden?

Jahr	Zeit	Autoren
???	O(m log n)	Borůvka, Jarník, Prim, Dijkstra, Kruskal, ?
1975	O(m log log n)	Yao
1986	O(m log* n)	Fredman, Tarjan
1995	O(m)‡	Karger, Klein, Tarjan
2000	O(nα(m,n))	Chazelle
2002	optimal	Pettie, Ramachandran

‡ = randomisiert

Eselsbrücke.
Prim=Primzahl=unteilbar=“wachse einen (zusammenhängenden) Baum”
Kruskal=“wachse Wald”

Minimale Spannbäume

- Minimale Spannbäume
- Darstellung von gewichteten Graphen
- Eigenschaften Minimaler Spannbäume
- Prims Algorithmus
- Kruskals Algorithmus