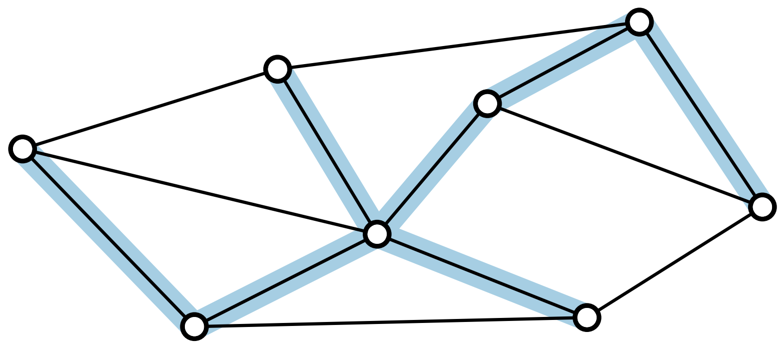
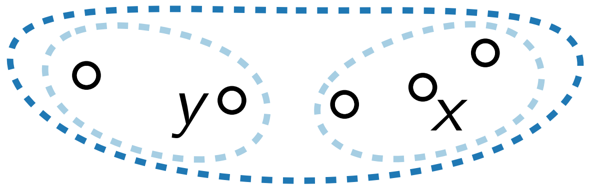
Algorithmen und Datenstrukturen

Vorlesung 21:
Minimale Spannbäume



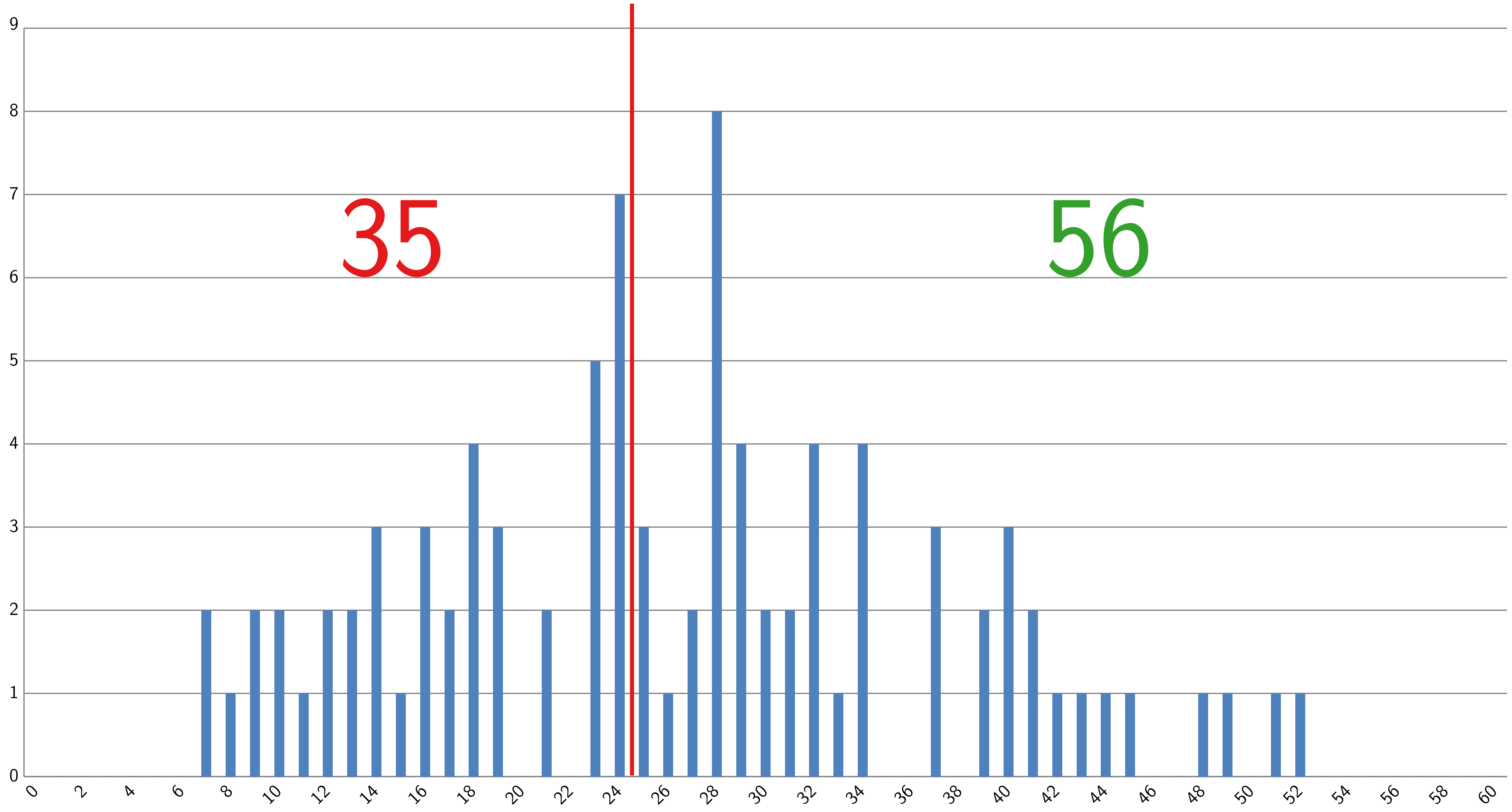


Alexander Wolff

Wintersemester 2025

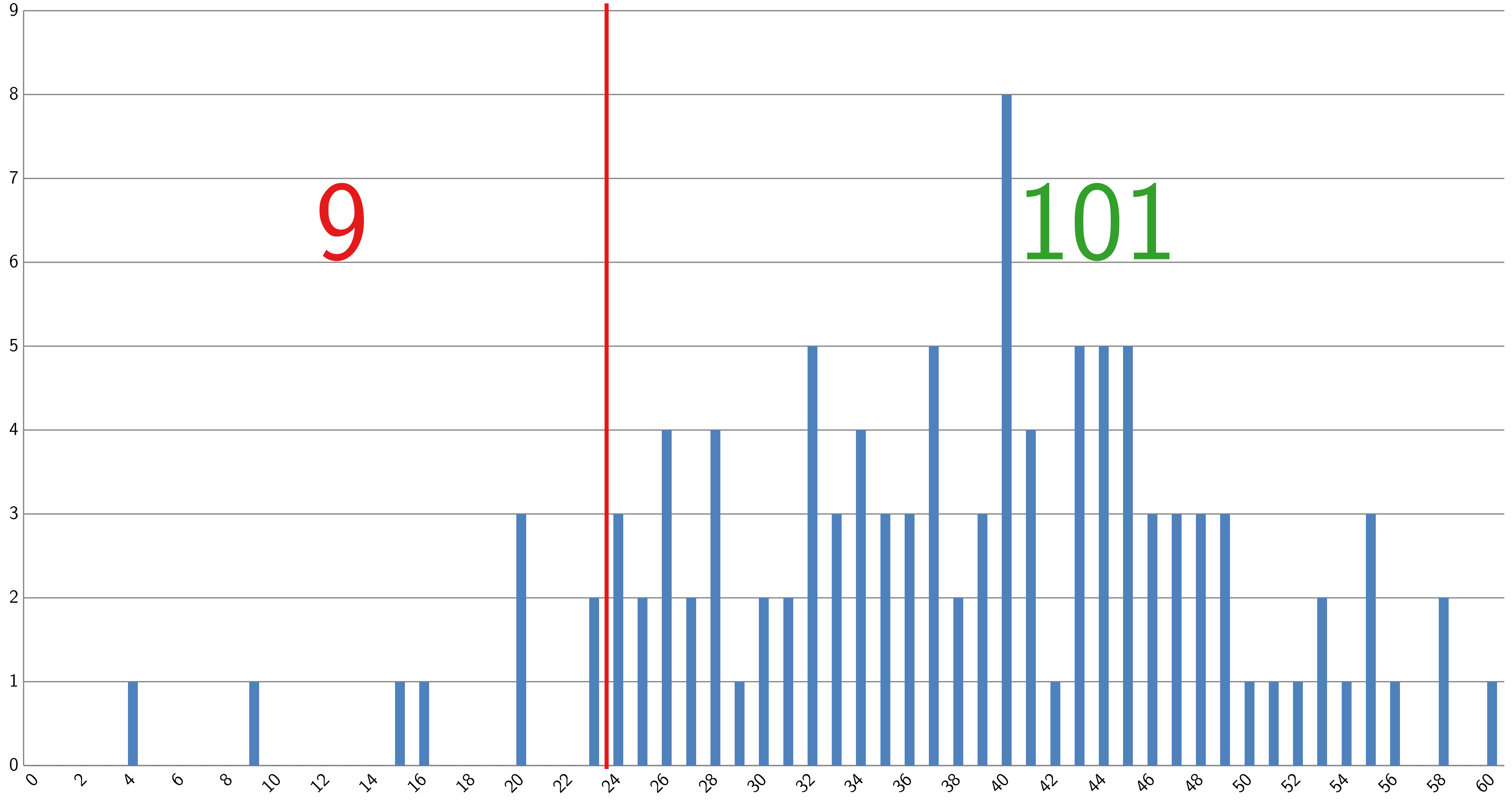
3. Zwischentest

$n = 91$; $\mu = 44{,}2$; Median $= 26$

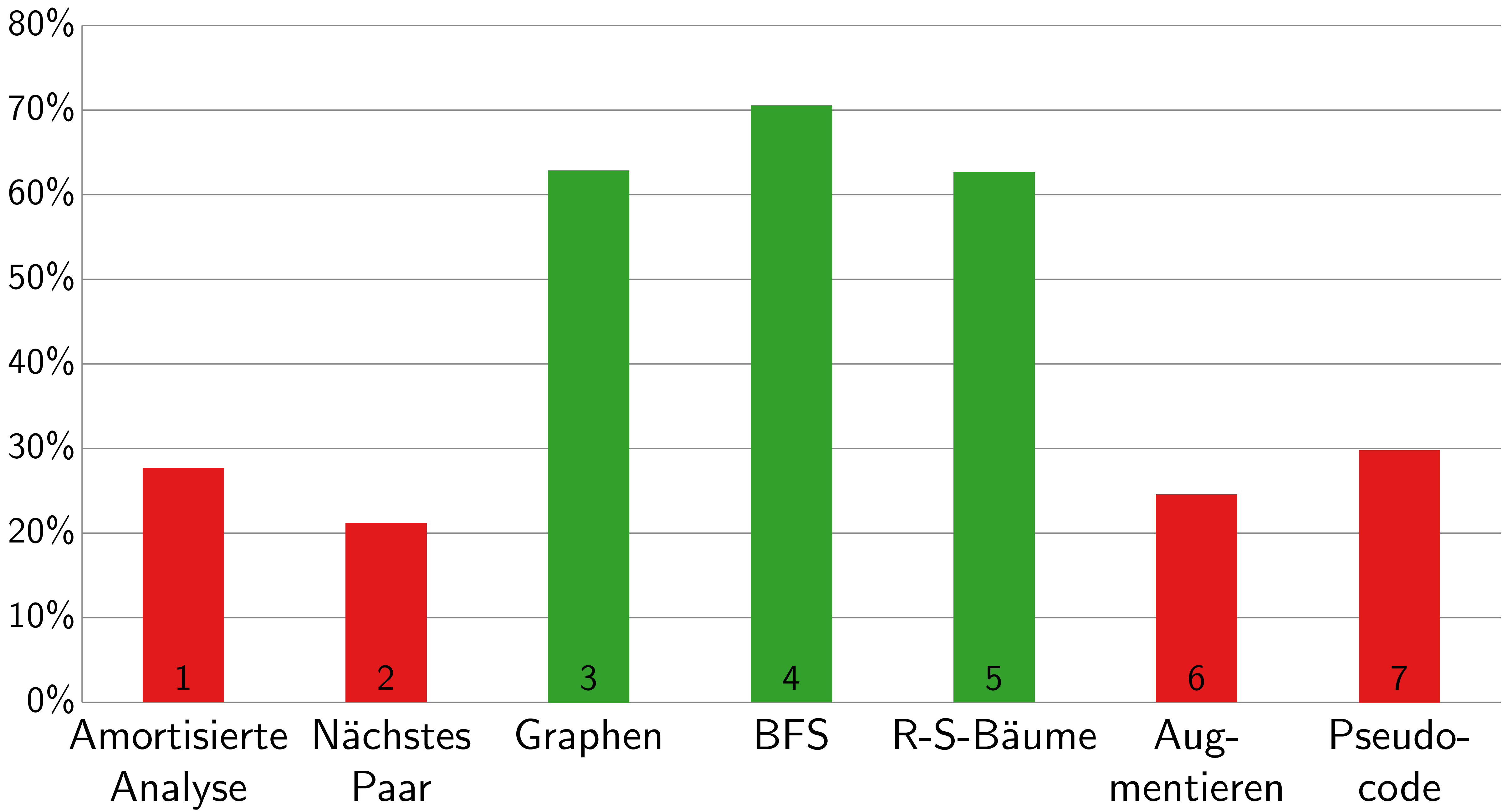


3. Zwischentest im WS 2024/25

$n = 110$; $\mu = 37{,}8$; Median $= 38$



Aufgaben


80%
70%
60%
50%
40%
30%
20%
10%
0%
1
2
3
4
5
6
7
Amortisierte Analyse
Nächstes Paar
Graphen
BFS
R-S-Bäume
Aug-mentieren
Pseudo-code

Motivation

Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

Motivation

ungerichteter, gewichteter Graph

Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

Motivation

ungerichteter, gewichteter Graph

Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

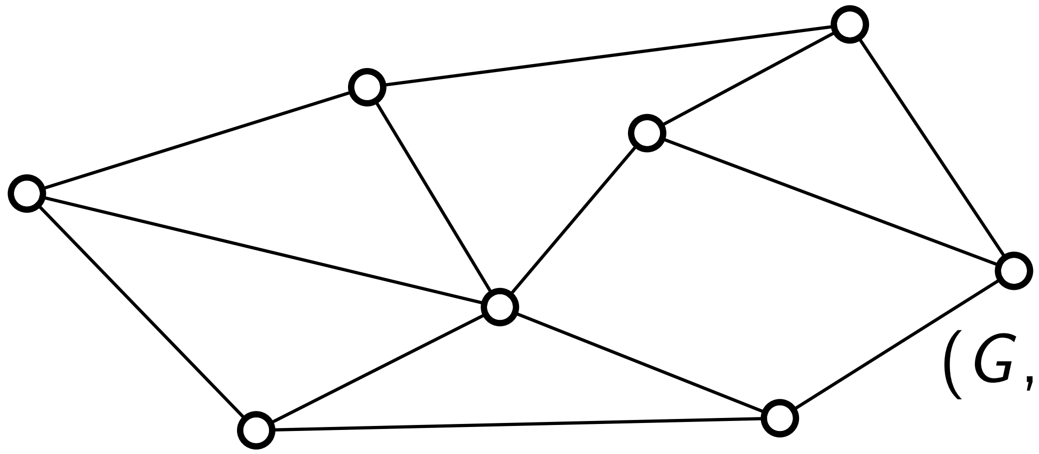
$w\colon E(G) \to \mathbb{R}_{>0}$ Kantengewichte

Motivation

ungerichteter, gewichteter Graph

Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

$w\colon E(G) \to \mathbb{R}_{>0}$ Kantengewichte



(G, w)

z.B. mit $w \equiv$ euklid. Abstände

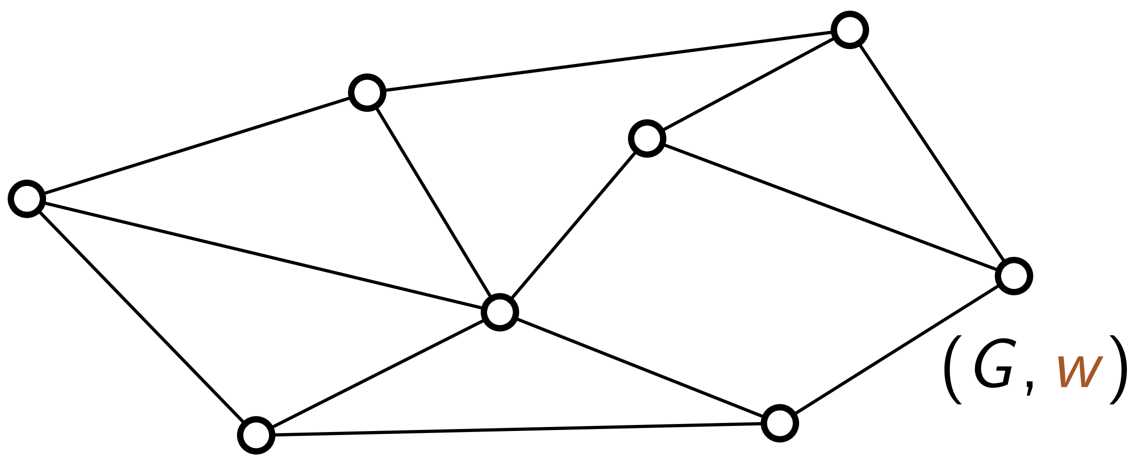
Motivation

ungerichteter, gewichteter Graph

Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

$w \colon E(G) \to \mathbb{R}_{>0}$ Kantengewichte

Gesucht. Teilnetz $T = (V(G), E')$ mit $E' \subseteq E(G)$, so dass



z.B. mit $w \equiv$ euklid. Abstände

Motivation

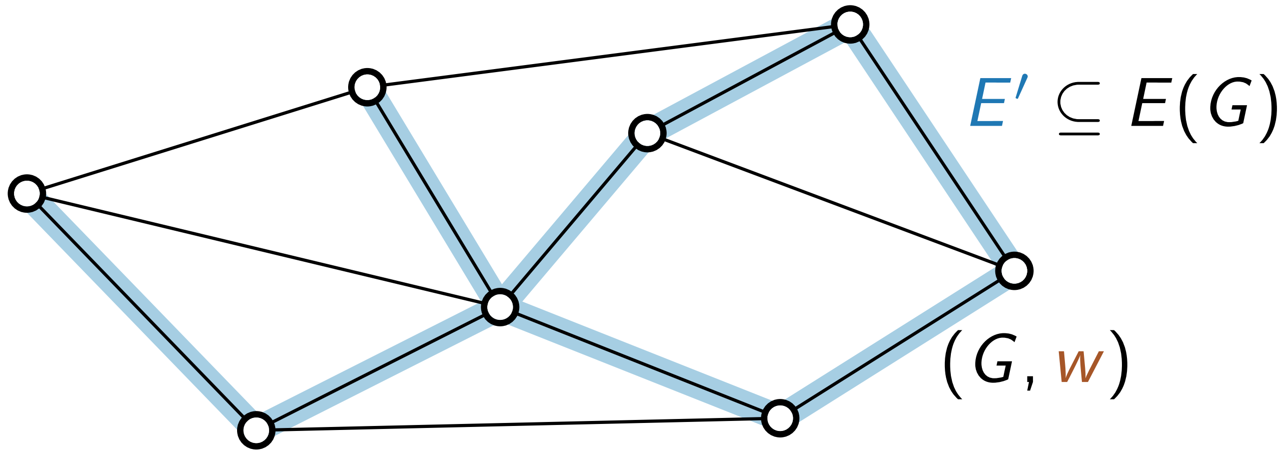
ungerichteter, gewichteter Graph

Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

$w\colon E(G) \to \mathbb{R}_{>0}$ Kantengewichte

Gesucht. Teilnetz $T = (V(G), E')$ mit $E' \subseteq E(G)$, so dass

- alle Städte in T erreichbar sind



z.B. mit $w \equiv$ euklid. Abstände

Motivation

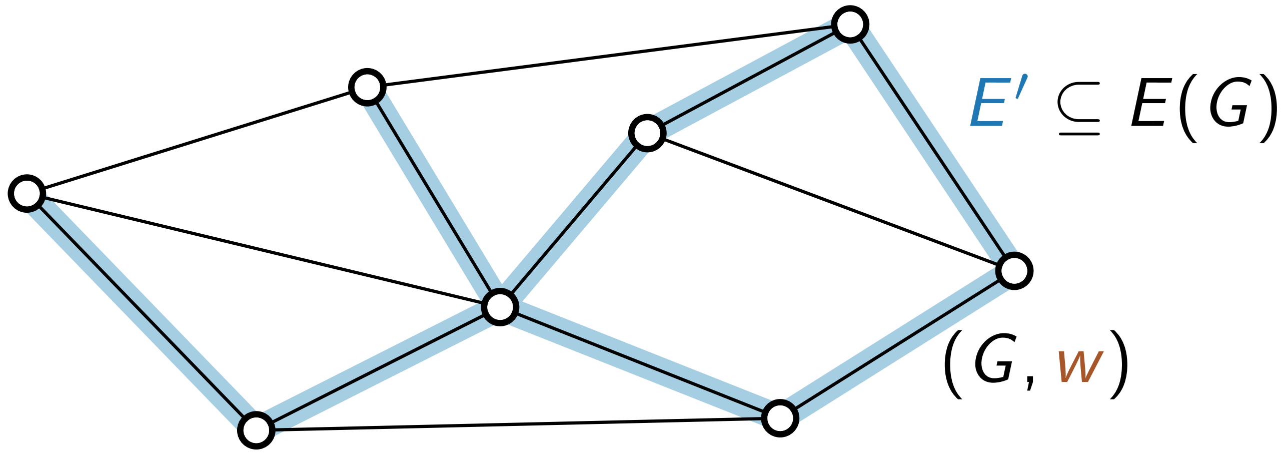
ungerichteter, gewichteter Graph

Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

$w\colon E(G) \to \mathbb{R}_{>0}$ Kantengewichte

Gesucht. Teilnetz $T = (V(G), E')$ mit $E' \subseteq E(G)$, so dass

- alle Städte in T erreichbar sind (T **spannt** G **auf**)



z.B. mit $w \equiv$ euklid. Abstände

Motivation

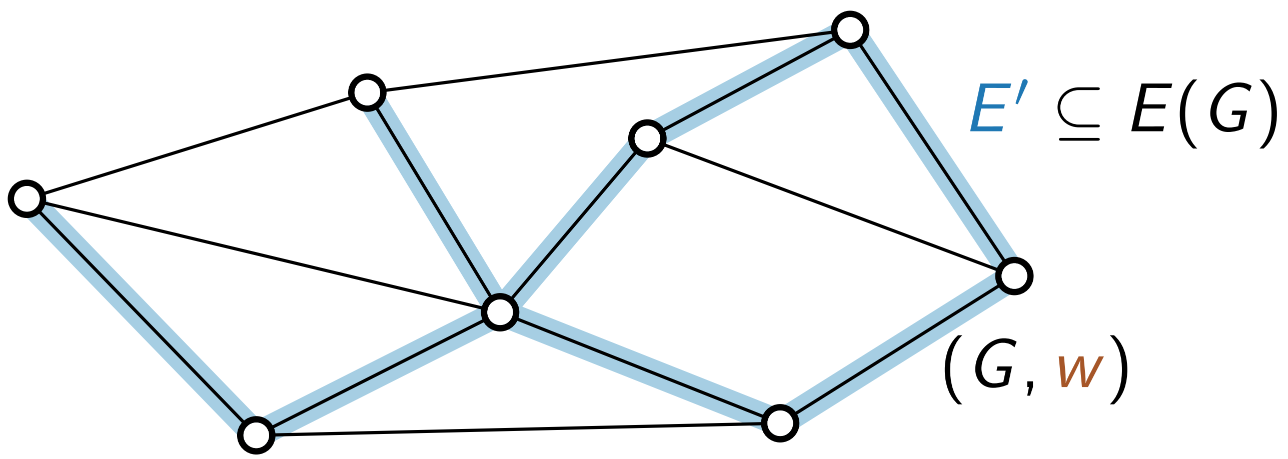
Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

ungerichteter, gewichteter Graph

$w \colon E(G) \to \mathbb{R}_{>0}$ Kantengewichte

Gesucht. Teilnetz $T = (V(G), E')$ mit $E' \subseteq E(G)$, so dass

- alle Städte in T erreichbar sind (T **spannt** G **auf**)
- die „Schneeräumkosten“ $w(E')$ minimal sind unter allen Teilnetzen, die G aufspannen.



z.B. mit $w \equiv$ euklid. Abstände

Motivation

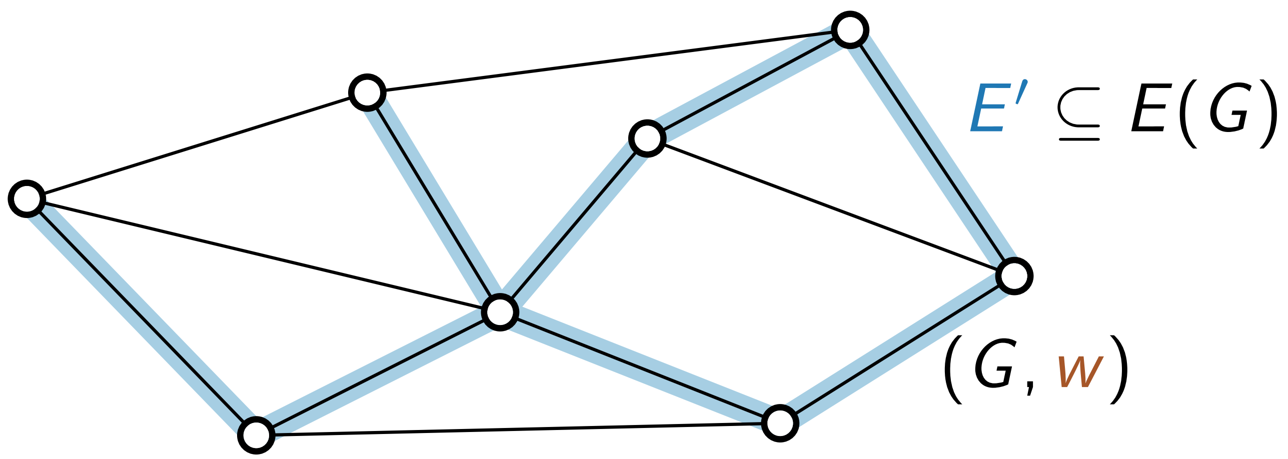
ungerichteter, gewichteter Graph

Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

$w\colon E(G) \to \mathbb{R}_{>0}$ Kantengewichte
$w(E') := \sum_{e \in E'} w(e)$

Gesucht. Teilnetz $T = (V(G), E')$ mit $E' \subseteq E(G)$, so dass

- alle Städte in T erreichbar sind (T **spannt** G **auf**)
- die „Schneeräumkosten" $w(E')$ minimal sind unter allen Teilnetzen, die G aufspannen.



z.B. mit $w \equiv$ euklid. Abstände

Motivation

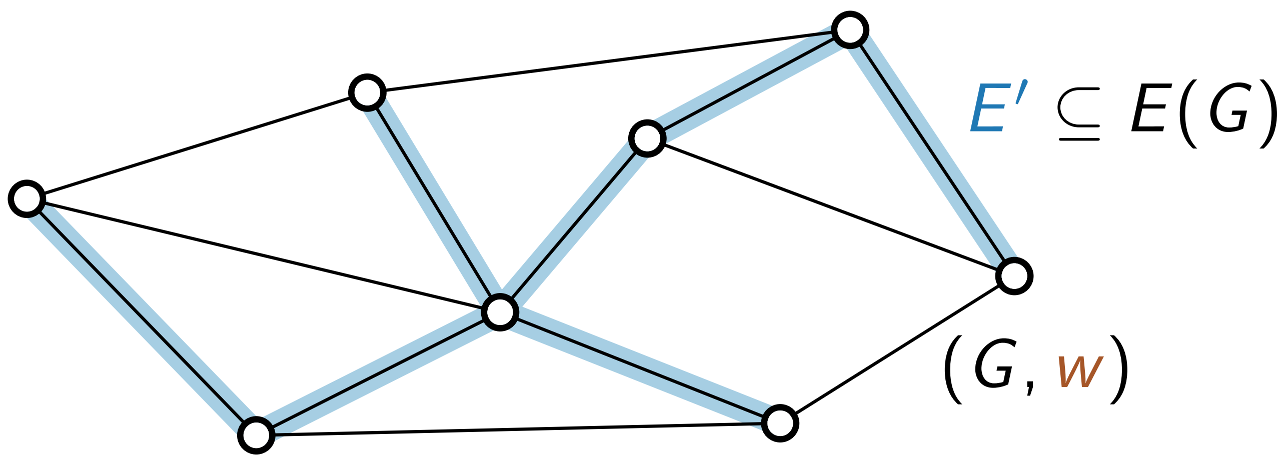
Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

ungerichteter, gewichteter Graph

$w\colon E(G) \to \mathbb{R}_{>0}$ Kantengewichte
$w(E') := \sum_{e \in E'} w(e)$

Gesucht. Teilnetz $T = (V(G), E')$ mit $E' \subseteq E(G)$, so dass

- alle Städte in T erreichbar sind (T **spannt** G **auf**)
- die „Schneeräumkosten" $w(E')$ minimal sind unter allen Teilnetzen, die G aufspannen.



z.B. mit $w \equiv$ euklid. Abstände

Motivation

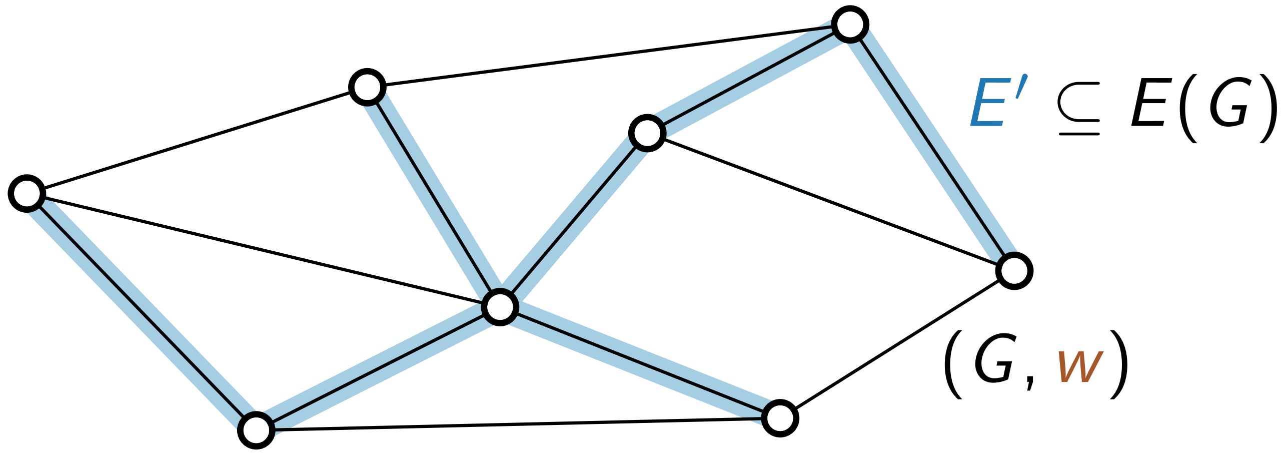
ungerichteter, gewichteter Graph

Gegeben. Zusammenhängendes Straßennetz (G, w), das eine Menge V von n Städten verbindet

$w\colon E(G) \to \mathbb{R}_{>0}$ Kantengewichte
$w(E') := \sum_{e \in E'} w(e)$

Gesucht. Teilnetz $T = (V(G), E')$ mit $E' \subseteq E(G)$, so dass

- alle Städte in T erreichbar sind (T **spannt** G **auf**)
- die „Schneeräumkosten“ $w(E')$ minimal sind unter allen Teilnetzen, die G aufspannen.

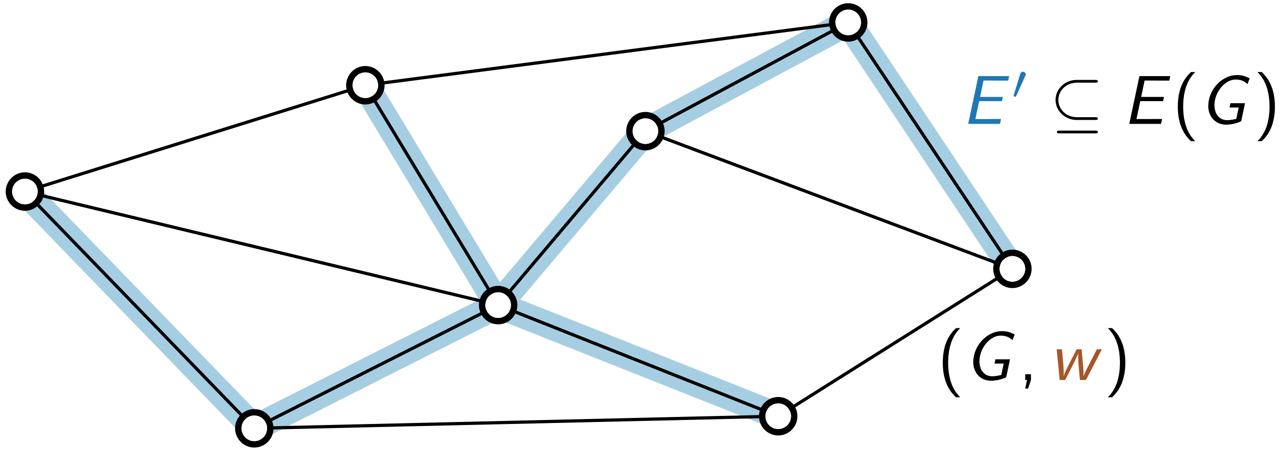


z.B. mit $w \equiv$ euklid. Abstände

Minimaler Spannbaum

Wegen der Minimalität von $w(E')$ gilt:

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn

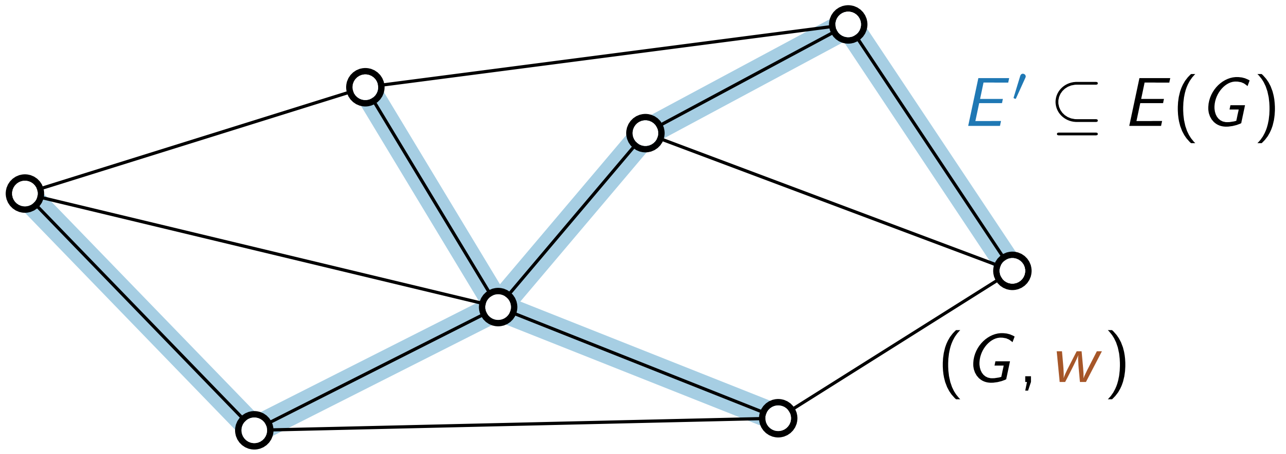


Minimaler Spannbaum

Wegen der Minimalität von $w(E')$ gilt:

T hat keine Kreise

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn

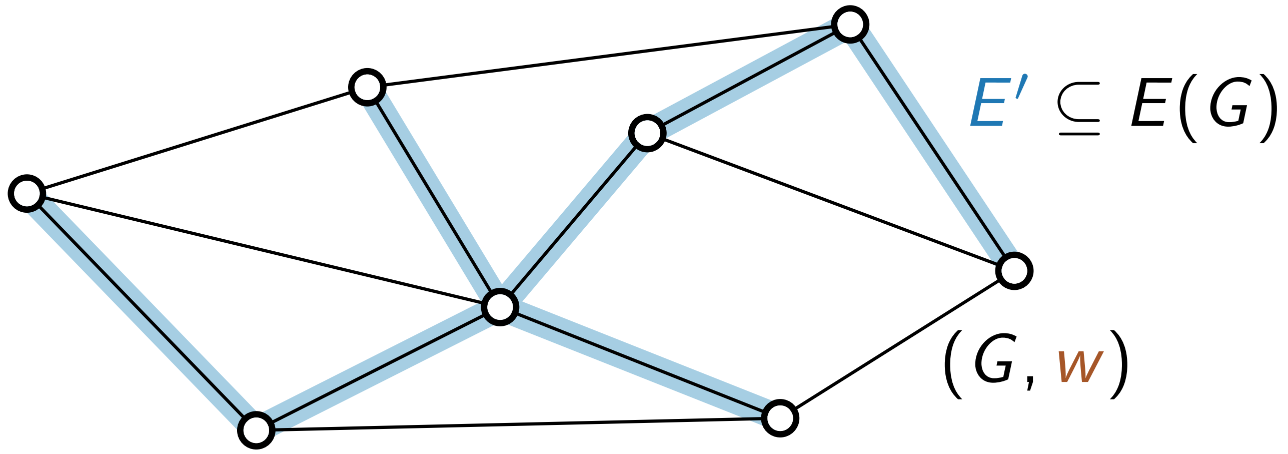


Minimaler Spannbaum

Wegen der Minimalität von $w(E')$ gilt:

T hat keine Kreise $\quad\Rightarrow\quad$ T ist ein Wald

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn



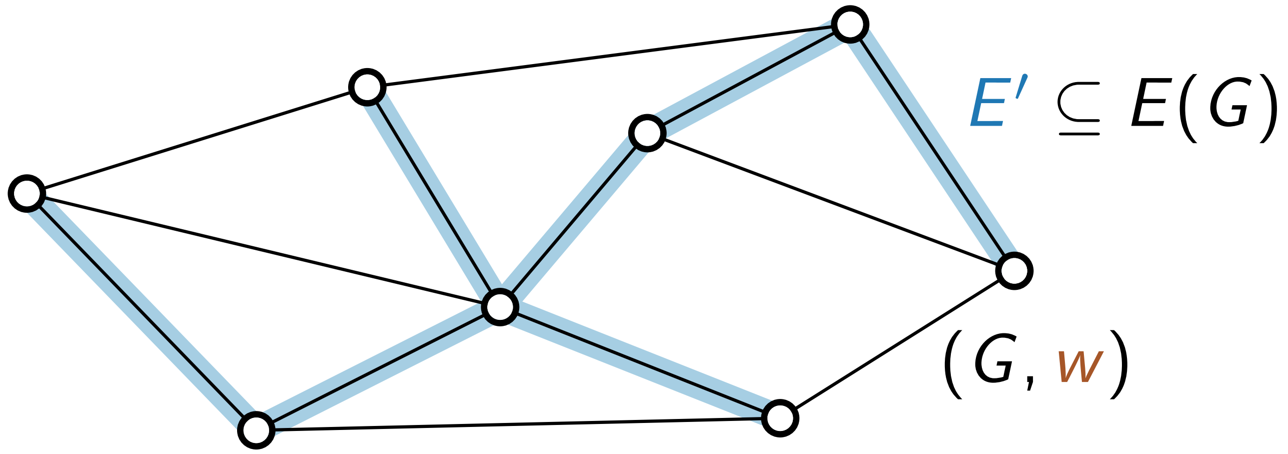
Minimaler Spannbaum

Wegen der Minimalität von $w(E')$ gilt:

T hat keine Kreise $\Rightarrow$ T ist ein Wald

T „erbt“ Zusammenhang von G

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn

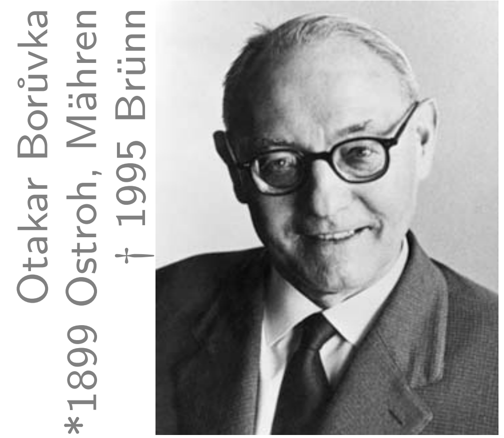


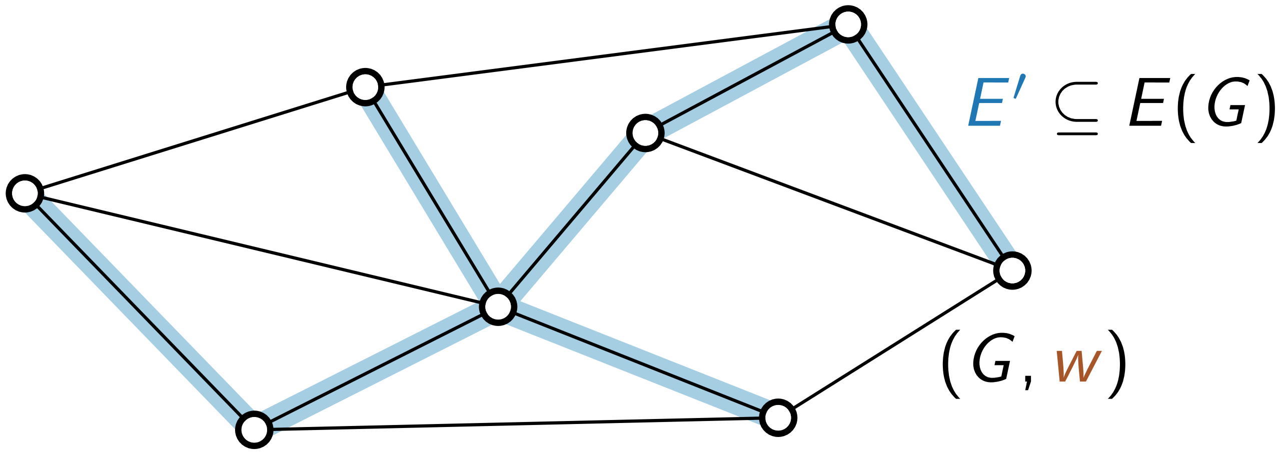
Minimaler Spannbaum

Wegen der Minimalität von $w(E')$ gilt:

T hat keine Kreise $\Rightarrow$ T ist ein Wald

T „erbt" Zusammenhang von G $\Rightarrow$ T ist ein Baum





Minimaler Spannbaum

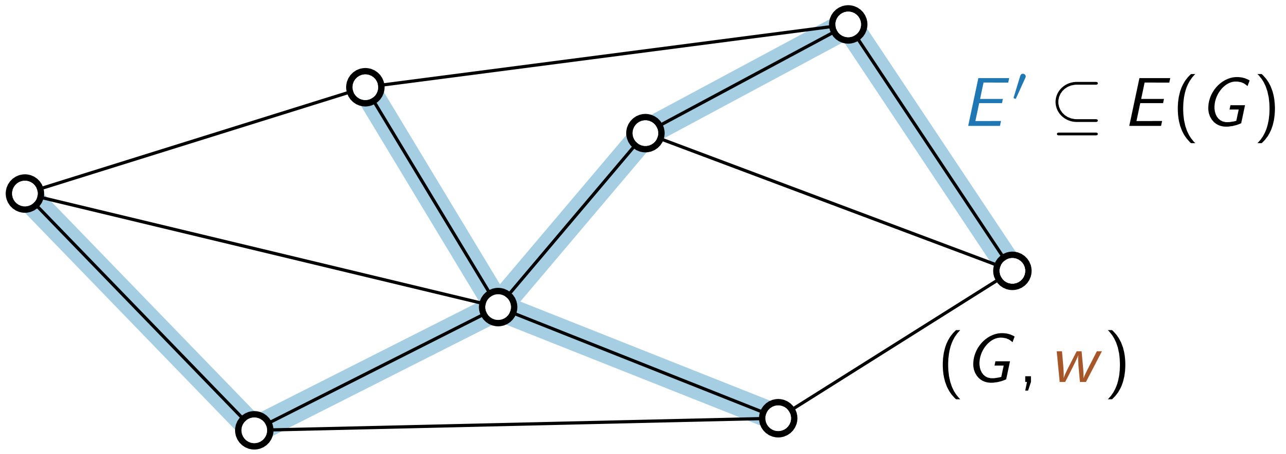
Wegen der Minimalität von $w(E')$ gilt:

T hat keine Kreise $\Rightarrow$ T ist ein Wald

T „erbt“ Zusammenhang von G $\Rightarrow$ T ist ein Baum

T spannt G auf

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn



Minimaler Spannbaum

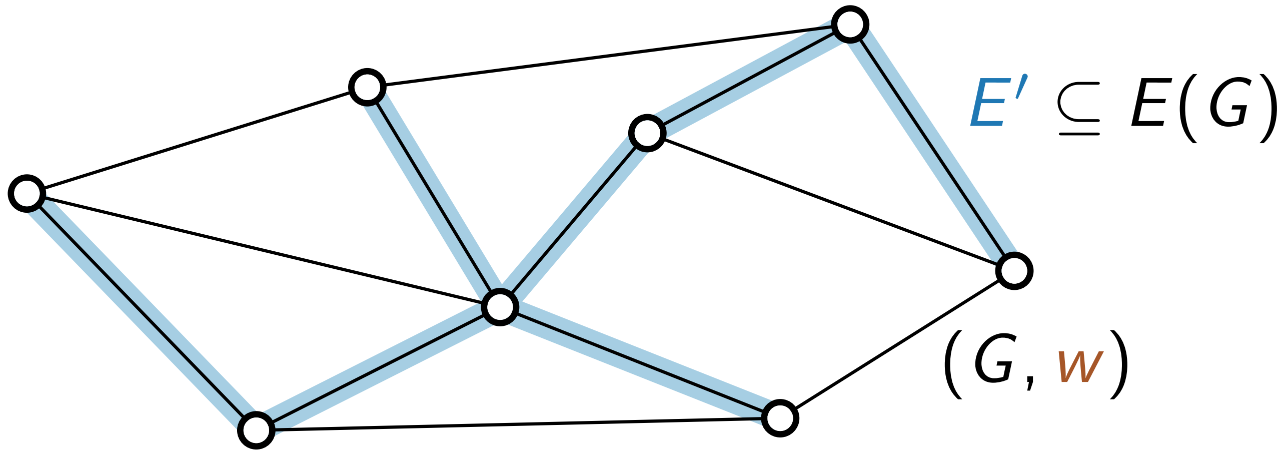
Wegen der Minimalität von $w(E')$ gilt:

T hat keine Kreise $\Rightarrow$ T ist ein Wald

T „erbt“ Zusammenhang von G $\Rightarrow$ T ist ein Baum

T spannt G auf $\Rightarrow$ T ist Spannbaum von G

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn



Minimaler Spannbaum

Wegen der Minimalität von $w(E')$ gilt:

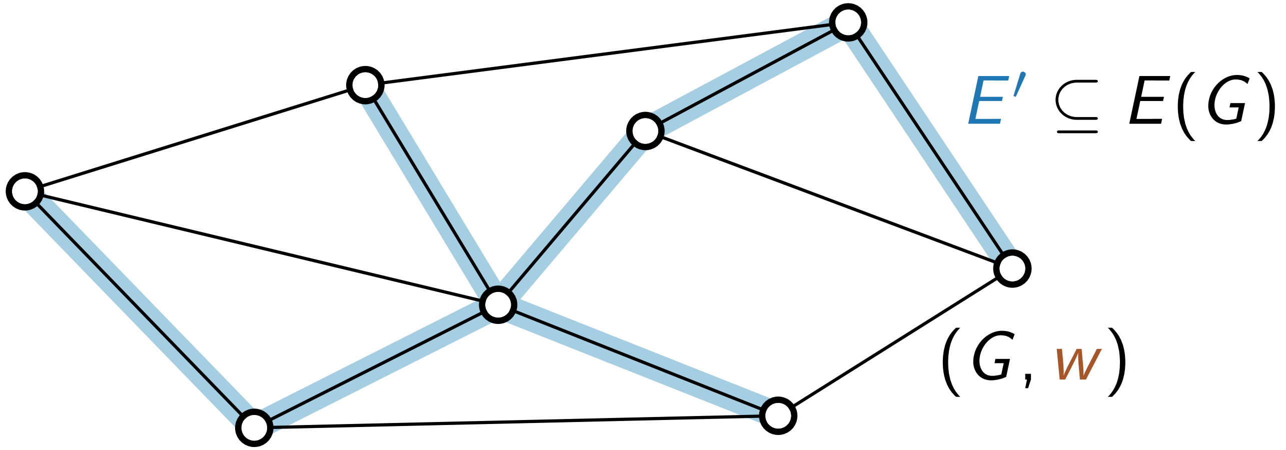
T hat keine Kreise $\Rightarrow$ T ist ein Wald

T „erbt" Zusammenhang von G $\Rightarrow$ T ist ein Baum

T spannt G auf $\Rightarrow$ T ist Spannbaum von G

T hat minimales Gewicht unter *allen* Spannbäumen von G.

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn



Minimaler Spannbaum

Wegen der Minimalität von $w(E')$ gilt:

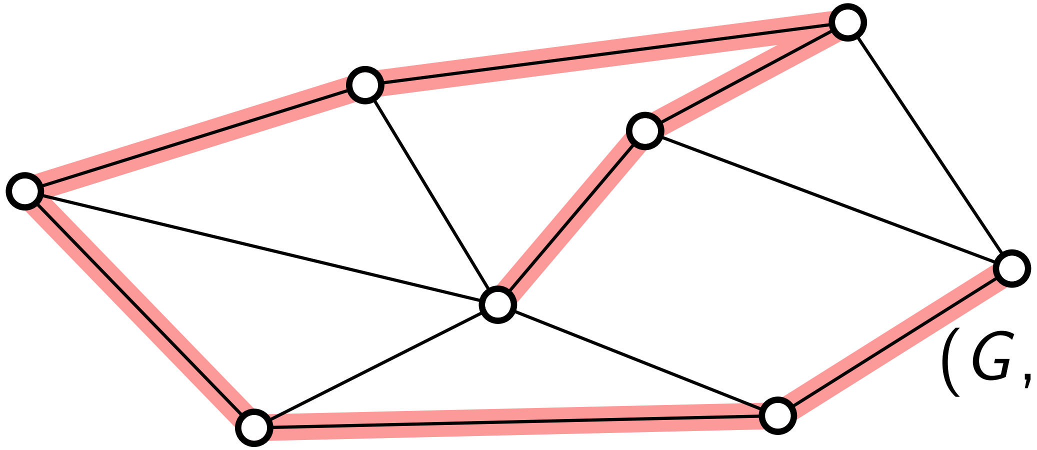
T hat keine Kreise $\Rightarrow$ T ist ein Wald

T „erbt“ Zusammenhang von G $\Rightarrow$ T ist ein Baum

T spannt G auf $\Rightarrow$ T ist Spannbaum von G

T hat minimales Gewicht unter *allen* Spannbäumen von G.

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn



(G, w)

Minimaler Spannbaum

Wegen der Minimalität von $w(E')$ gilt:

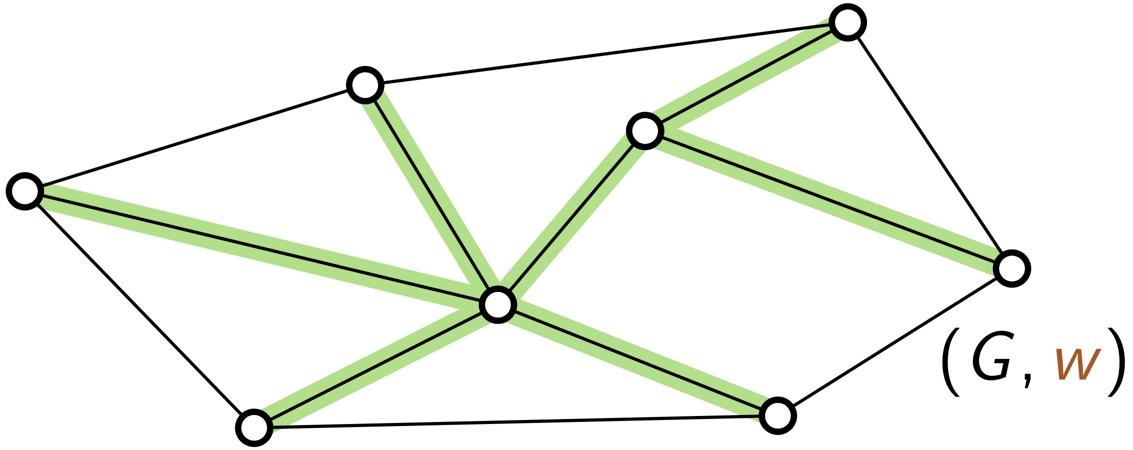
T hat keine Kreise $\Rightarrow$ T ist ein Wald

T „erbt“ Zusammenhang von G $\Rightarrow$ T ist ein Baum

T spannt G auf $\Rightarrow$ T ist Spannbaum von G

T hat minimales Gewicht unter *allen* Spannbäumen von G.

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn



Minimaler Spannbaum

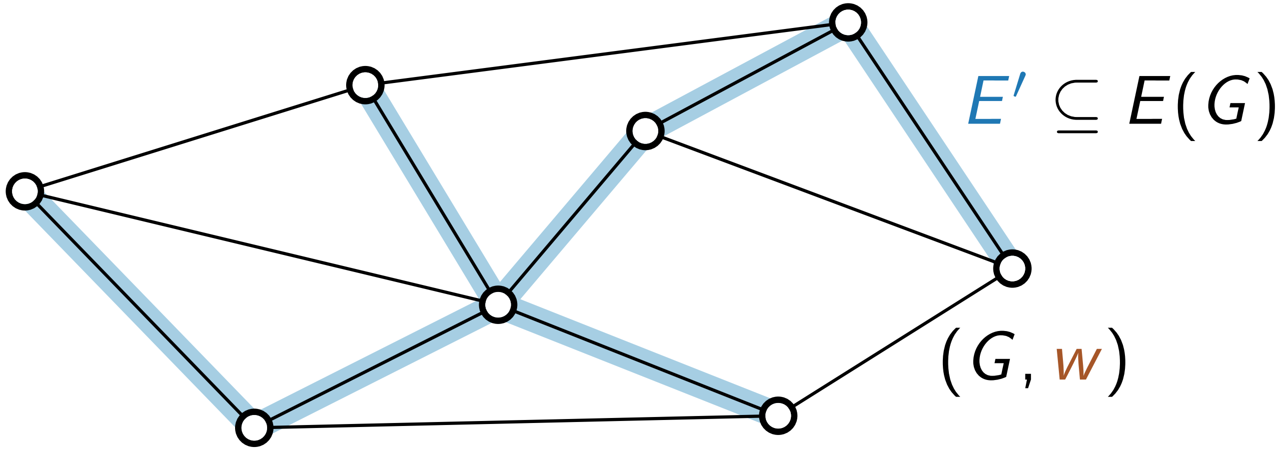
Wegen der Minimalität von $w(E')$ gilt:

T hat keine Kreise	$\Rightarrow$	T ist ein Wald
T „erbt“ Zusammenhang von G	$\Rightarrow$	T ist ein Baum
T spannt G auf	$\Rightarrow$	T ist Spannbaum von G

T hat minimales Gewicht unter *allen* Spannbäumen von G.

Wir nennen T kurz **minimalen Spannbaum (MSB)** von G.

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn



Minimaler Spannbaum

Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn

Wegen der Minimalität von $w(E')$ gilt:

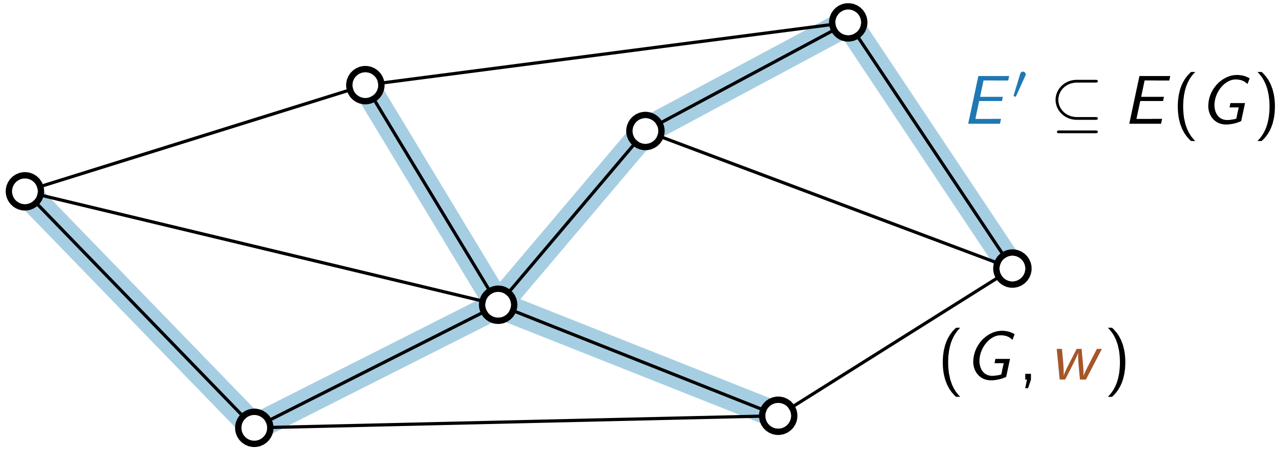
T hat keine Kreise $\Rightarrow$ T ist ein Wald

T „erbt" Zusammenhang von G $\Rightarrow$ T ist ein Baum

T spannt G auf $\Rightarrow$ T ist Spannbaum von G

T hat minimales Gewicht unter *allen* Spannbäumen von G.

Wir nennen T kurz **minimalen Spannbaum (MSB)** von G.



Beob. $|E'| =$ **?**

Minimaler Spannbaum

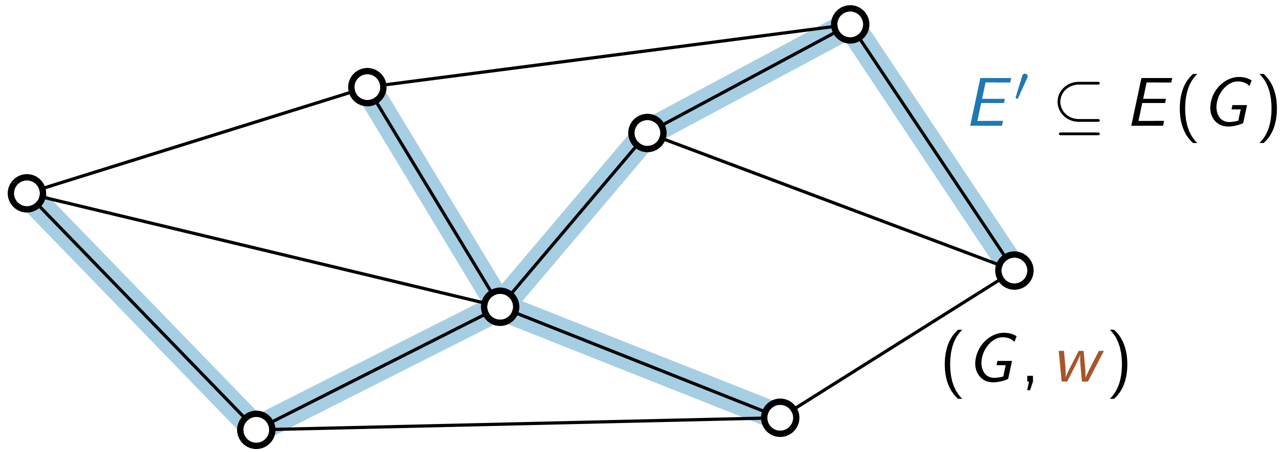
Wegen der Minimalität von $w(E')$ gilt:

T hat keine Kreise	$\Rightarrow$	T ist ein Wald
T „erbt“ Zusammenhang von G	$\Rightarrow$	T ist ein Baum
T spannt G auf	$\Rightarrow$	T ist Spannbaum von G

T hat minimales Gewicht unter *allen* Spannbäumen von G.

Wir nennen T kurz **minimalen Spannbaum (MSB)** von G.

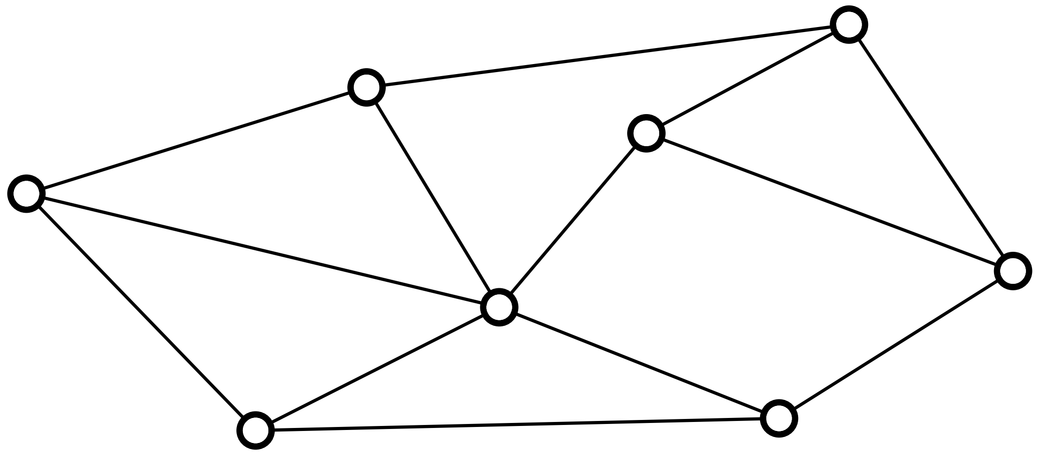
Otakar Borůvka
*1899 Ostroh, Mähren
† 1995 Brünn



Beob. $|E'| = |V(G)| - 1$

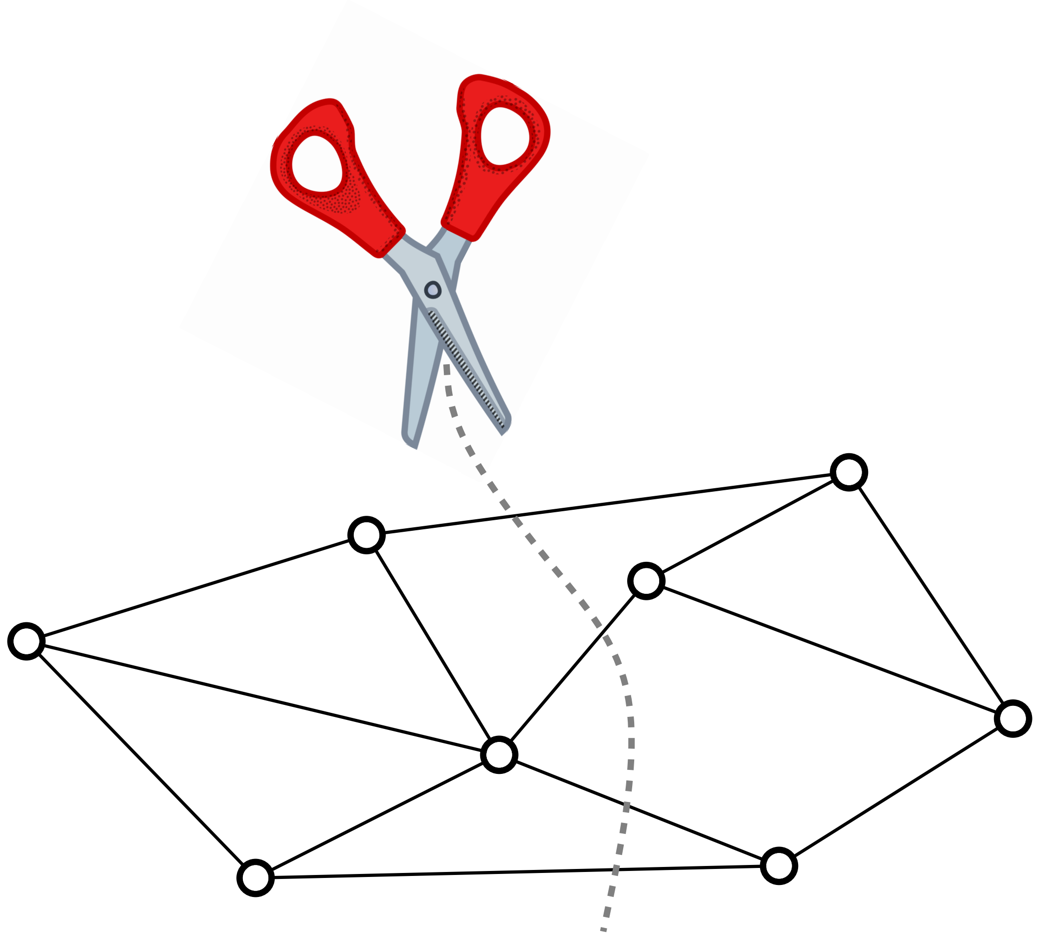
Schnitte

Def. Ein **Schnitt** $(S, V(G) \setminus S)$ eines Graphen G ist eine Zerlegung von $V(G)$ in zwei Teilmengen.



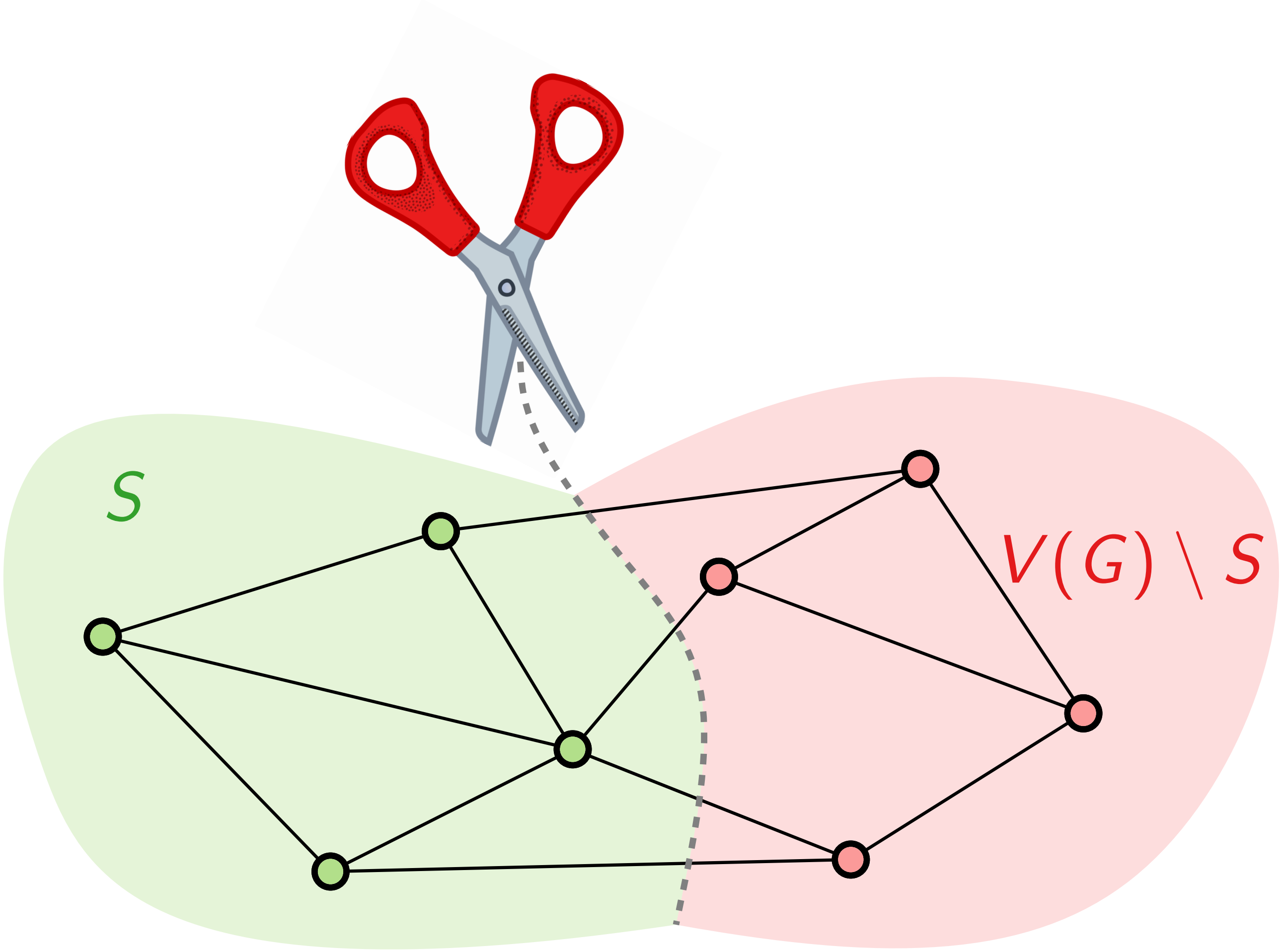
Schnitte

Def. Ein **Schnitt** $(S, V(G) \setminus S)$ eines Graphen G ist eine Zerlegung von $V(G)$ in zwei Teilmengen.



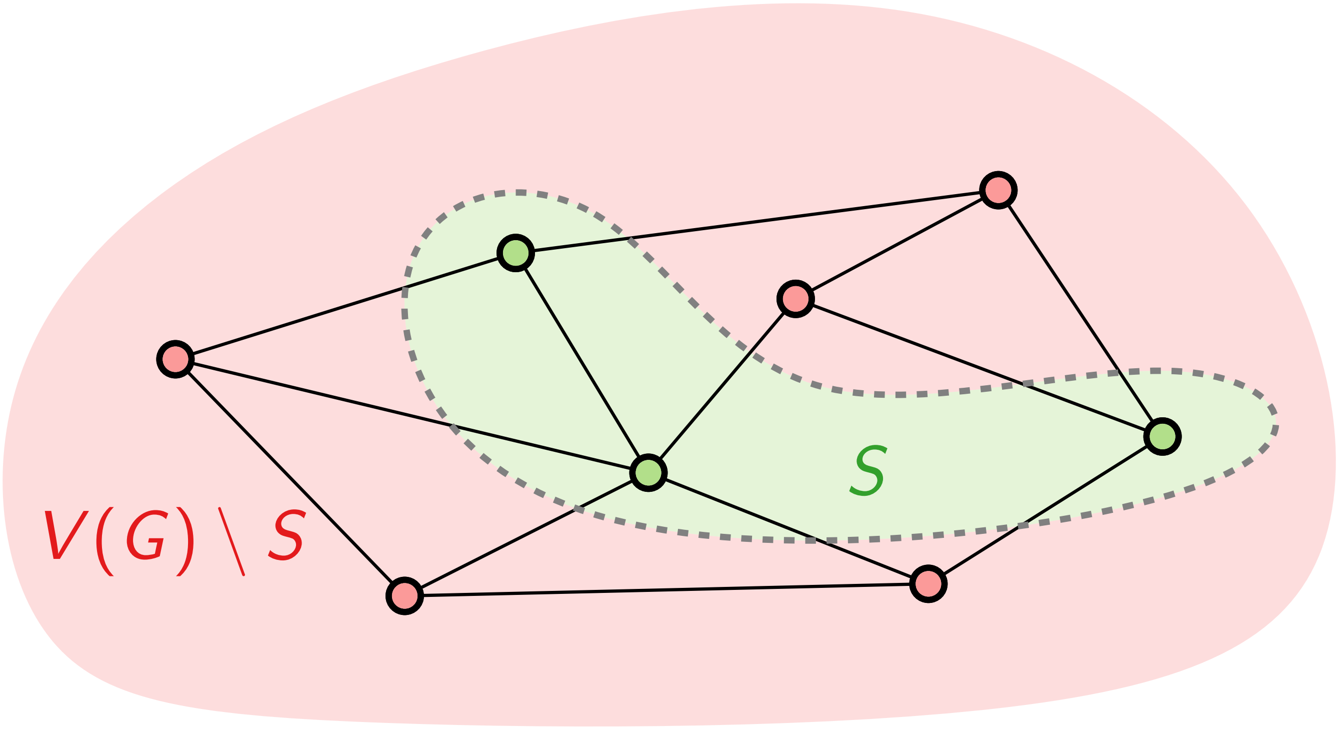
Schnitte

Def. Ein **Schnitt** $(S, V(G) \setminus S)$ eines Graphen G ist eine Zerlegung von $V(G)$ in zwei Teilmengen.



Schnitte

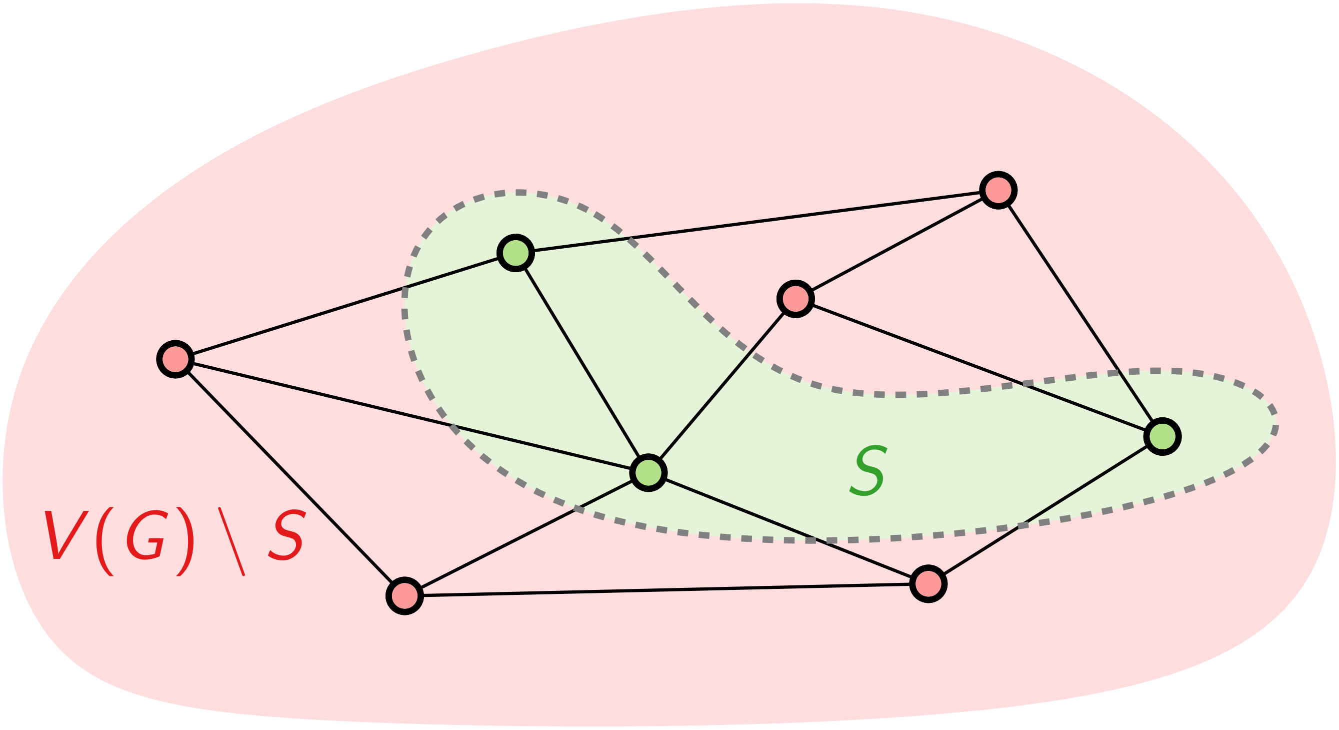
Def. Ein **Schnitt** $(S, V(G) \setminus S)$ eines Graphen G ist eine Zerlegung von $V(G)$ in zwei Teilmengen.



Schnitte

Def. Ein **Schnitt** $(S, V(G) \setminus S)$ eines Graphen G ist eine Zerlegung von $V(G)$ in zwei Teilmengen.

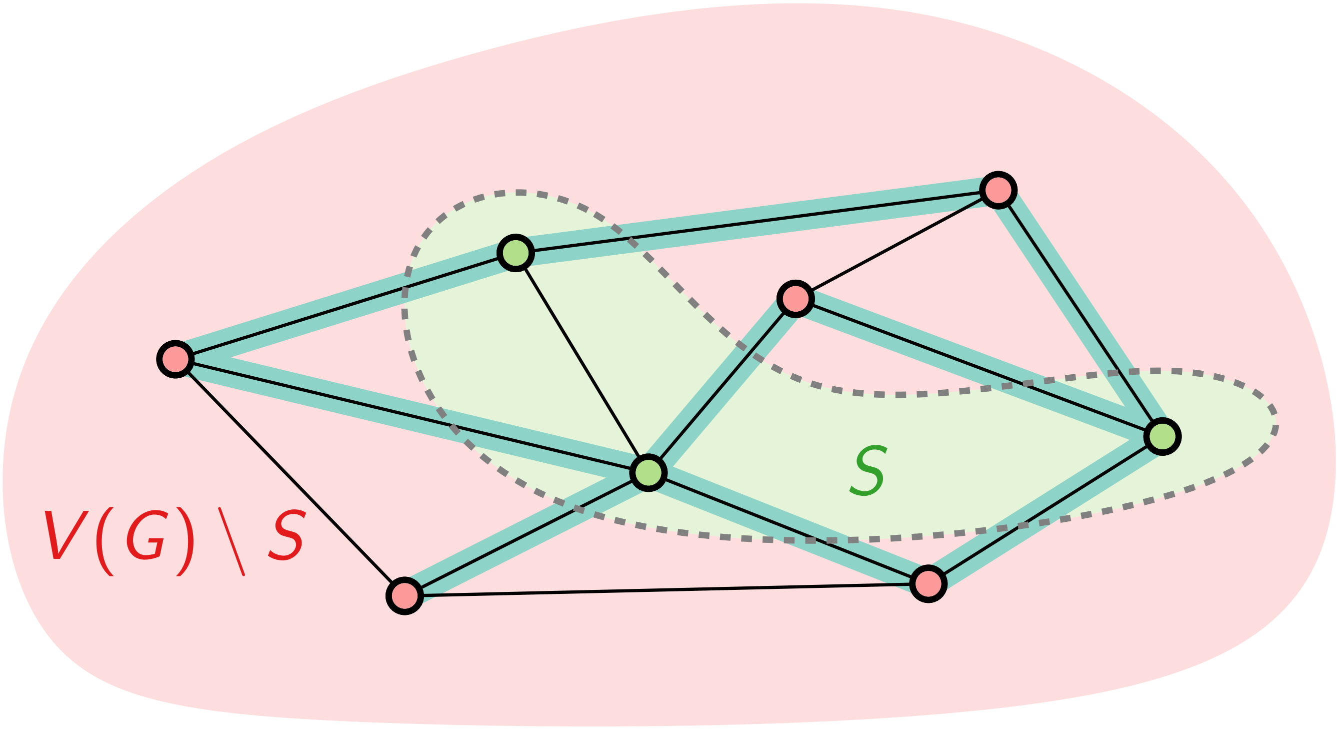
Eine Kante uv **kreuzt** $(S, V(G) \setminus S)$, wenn $u \in S$ und $v \in V(G) \setminus S$ (oder andersherum).



Schnitte

Def. Ein **Schnitt** $(S, V(G) \setminus S)$ eines Graphen G ist eine Zerlegung von $V(G)$ in zwei Teilmengen.

Eine Kante uv **kreuzt** $(S, V(G) \setminus S)$, wenn $u \in S$ und $v \in V(G) \setminus S$ (oder andersherum).

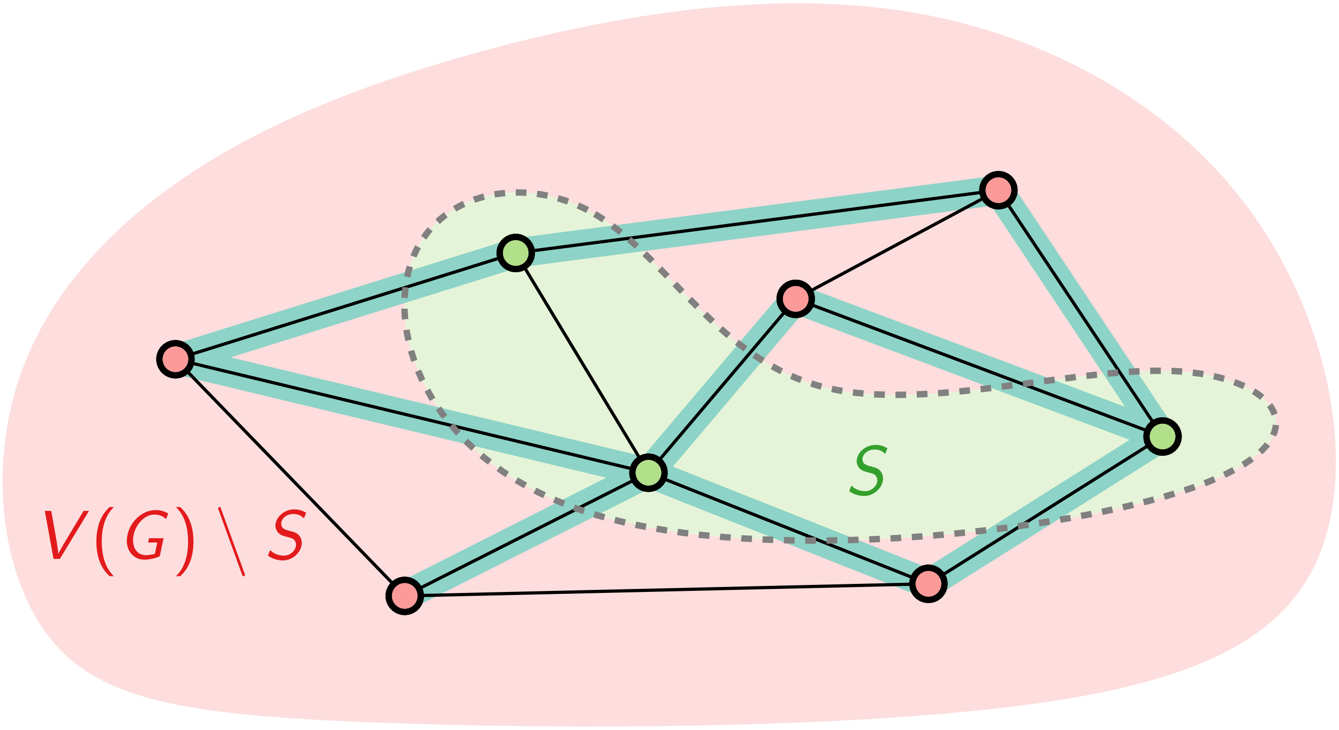


Schnitte

Def. Ein **Schnitt** $(S, V(G) \setminus S)$ eines Graphen G ist eine Zerlegung von $V(G)$ in zwei Teilmengen.

Eine Kante uv **kreuzt** $(S, V(G) \setminus S)$, wenn $u \in S$ und $v \in V(G) \setminus S$ (oder andersherum).

Eine Kante uv, die einen Schnitt kreuzt, ist **leicht**, wenn alle Kanten, die den Schnitt kreuzen, mindestens $w(uv)$ wiegen.

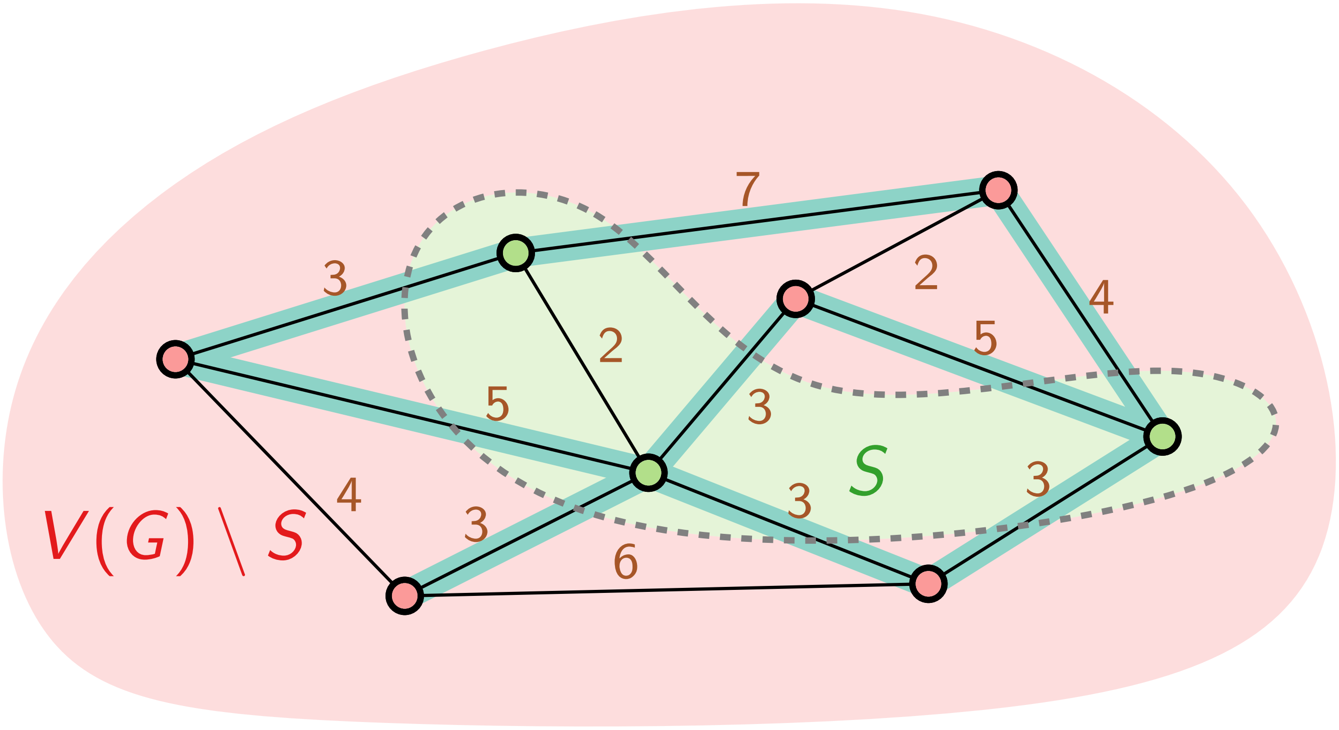


Schnitte

Def. Ein **Schnitt** $(S, V(G) \setminus S)$ eines Graphen G ist eine Zerlegung von $V(G)$ in zwei Teilmengen.

Eine Kante uv **kreuzt** $(S, V(G) \setminus S)$, wenn $u \in S$ und $v \in V(G) \setminus S$ (oder andersherum).

Eine Kante uv, die einen Schnitt kreuzt, ist **leicht**, wenn alle Kanten, die den Schnitt kreuzen, mindestens $w(uv)$ wiegen.

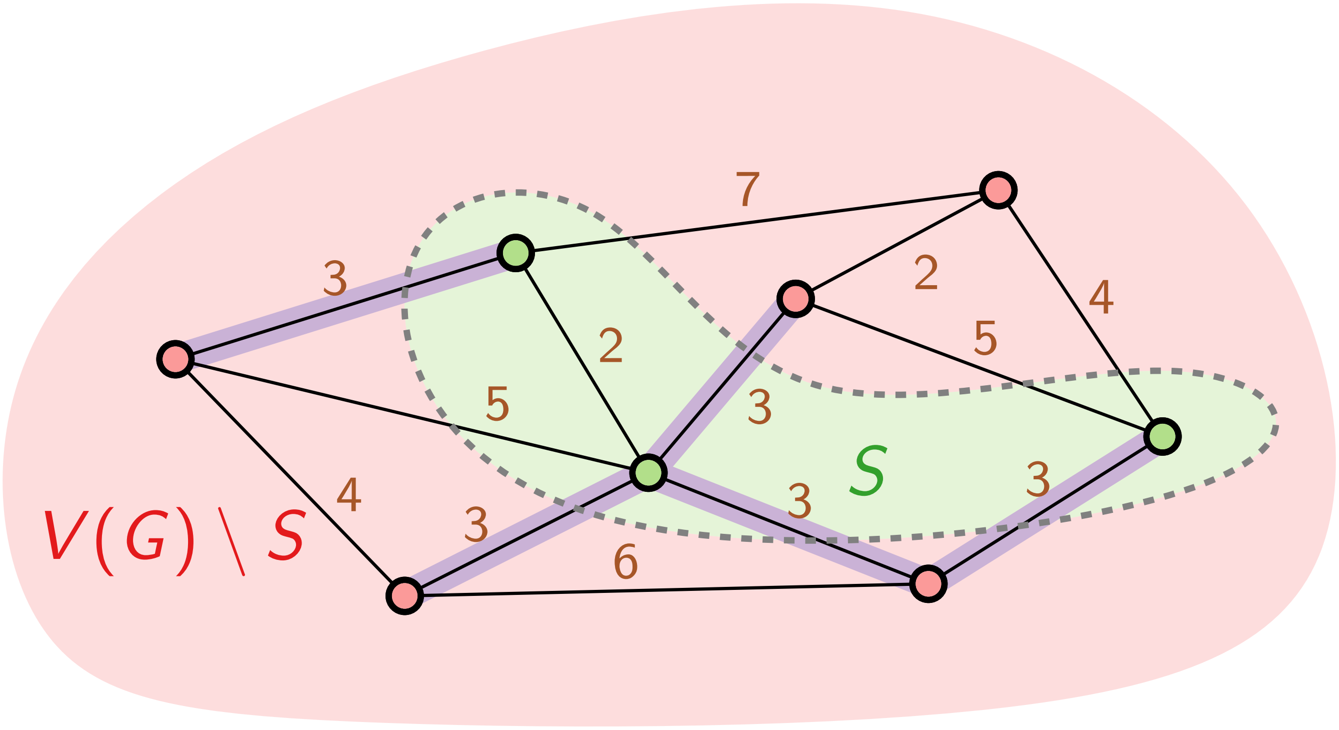


Schnitte

Def. Ein **Schnitt** $(S, V(G) \setminus S)$ eines Graphen G ist eine Zerlegung von $V(G)$ in zwei Teilmengen.

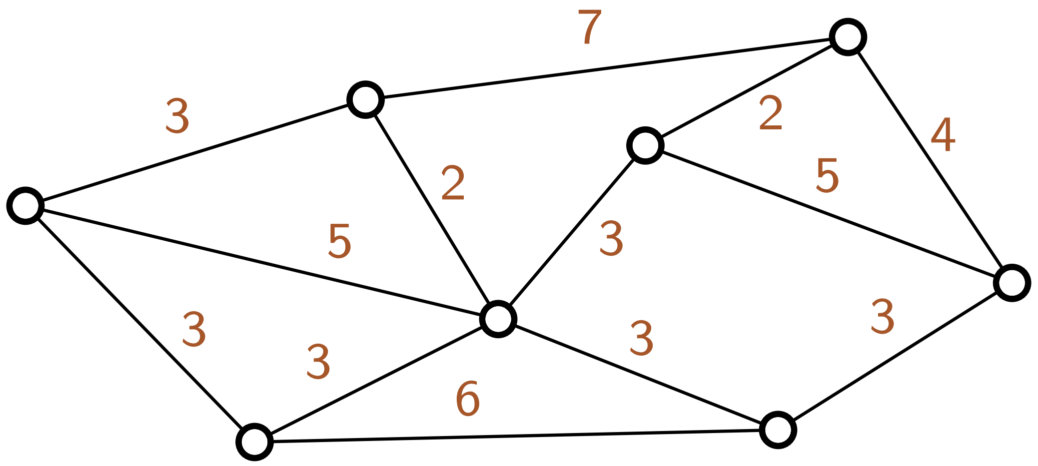
Eine Kante uv **kreuzt** $(S, V(G) \setminus S)$, wenn $u \in S$ und $v \in V(G) \setminus S$ (oder andersherum).

Eine Kante uv, die einen Schnitt kreuzt, ist **leicht**, wenn alle Kanten, die den Schnitt kreuzen, mindestens $w(uv)$ wiegen.



Allgemeiner Greedy-Algorithmus

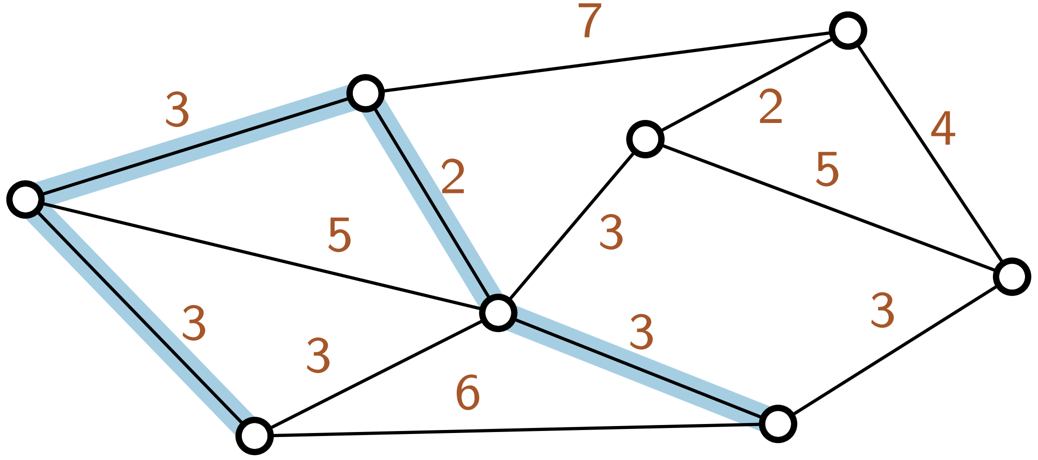
Färbe alle Kanten des Graphen:



Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

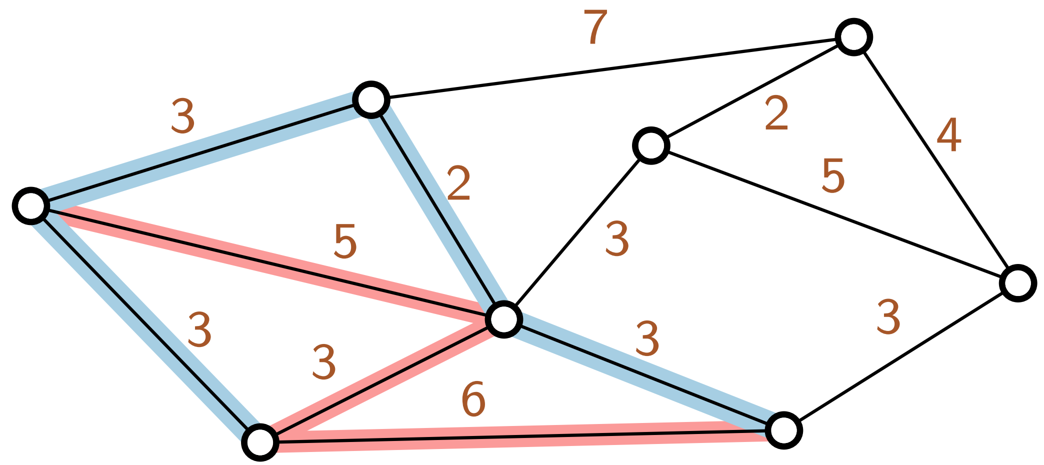
- blau: Kante für den MSB



Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

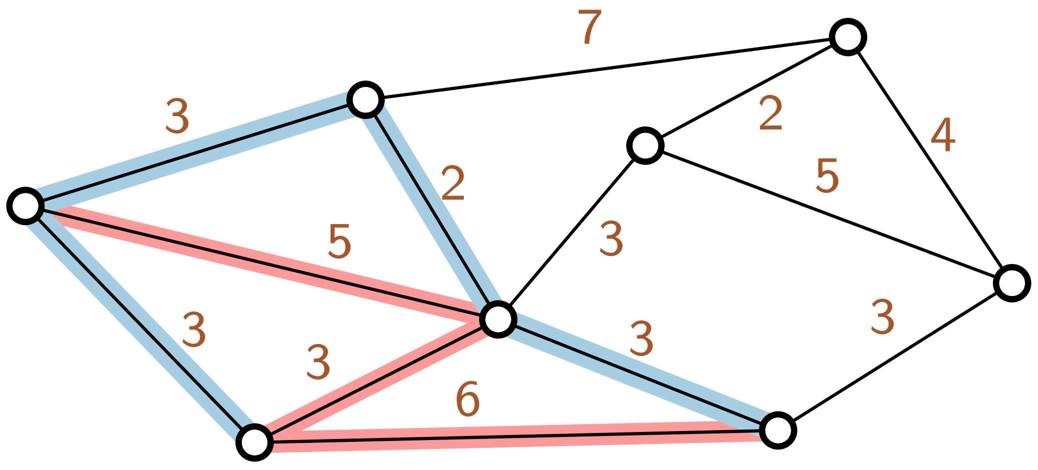
- blau: Kante für den MSB
- rot: Kante nicht für den MSB



Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

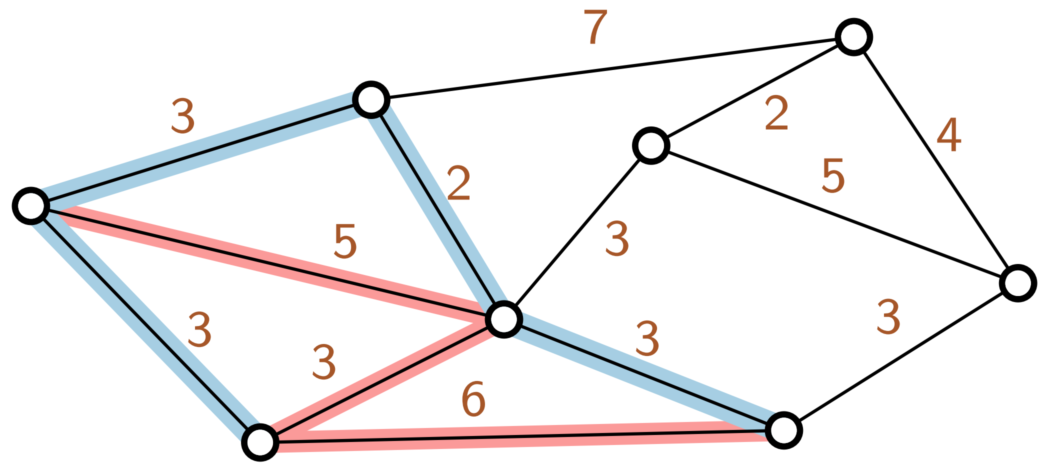
- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden



Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden

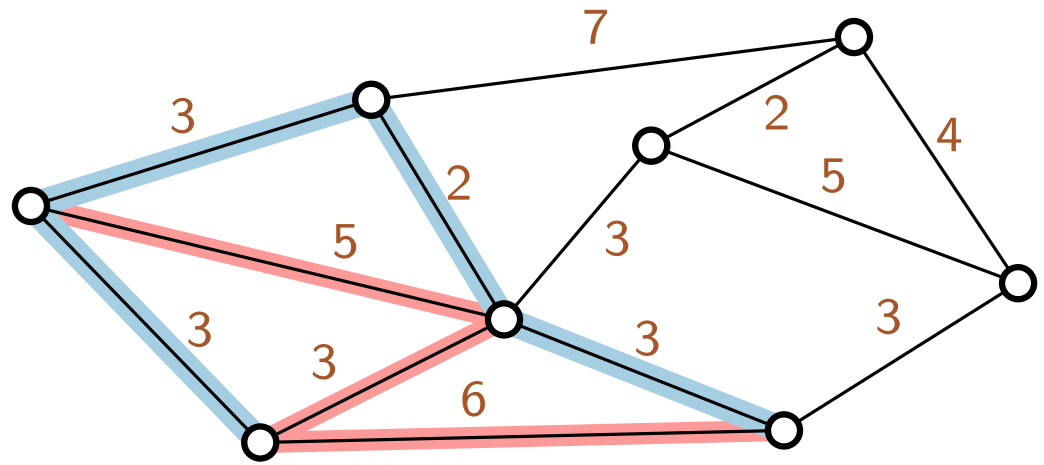


Verwende zwei Regeln:

Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden



Verwende zwei Regeln:

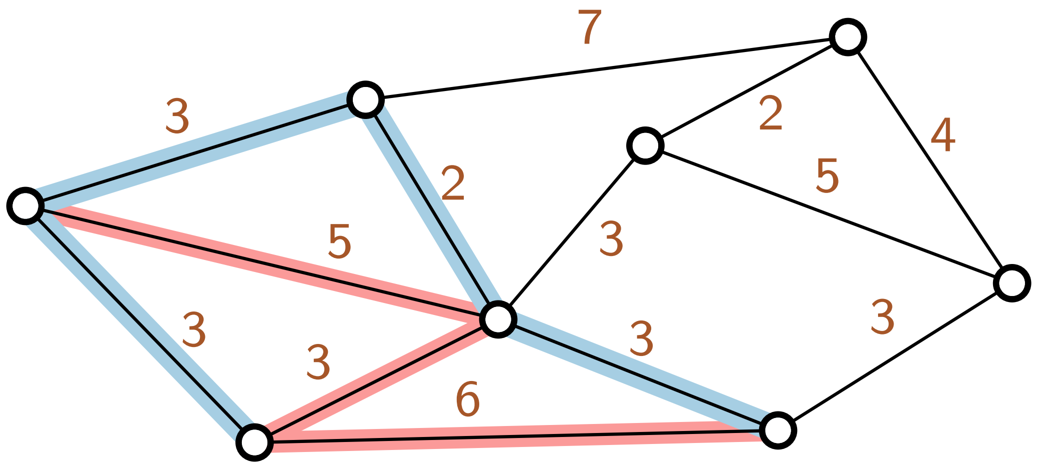
Blaue Regel:

Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden

Verwende zwei Regeln:



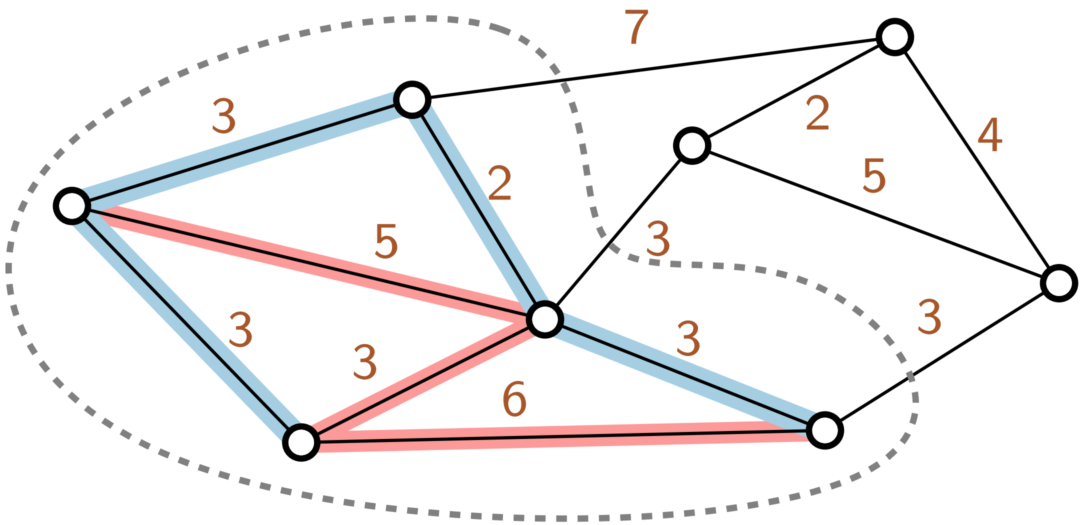
Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt

Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden

Verwende zwei Regeln:



Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt

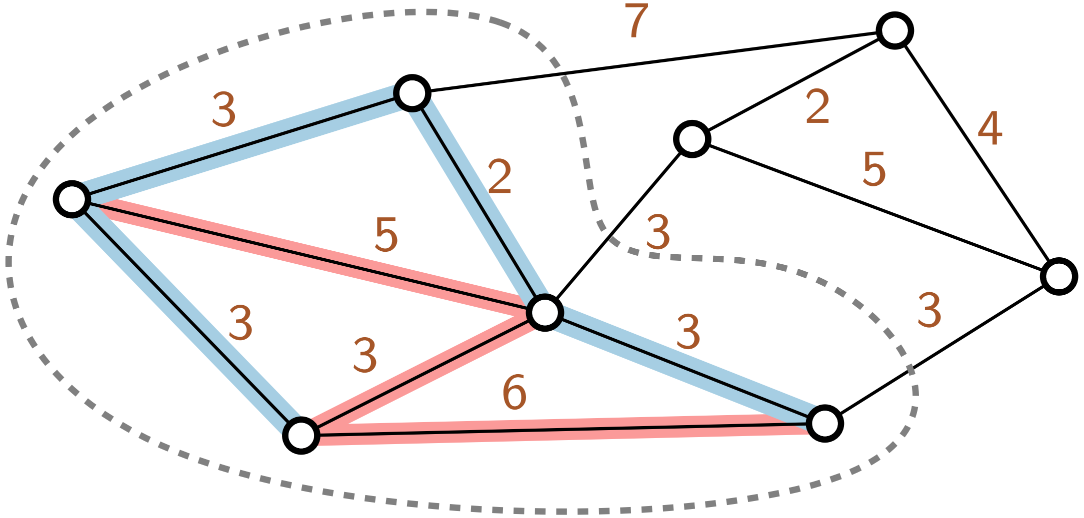
Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden

Verwende zwei Regeln:

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

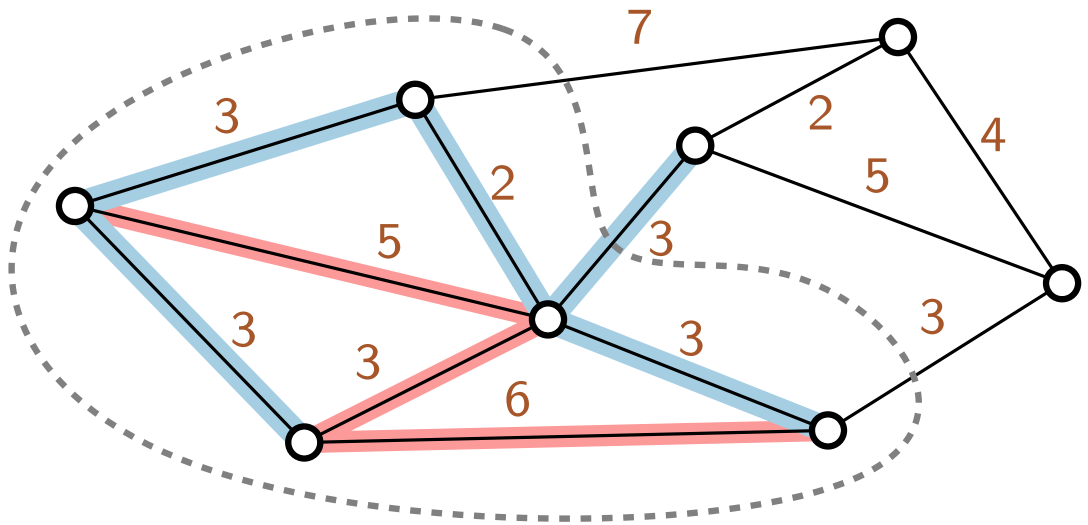


Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden

Verwende zwei Regeln:

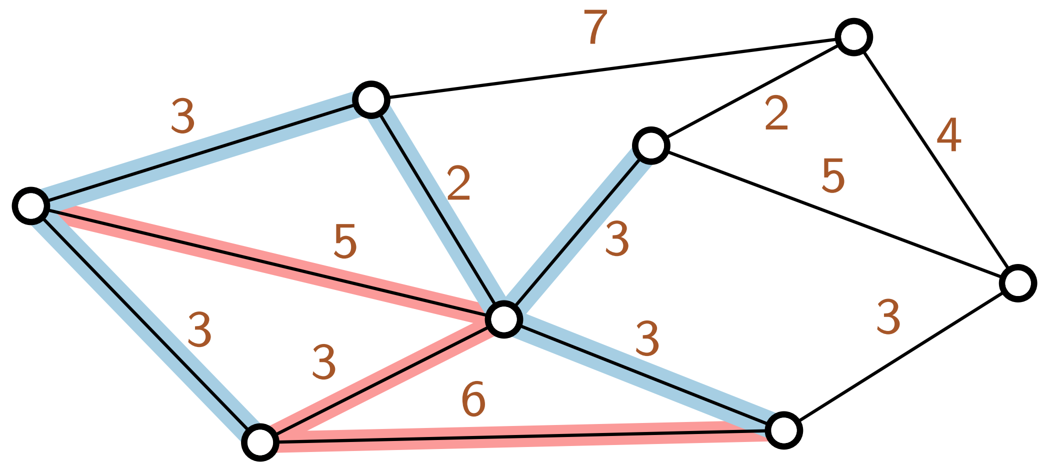


Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden



Verwende zwei Regeln:

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

Rote Regel:

Allgemeiner Greedy-Algorithmus

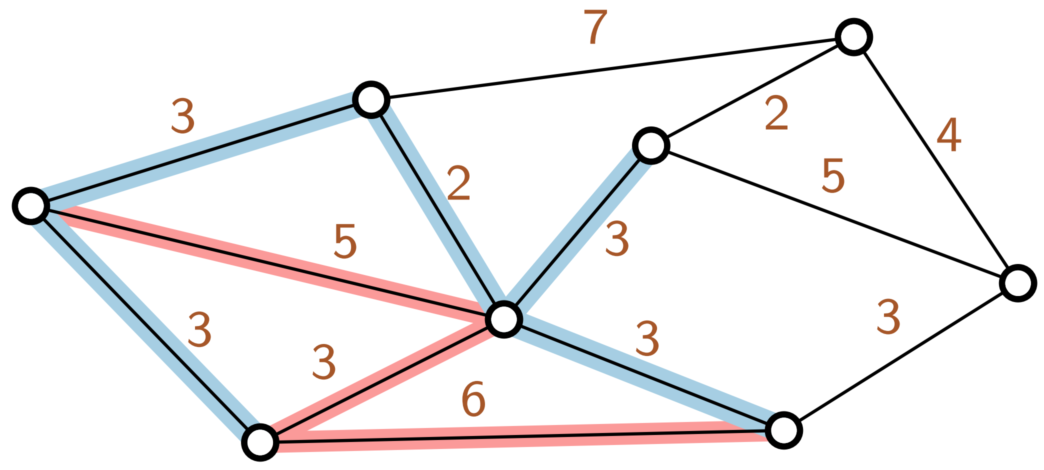
Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden

Verwende zwei Regeln:

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

Rote Regel:
Wähle Kreis ohne rote Kante

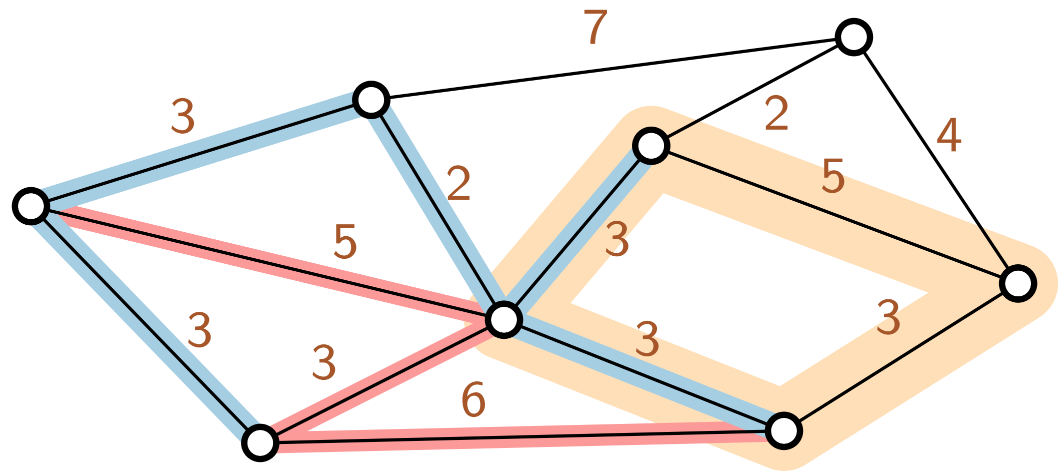


Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden

Verwende zwei Regeln:



Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

Rote Regel:
Wähle Kreis ohne rote Kante

Allgemeiner Greedy-Algorithmus

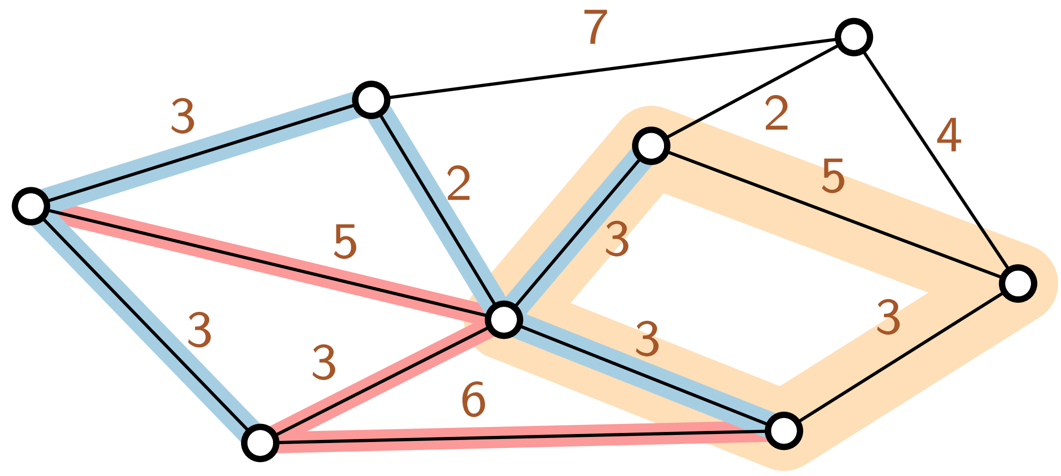
Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden

Verwende zwei Regeln:

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

Rote Regel:
Wähle Kreis ohne rote Kante
Färbe größte ungefärbte Kante auf Kreis rot

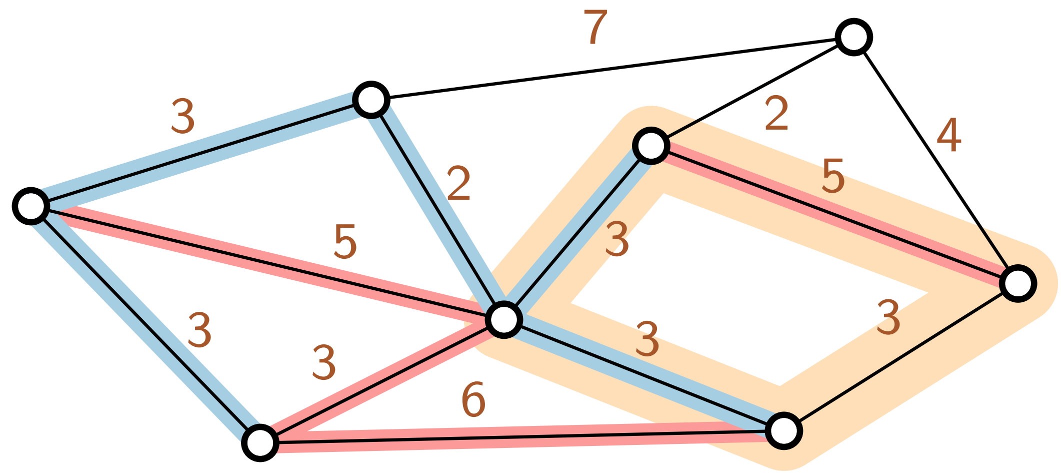


Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden

Verwende zwei Regeln:



Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

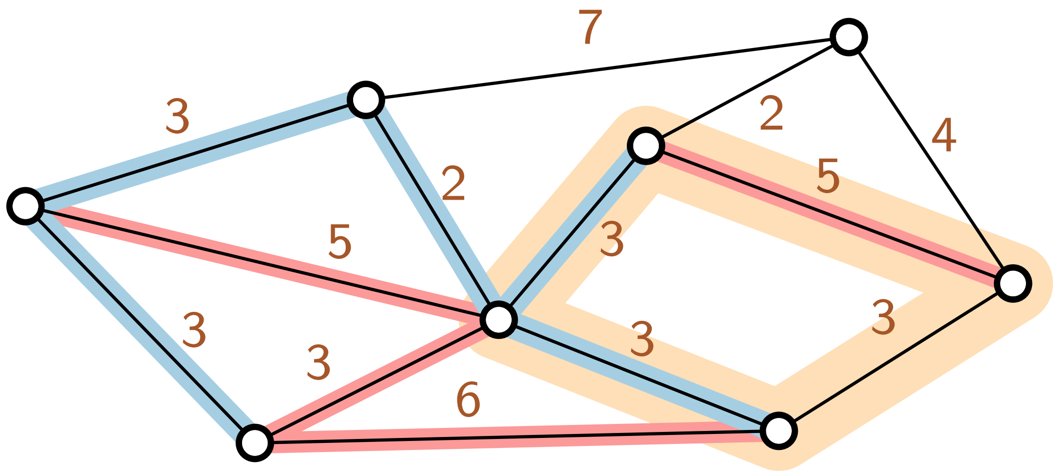
Rote Regel:
Wähle Kreis ohne rote Kante
Färbe größte ungefärbte Kante auf Kreis rot

Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden

Verwende zwei Regeln:



Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

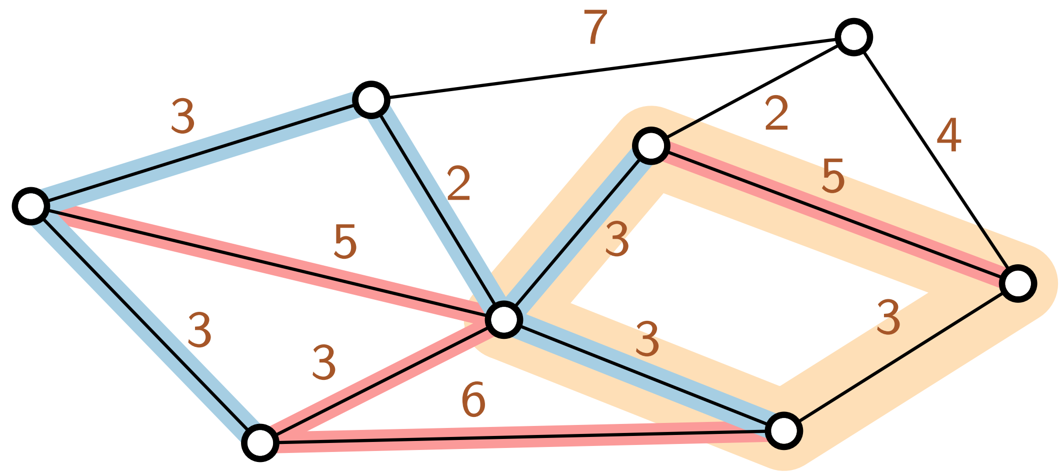
Rote Regel:
Wähle Kreis ohne rote Kante
Färbe größte ungefärbte Kante auf Kreis rot

GreedySpannbaum(G, w)

Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden



Verwende zwei Regeln:

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

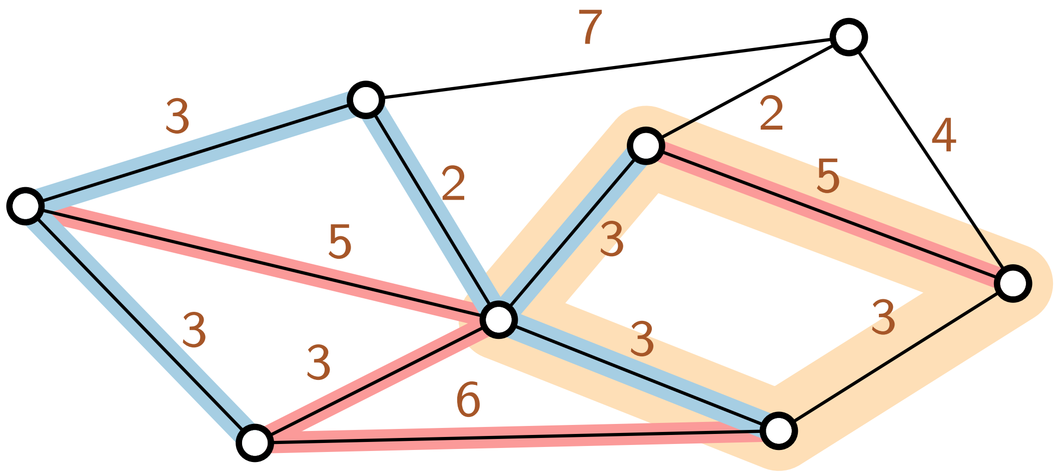
Rote Regel:
Wähle Kreis ohne rote Kante
Färbe größte ungefärbte Kante auf Kreis rot

GreedySpannbaum(G, w)
Wende blaue Regel oder rote Regel an,
bis alle Kanten gefärbt sind.

Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden



Verwende zwei Regeln:

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

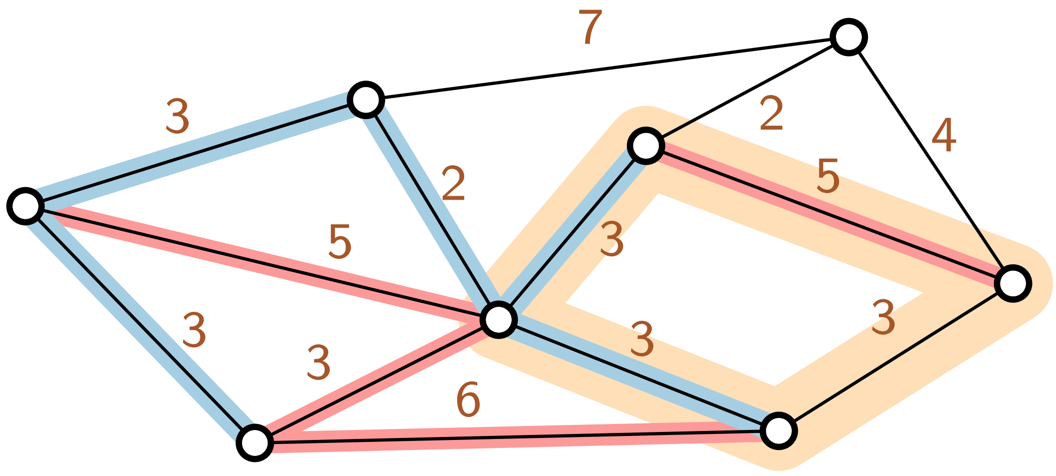
Rote Regel:
Wähle Kreis ohne rote Kante
Färbe größte ungefärbte Kante auf Kreis rot

GreedySpannbaum(G, w)
Wende blaue Regel oder rote Regel an,
bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück

Allgemeiner Greedy-Algorithmus

Färbe alle Kanten des Graphen:

- blau: Kante für den MSB
- rot: Kante nicht für den MSB
- ungefärbt: Noch nicht entschieden



Verwende zwei Regeln:

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt
Färbe leichte Kante blau

Rote Regel:
Wähle Kreis ohne rote Kante
Färbe größte ungefärbte Kante auf Kreis rot

GreedySpannbaum(G, w)
  Wende blaue Regel oder rote Regel an,
   bis alle Kanten gefärbt sind.
  Gib $E' = \{$blaue Kanten$\}$ zurück

Satz. GreedySpannbaum findet einen minimalen Spannbaum.

Beweis Greedy Algorithmus

Farbinvariante (FI): Es gibt einen MSB T:

$\textsc{GreedySpannbaum}(G, w)$

Wende blaue Regel oder rote Regel an,
bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück

Beweis Greedy Algorithmus

Farbinvariante (FI): Es gibt einen MSB T:

- T enthält alle blauen Kanten.

GreedySpannbaum(G, w)

Wende blaue Regel oder rote Regel an,
bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück

Beweis Greedy Algorithmus

Farbinvariante (FI): Es gibt einen MSB T:

- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

GREEDYSPANNBAUM(G, w)

Wende blaue Regel oder rote Regel an,
bis alle Kanten gefärbt sind.

Gib $E' = \{$blaue Kanten$\}$ zurück

Beweis Greedy Algorithmus

Farbinvariante (FI): Es gibt einen MSB T:

- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

FI ist am Anfang offensichtlich erfüllt.

GreedySpannbaum(G, w)
Wende blaue Regel oder rote Regel an, bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück

Beweis Greedy Algorithmus

GreedySpannbaum(G, w)

Wende blaue Regel oder rote Regel an, bis alle Kanten gefärbt sind.

Gib $E' = \{$blaue Kanten$\}$ zurück

Farbinvariante (FI): Es gibt einen MSB T:

- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

FI ist am Anfang offensichtlich erfüllt.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis Greedy Algorithmus

Farbinvariante (FI): Es gibt einen MSB T:

- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

FI ist am Anfang offensichtlich erfüllt.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

GreedySpannbaum(G, w)
Wende blaue Regel oder rote Regel an,
bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück

Beweis Greedy Algorithmus

GreedySpannbaum(G, w)
Wende blaue Regel oder rote Regel an, bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

FI ist am Anfang offensichtlich erfüllt.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis Greedy Algorithmus

$\textsc{GreedySpannbaum}(G, w)$
Wende blaue Regel oder rote Regel an,
bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

FI ist am Anfang offensichtlich erfüllt.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Lemma. GreedySpannbaum färbt alle Kanten.

Satz. GreedySpannbaum findet einen minimalen Spannbaum.

Beweis Greedy Algorithmus

GreedySpannbaum(G, w)
Wende blaue Regel oder rote Regel an,
bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

FI ist am Anfang offensichtlich erfüllt.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Lemma. GreedySpannbaum färbt alle Kanten.

Satz. GreedySpannbaum findet einen minimalen Spannbaum.

Beweis.
- Jede Kante ist entweder blau oder rot.
-

Beweis Greedy Algorithmus

$\textsc{GreedySpannbaum}(G, w)$
Wende blaue Regel oder rote Regel an,
bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

FI ist am Anfang offensichtlich erfüllt.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Lemma. $\textsc{GreedySpannbaum}$ färbt alle Kanten.

Satz. $\textsc{GreedySpannbaum}$ findet einen minimalen Spannbaum.

Beweis.
- Jede Kante ist entweder blau oder rot.
- Es gibt einen MSB T, der alle blauen Kanten und keine rote Kante enthält.

Beweis Greedy Algorithmus

GreedySpannbaum(G, w)
Wende blaue Regel oder rote Regel an,
bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

FI ist am Anfang offensichtlich erfüllt.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Lemma. GreedySpannbaum färbt alle Kanten.

Satz. GreedySpannbaum findet einen minimalen Spannbaum.

Beweis.
- Jede Kante ist entweder blau oder rot.
- Es gibt einen MSB T, der alle blauen Kanten und keine rote Kante enthält.

$\Rightarrow$ Blaue Kanten bilden MSB $\square$

Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:

- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:

Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:

- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:

Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis.

Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:

- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:

Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.

Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:

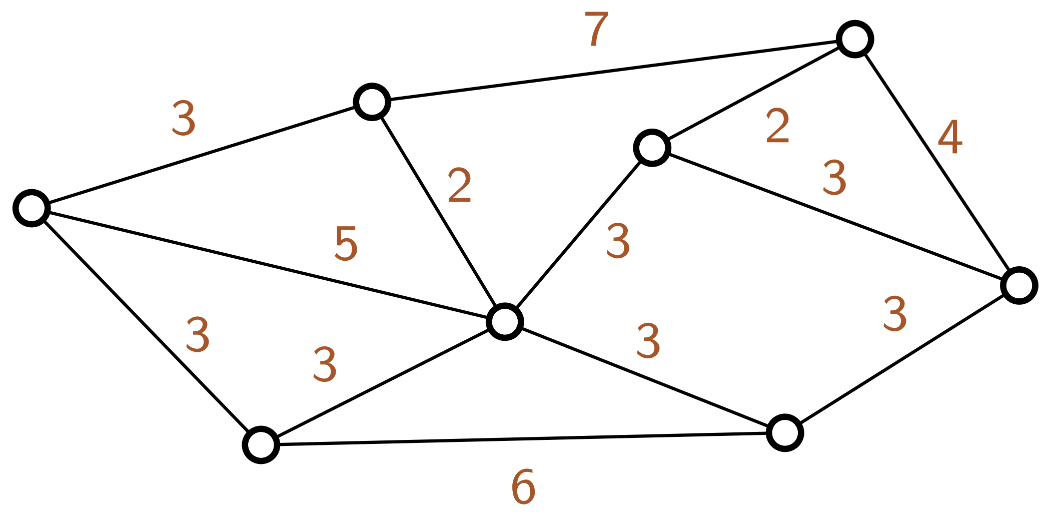
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:

Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.



Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:

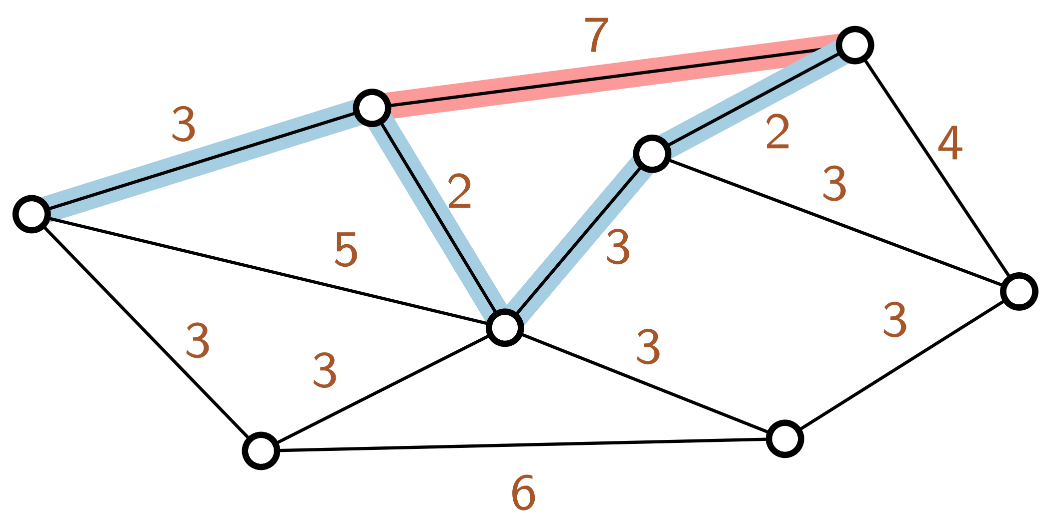
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:

Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.



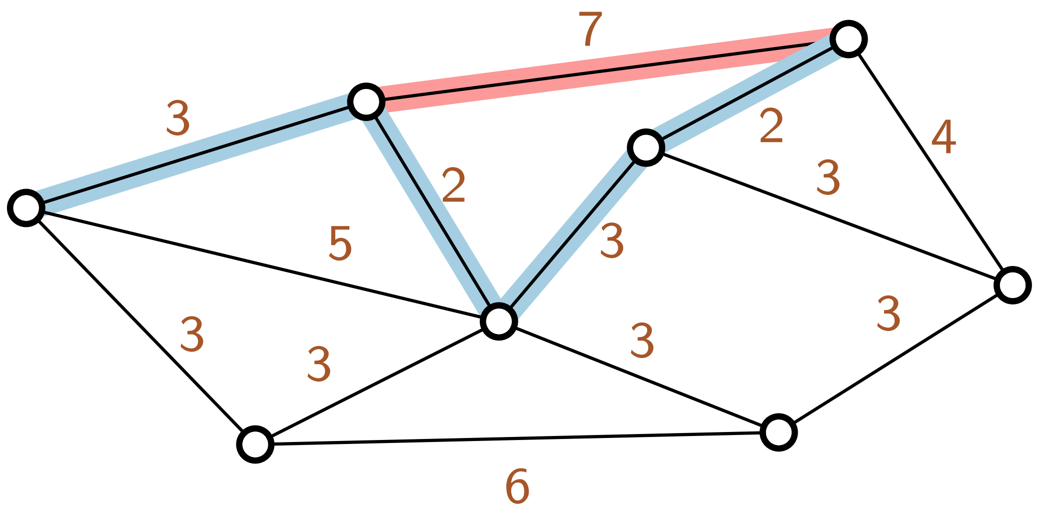
Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.



Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:

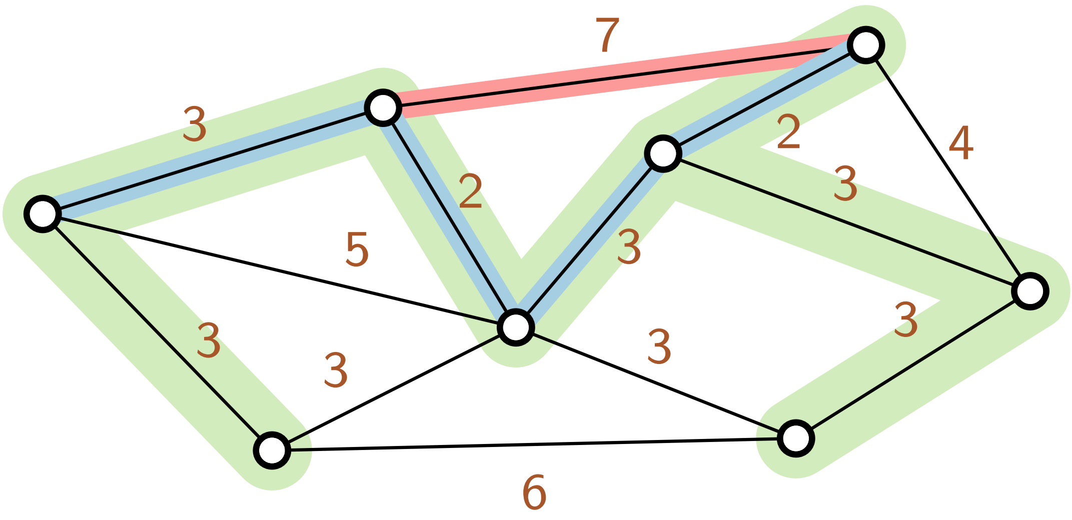
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:

Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.



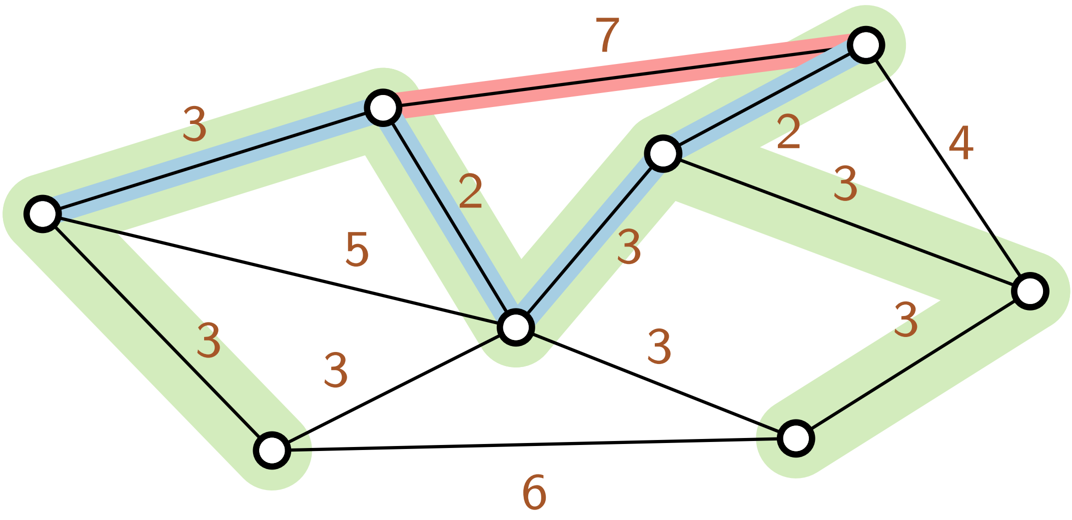
Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.



Beweis der blauen Regel

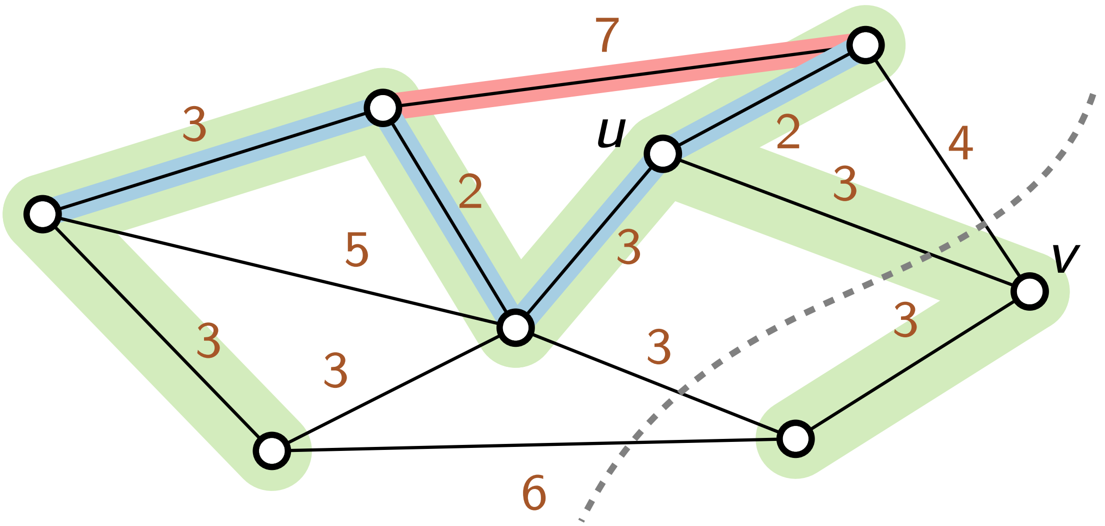
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$



Beweis der blauen Regel

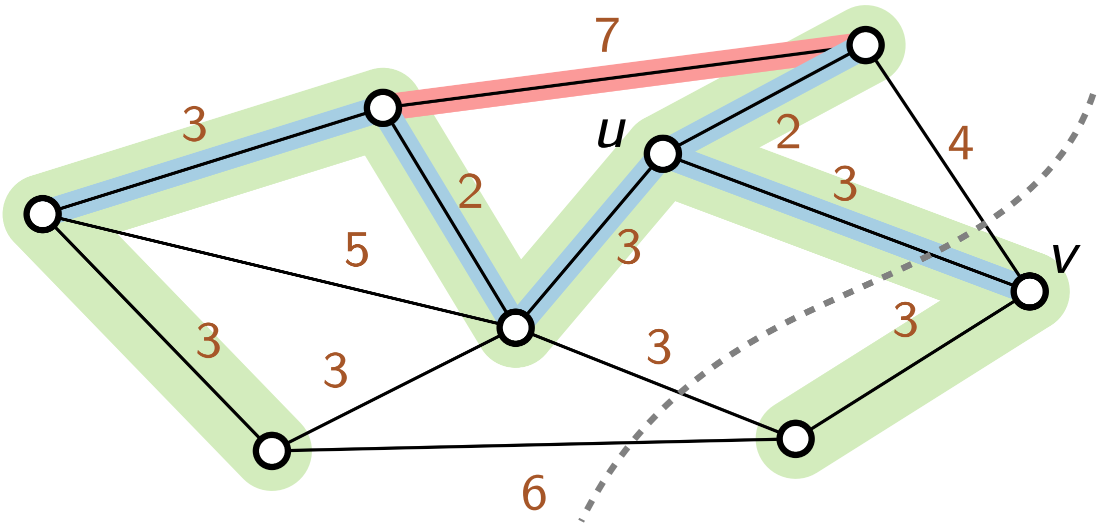
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$



Beweis der blauen Regel

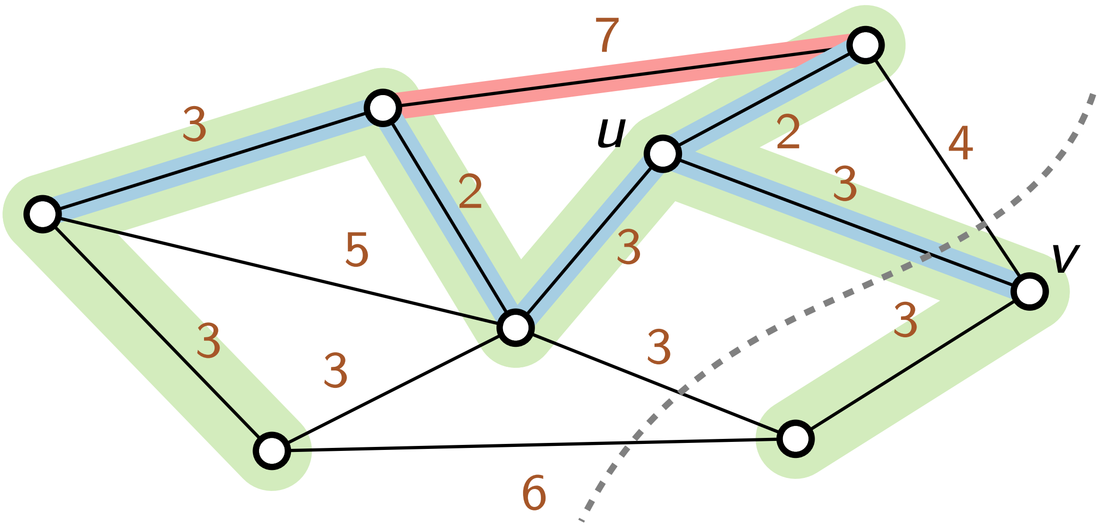
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.



Beweis der blauen Regel

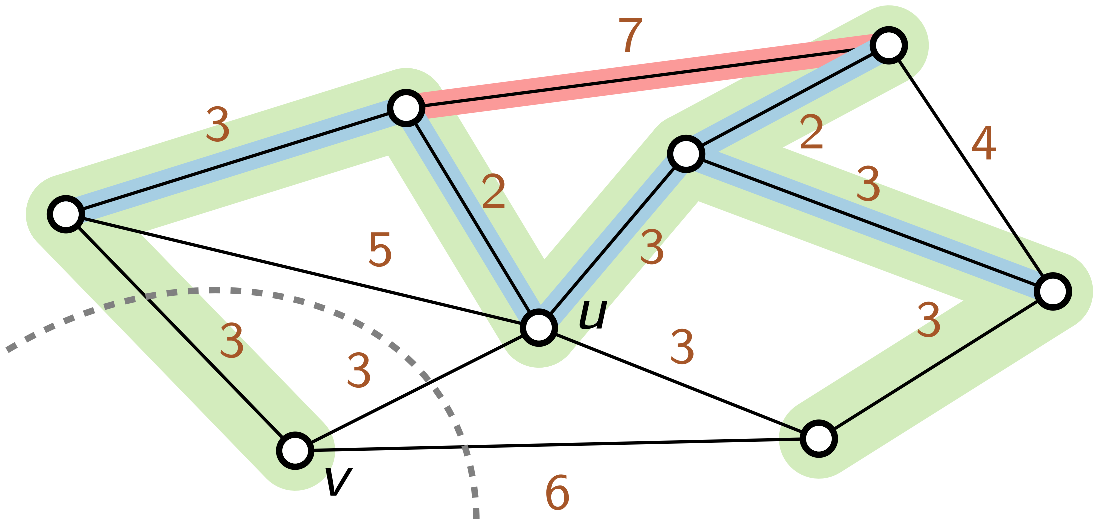
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$



Beweis der blauen Regel

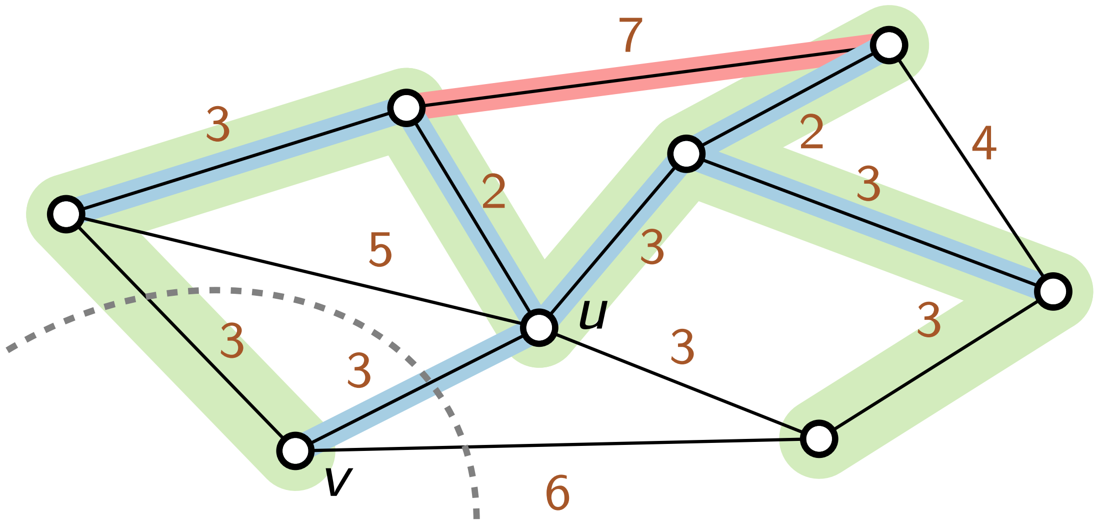
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$



Beweis der blauen Regel

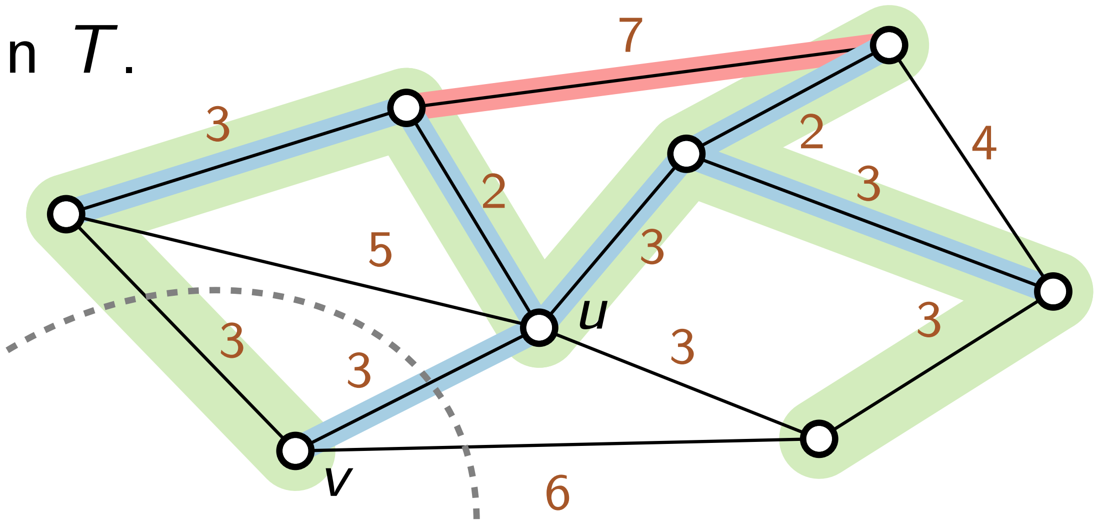
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.



Beweis der blauen Regel

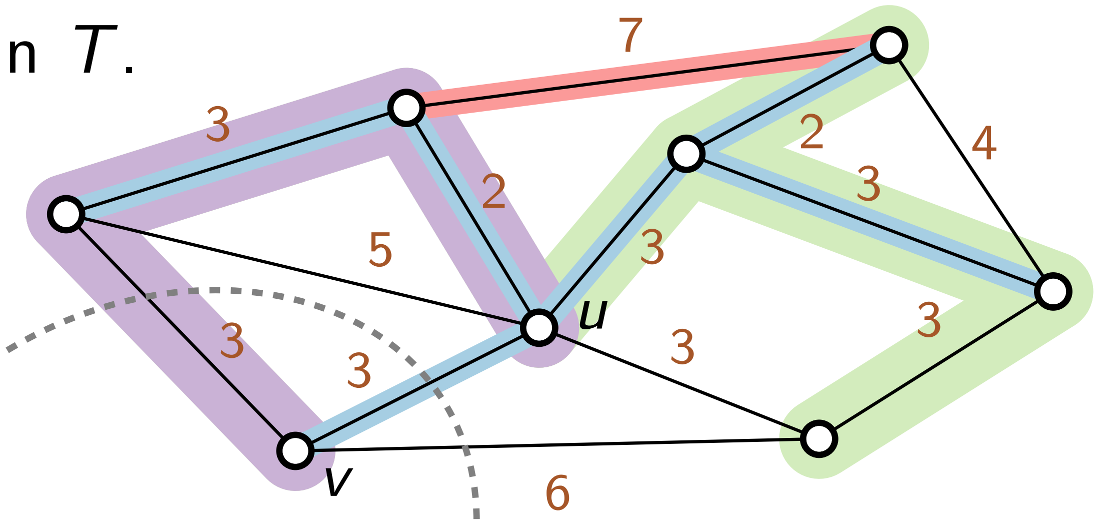
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.



Beweis der blauen Regel

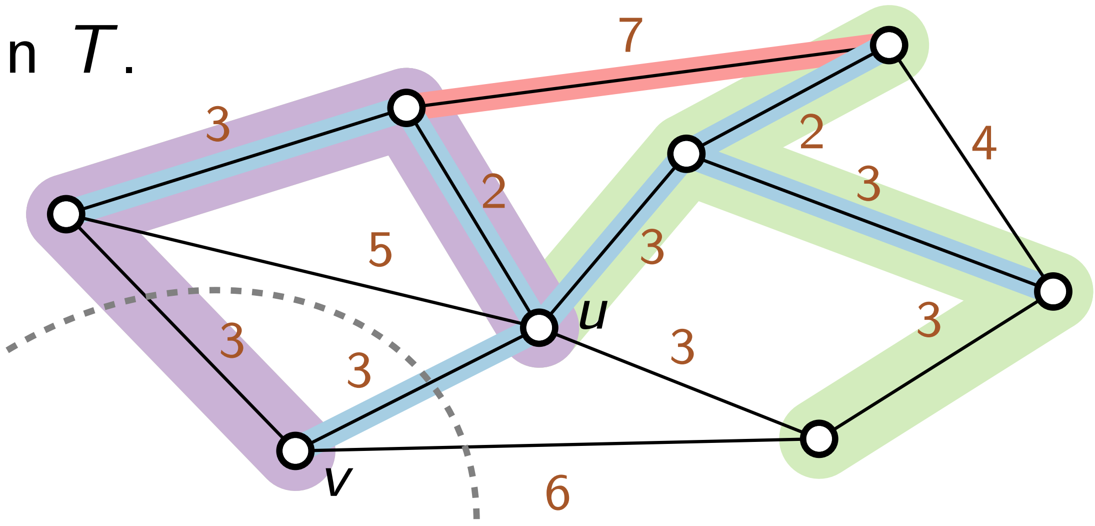
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.
   $\Rightarrow$ p enthält Kante xy, die Schnitt kreuzt



Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

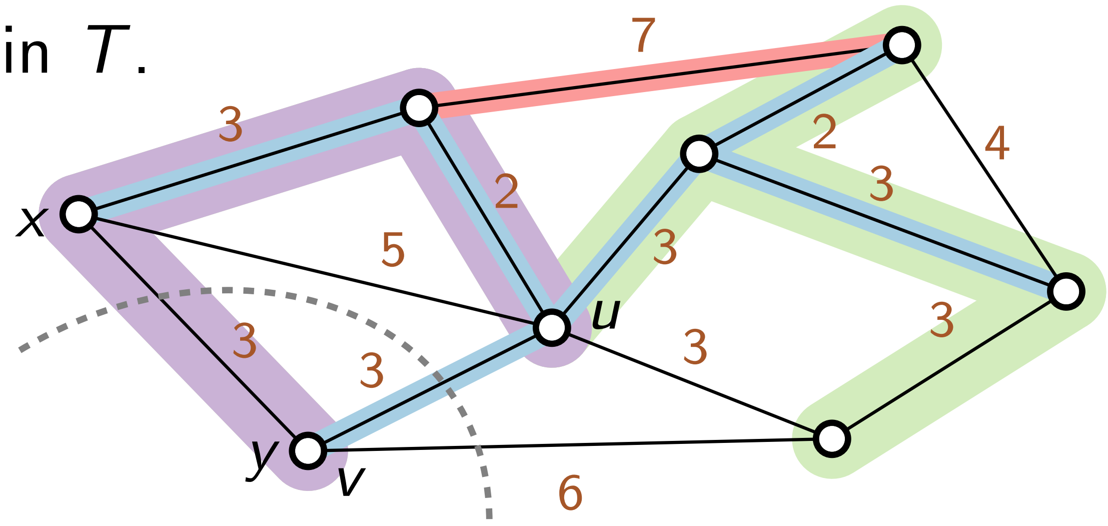
Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T) \Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T) \Rightarrow$ Es gibt Pfad p von u nach v in T.

$\Rightarrow p$ enthält Kante xy, die Schnitt kreuzt



Beweis der blauen Regel

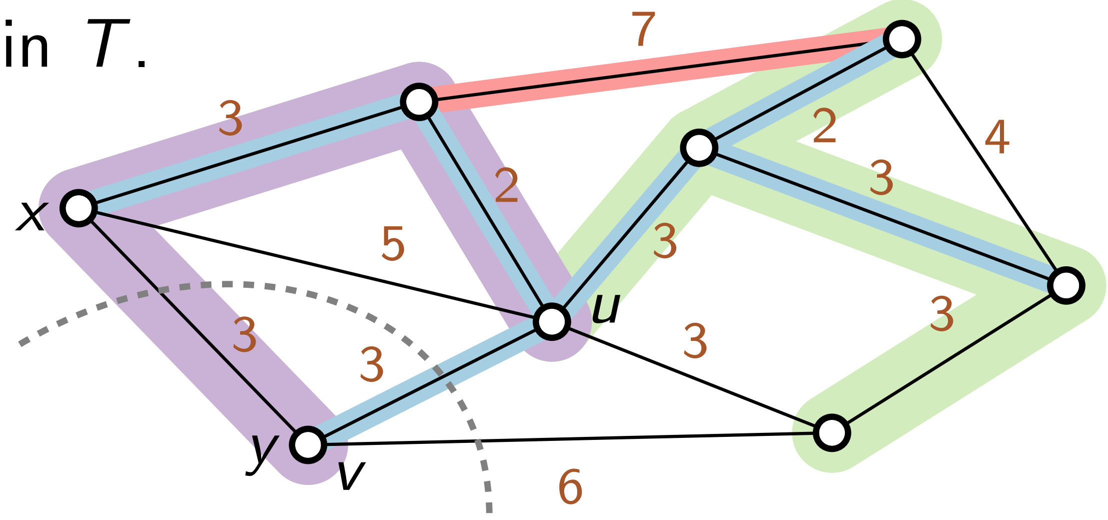
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.
$\Rightarrow$ p enthält Kante xy, die Schnitt kreuzt



Beweis der blauen Regel

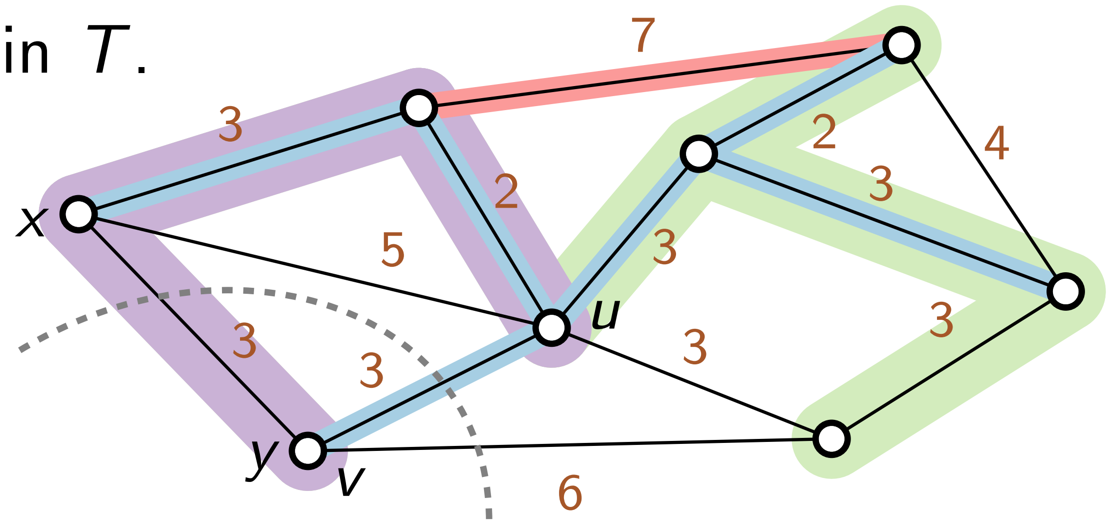
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.
$\Rightarrow$ p enthält Kante xy, die Schnitt kreuzt
keine blaue Kante kreuzt $\Rightarrow$ Kante xy ist ungefärbt.



Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

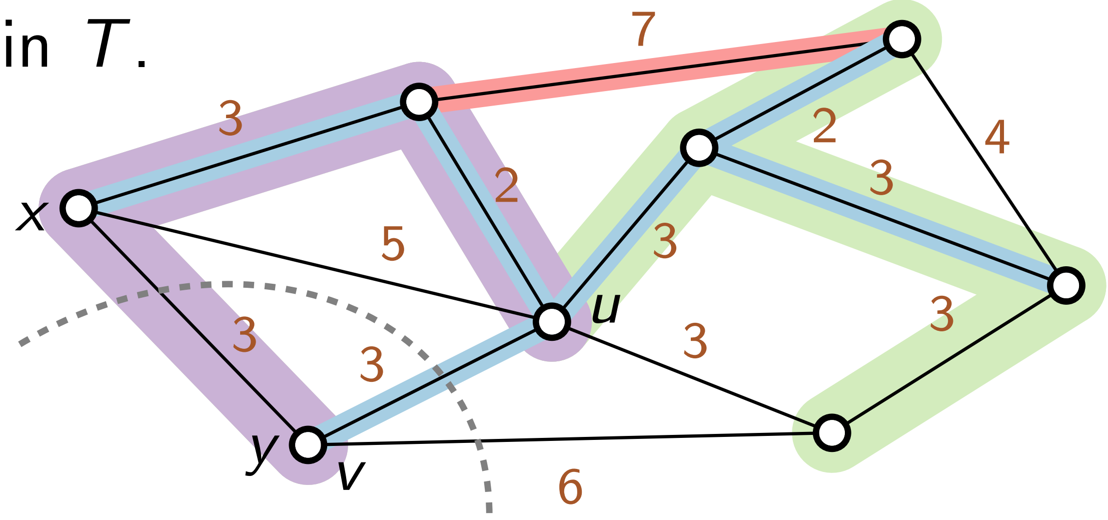
Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.
$\Rightarrow$ p enthält Kante xy, die Schnitt kreuzt

keine blaue Kante kreuzt $\Rightarrow$ Kante xy ist ungefärbt.



Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

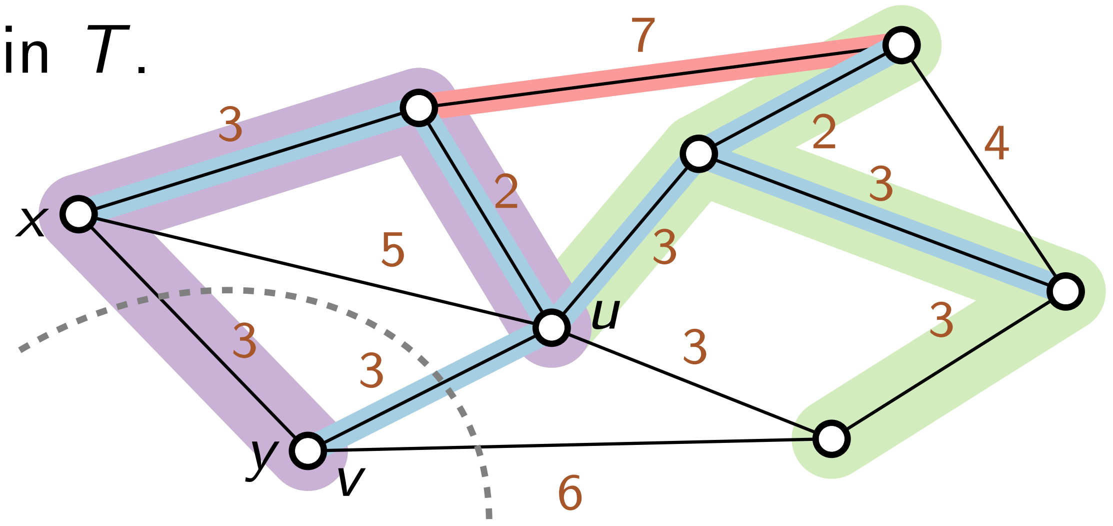
Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.

$\Rightarrow$ p enthält Kante xy, die Schnitt kreuzt

keine blaue Kante kreuzt $\Rightarrow$ Kante xy ist ungefärbt.

leichte Kante $\Rightarrow$ $w(xy) \geq w(uv)$



Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

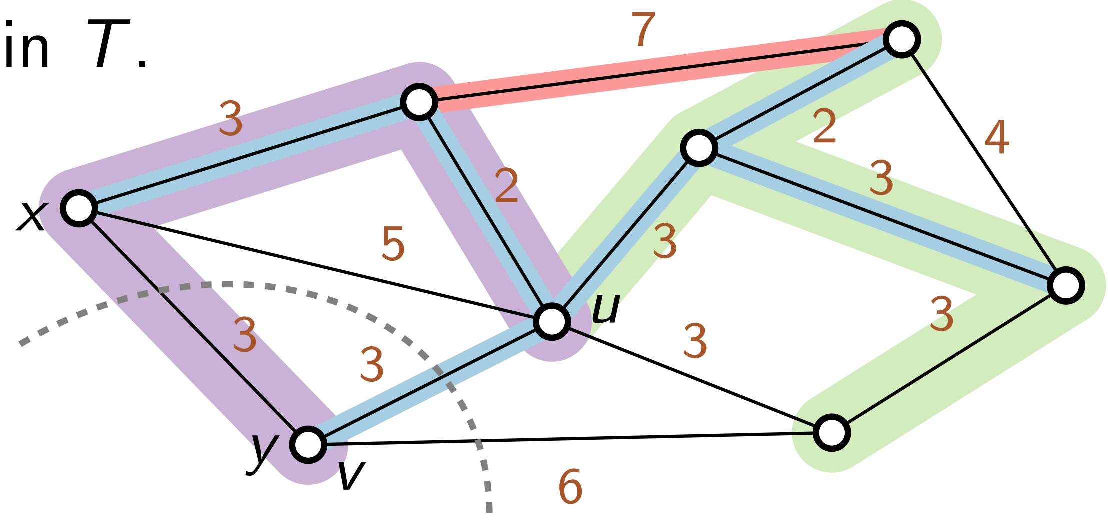
1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.

$\Rightarrow$ p enthält Kante xy, die Schnitt kreuzt

keine blaue Kante kreuzt $\Rightarrow$ Kante xy ist ungefärbt.

leichte Kante $\Rightarrow$ $w(xy) \geq w(uv)$

Wähle $E' = E(T) \cup \{uv\} \setminus \{xy\}$.



Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

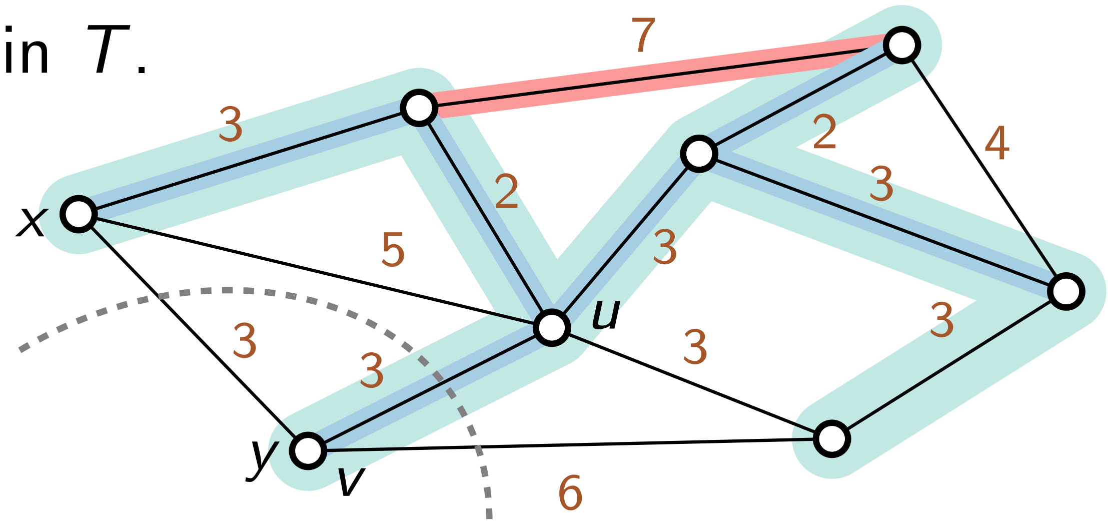
1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.

$\Rightarrow$ p enthält Kante xy, die Schnitt kreuzt

keine blaue Kante kreuzt $\Rightarrow$ Kante xy ist ungefärbt.

leichte Kante $\Rightarrow$ $w(xy) \geq w(uv)$

Wähle $E' = E(T) \cup \{uv\} \setminus \{xy\}$.



Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.

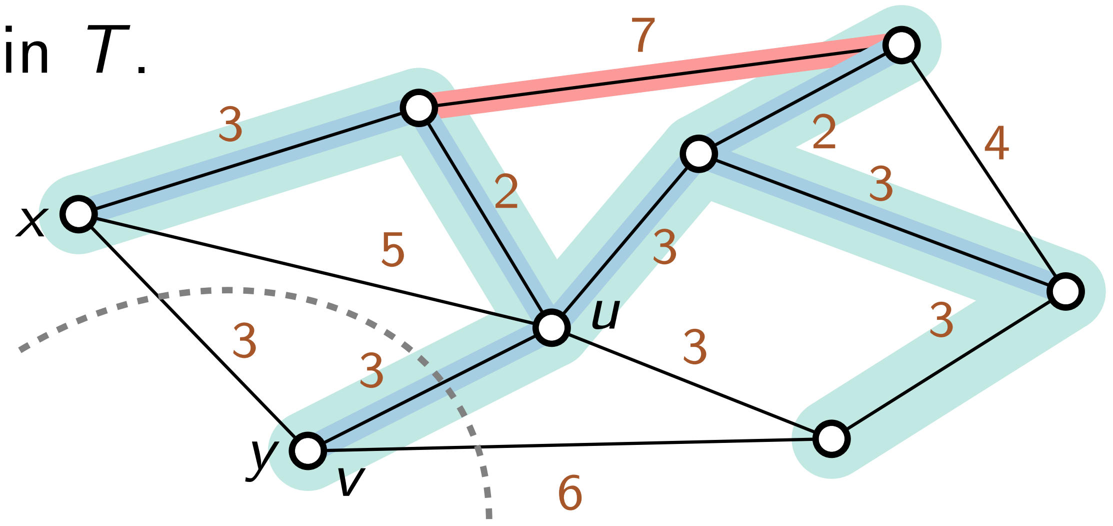
$\Rightarrow$ p enthält Kante xy, die Schnitt kreuzt

keine blaue Kante kreuzt $\Rightarrow$ Kante xy ist ungefärbt.

leichte Kante $\Rightarrow$ $w(xy) \geq w(uv)$

Wähle $E' = E(T) \cup \{uv\} \setminus \{xy\}$.

$\Rightarrow$ $T' = (V(T), E')$ ist MSB, der FI bezeugt.



Beweis der blauen Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten.
- T enthält keine rote Kante.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. Die blaue Regel hält die Farbinvariante aufrecht.

Beweis. Alle Kanten ungefärbt $\Rightarrow$ jeder MSB **bezeugt** FI.
Sei T minimaler Spannbaum, der FI bezeugt.
Sei $uv \in E$ von blauer Regel ausgewählte Kante.

1. Fall: $uv \in E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \notin E(T)$ $\Rightarrow$ Es gibt Pfad p von u nach v in T.

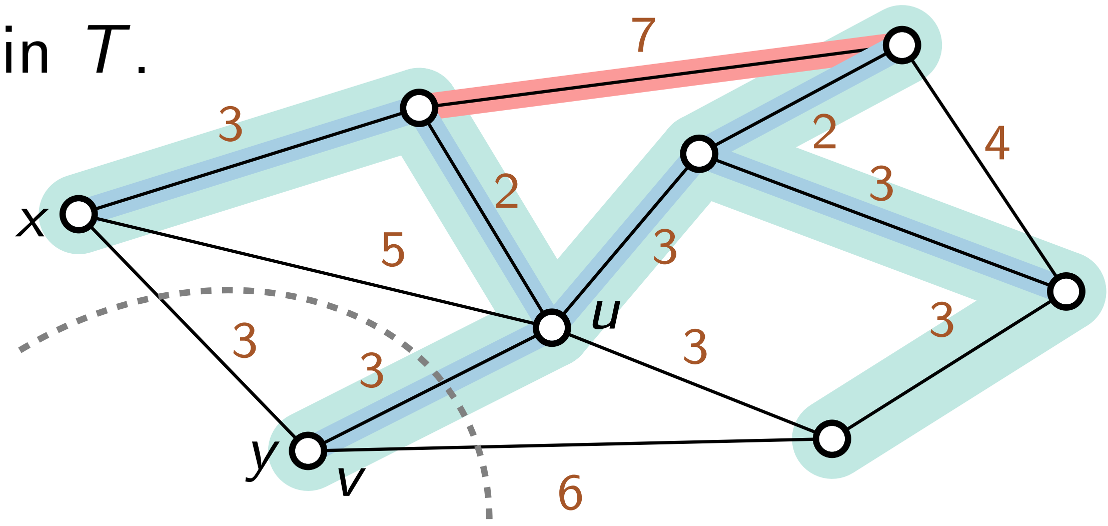
$\Rightarrow$ p enthält Kante xy, die Schnitt kreuzt

keine blaue Kante kreuzt $\Rightarrow$ Kante xy ist ungefärbt.

leichte Kante $\Rightarrow$ $w(xy) \geq w(uv)$

Wähle $E' = E(T) \cup \{uv\} \setminus \{xy\}$.

$\Rightarrow$ $T' = (V(T), E')$ ist MSB, der FI bezeugt. $\square$



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:

- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:

Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

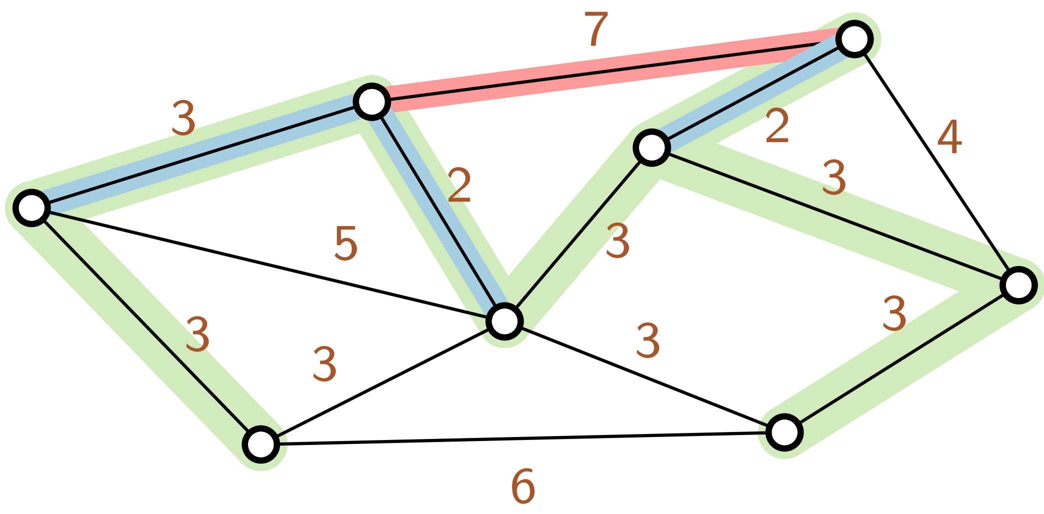
Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.



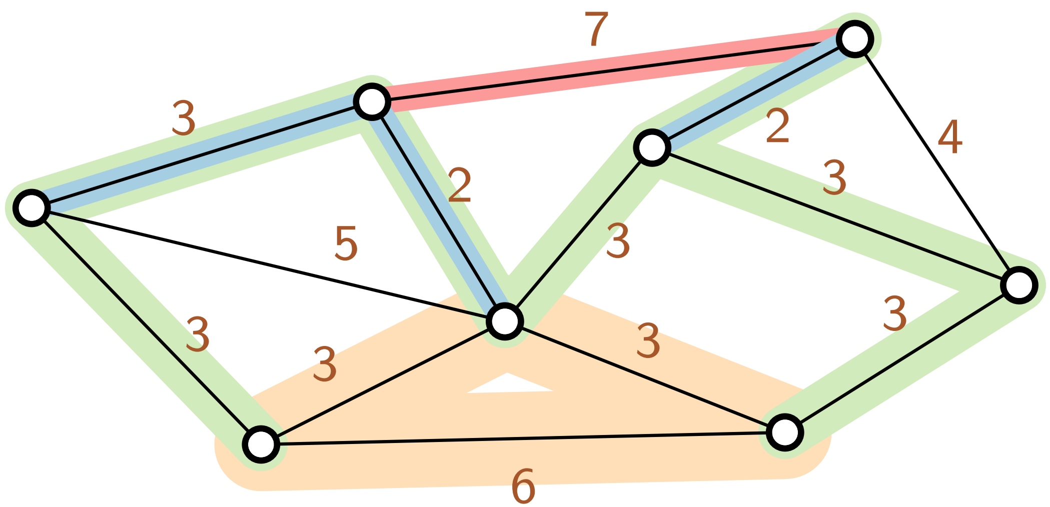
Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.



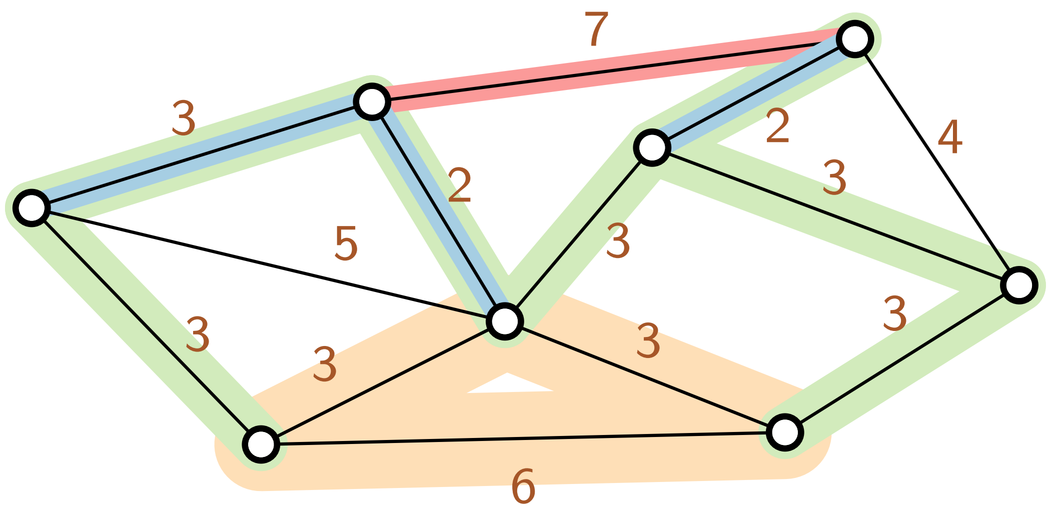
Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.



Beweis der roten Regel

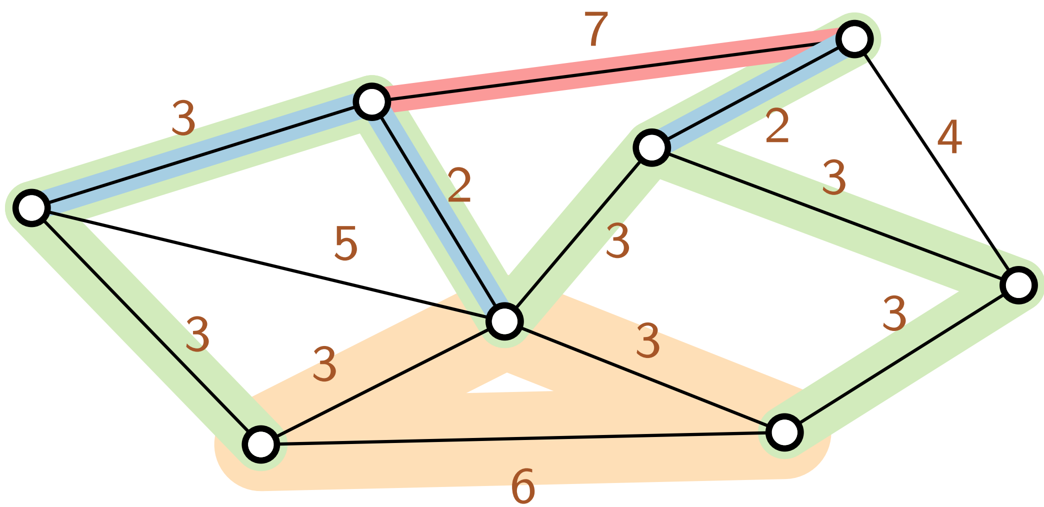
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$



Beweis der roten Regel

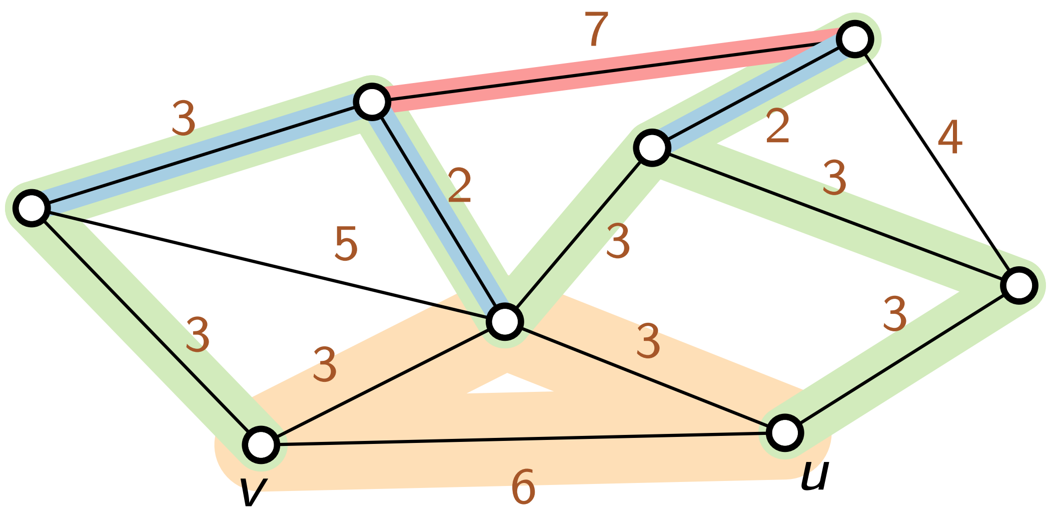
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$



Beweis der roten Regel

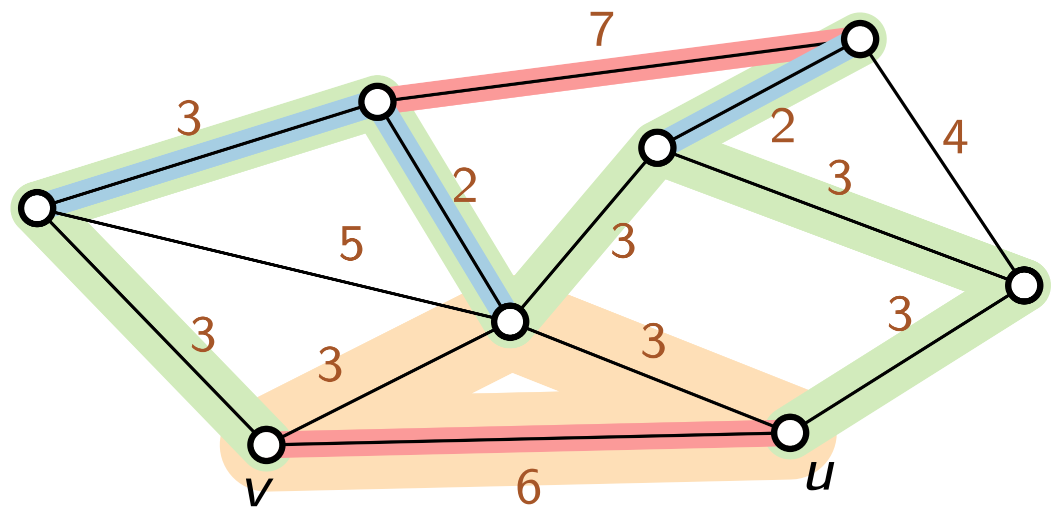
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$



Beweis der roten Regel

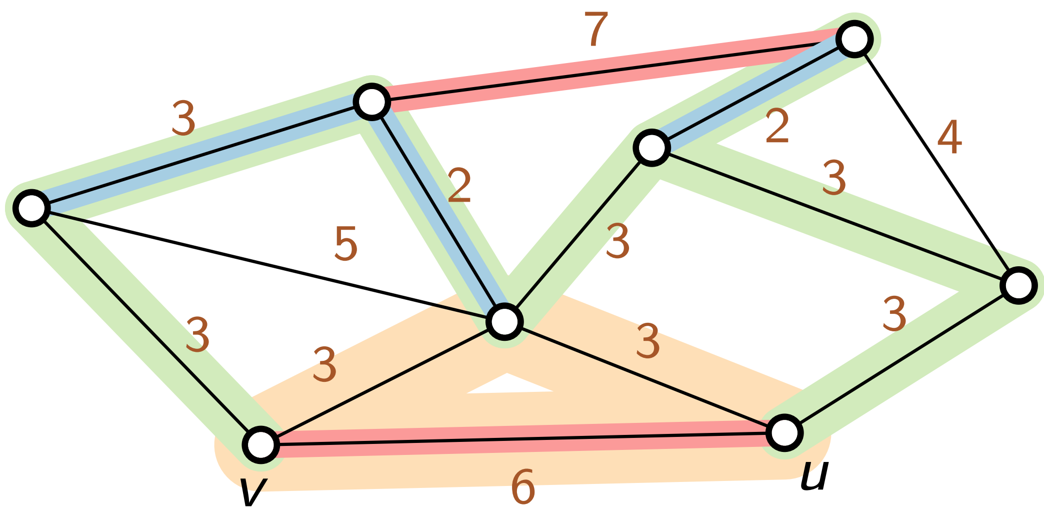
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.



Beweis der roten Regel

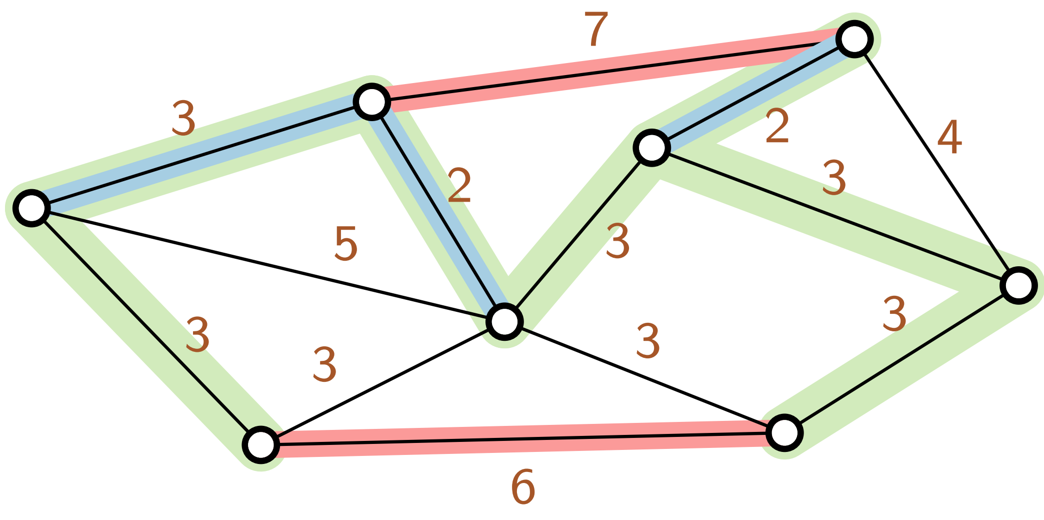
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$



Beweis der roten Regel

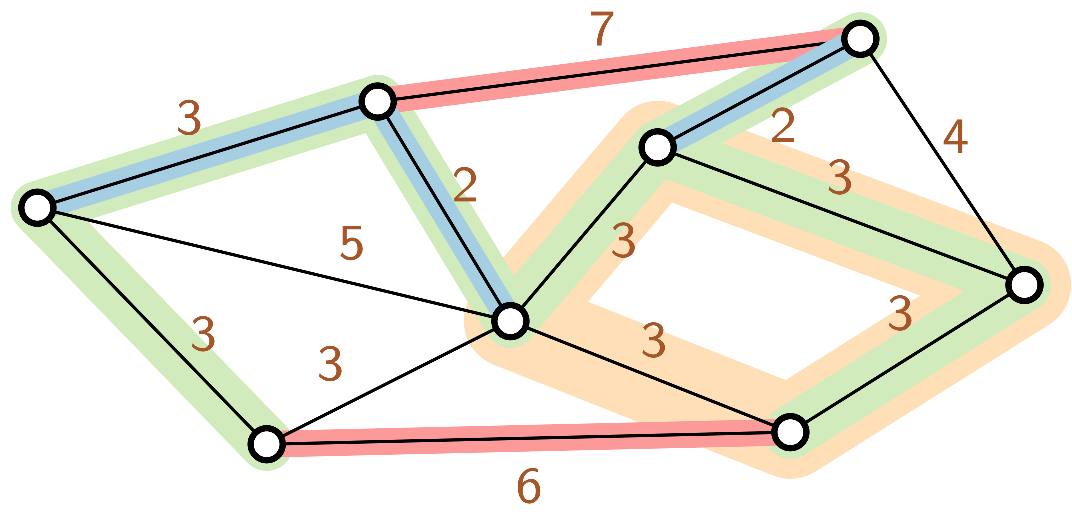
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T) \Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$



Beweis der roten Regel

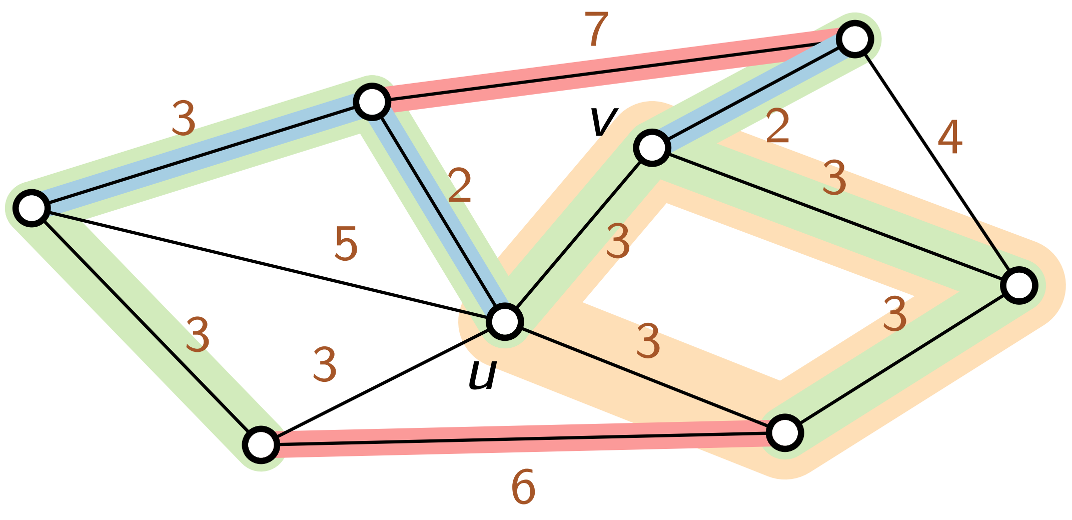
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$



Beweis der roten Regel

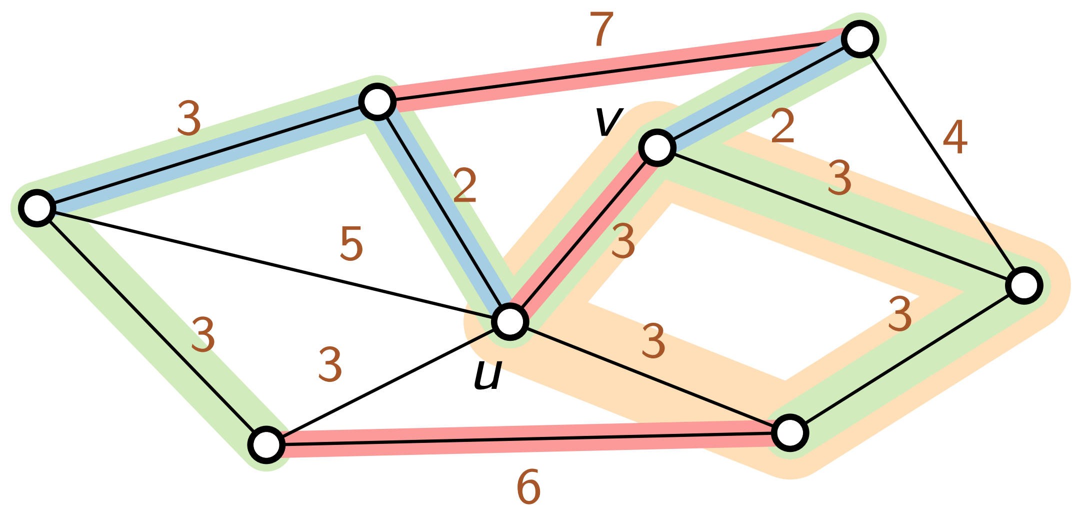
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T) \Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$



Beweis der roten Regel

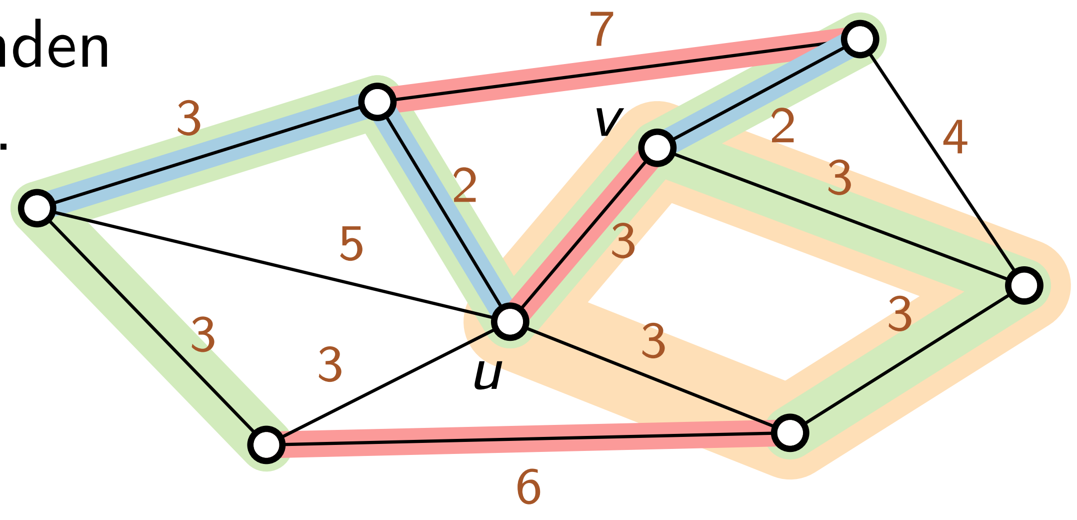
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.



Beweis der roten Regel

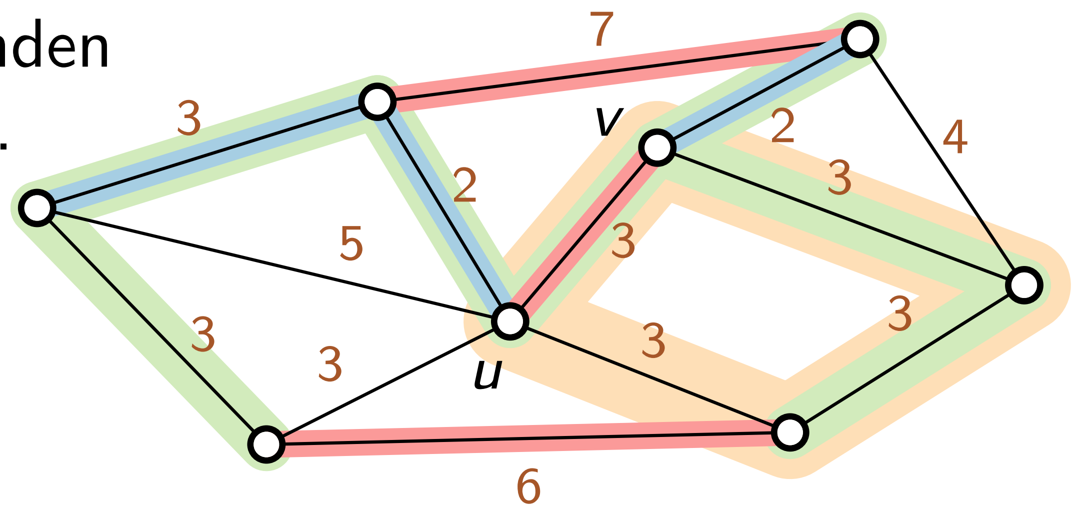
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.
Sei $u \in T_1$, $v \in T_2$.



Beweis der roten Regel

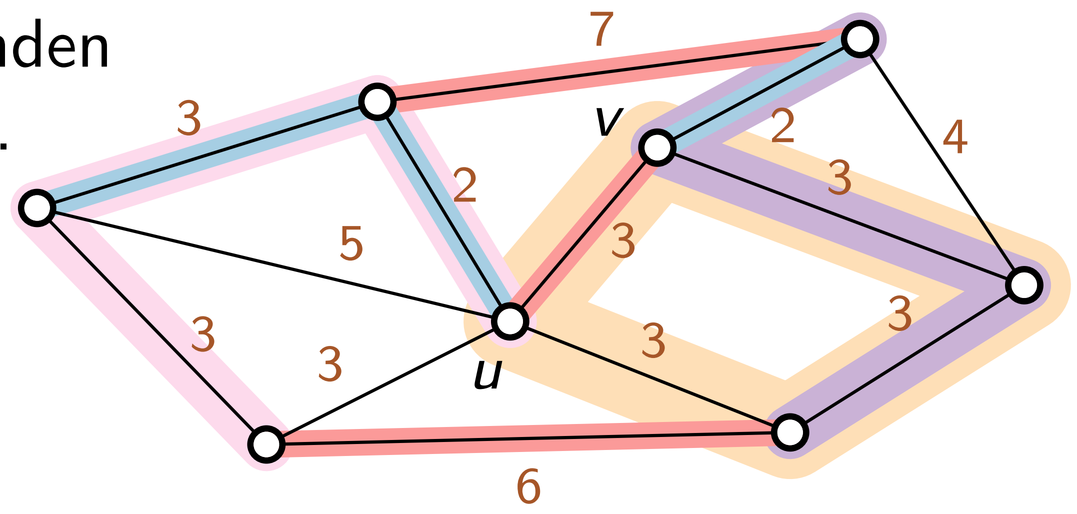
Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T) \Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T) \Rightarrow E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.
Sei $u \in T_1$, $v \in T_2$.



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

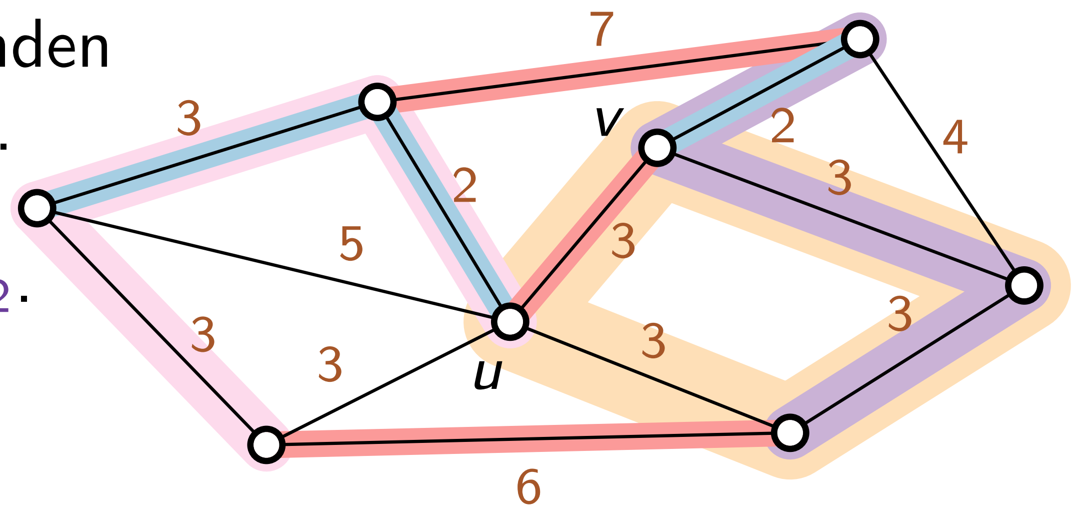
Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.
Sei $u \in T_1$, $v \in T_2$.

$\Rightarrow$ Es gibt Kante $xy \neq uv$ in K mit $x \in T_1, y \in T_2$.



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

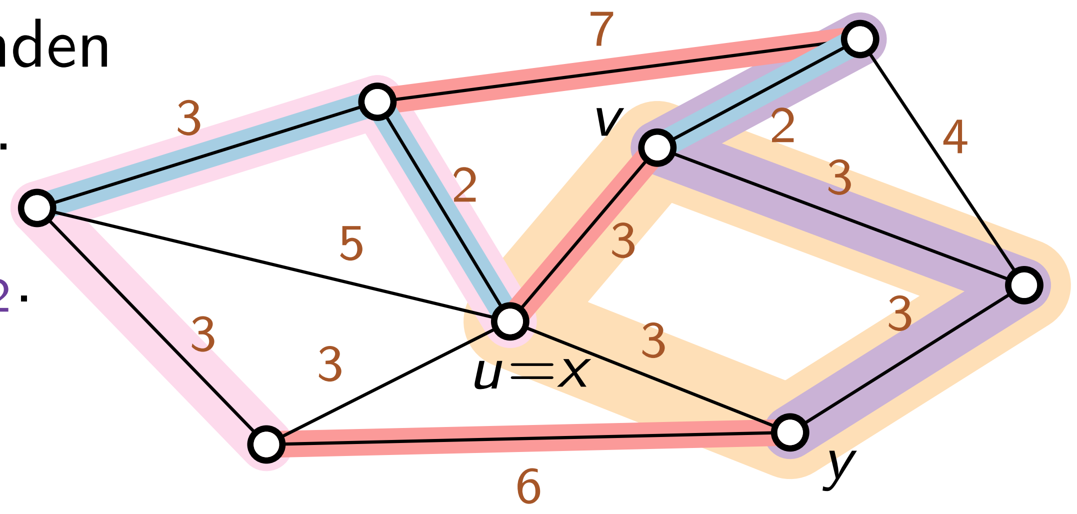
Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2. Sei $u \in T_1$, $v \in T_2$.

$\Rightarrow$ Es gibt Kante $xy \neq uv$ in K mit $x \in T_1, y \in T_2$.



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

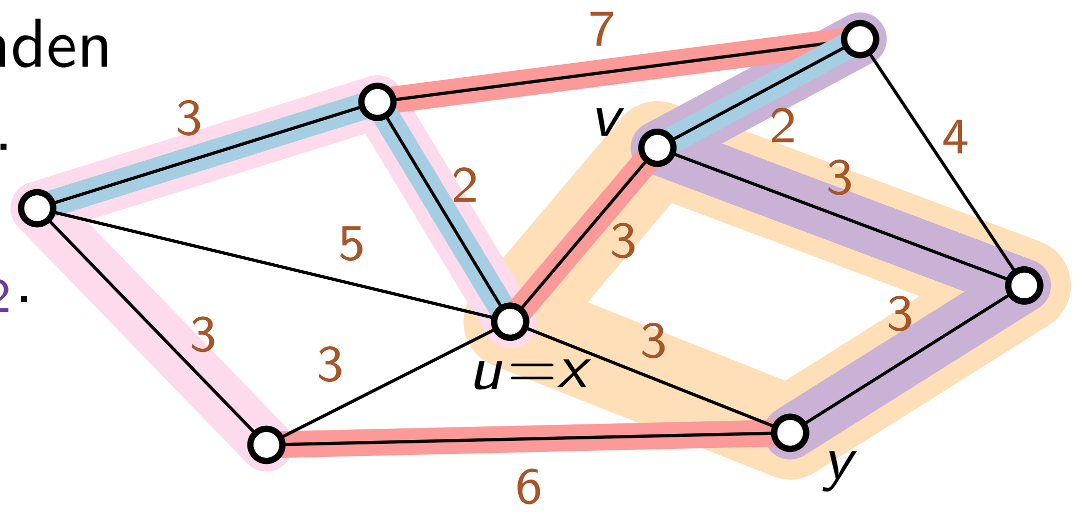
Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.
Sei $u \in T_1$, $v \in T_2$.

$xy \notin E(T)$

$\Rightarrow$ Es gibt Kante $xy \neq uv$ in K mit $x \in T_1, y \in T_2$.



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

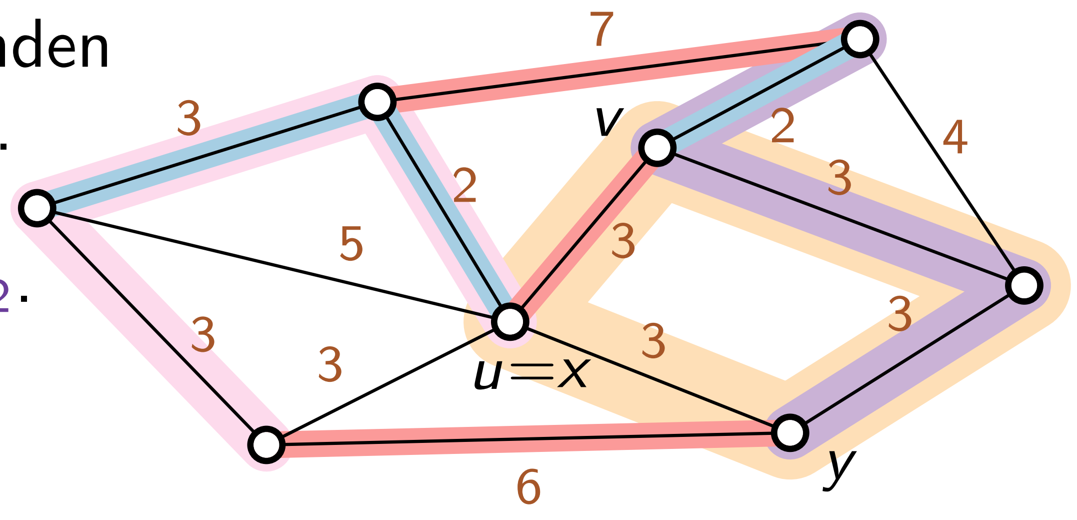
Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.
Sei $u \in T_1$, $v \in T_2$.

$xy \notin E(T)$

$\Rightarrow$ Es gibt Kante $xy \neq uv$ in K mit $x \in T_1, y \in T_2$.



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

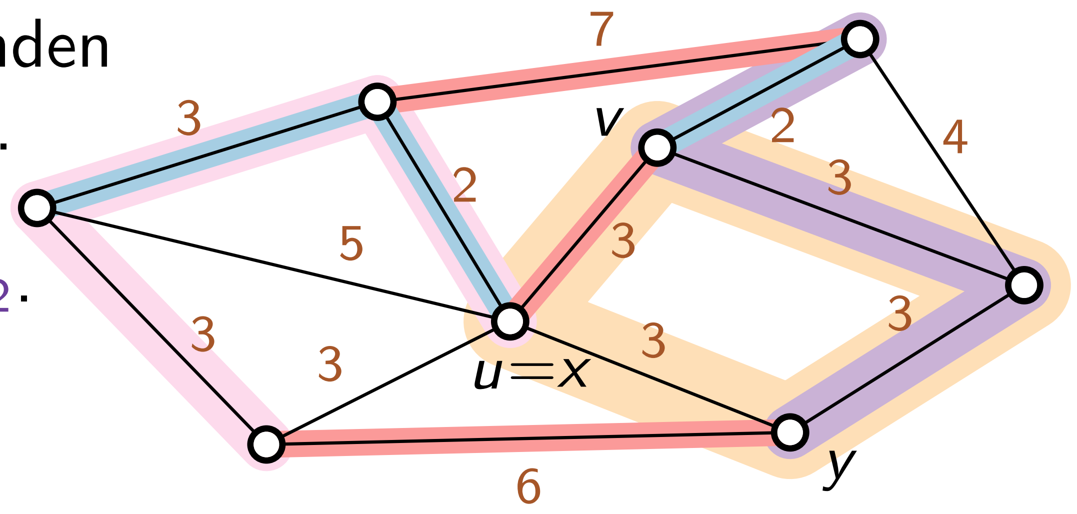
Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.
Sei $u \in T_1$, $v \in T_2$.

$xy \notin E(T)$

$\Rightarrow$ Es gibt Kante $xy \neq uv$ in K mit $x \in T_1, y \in T_2$.

größte ungefärbte Kante $\Rightarrow w(xy) \leq w(uv)$



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

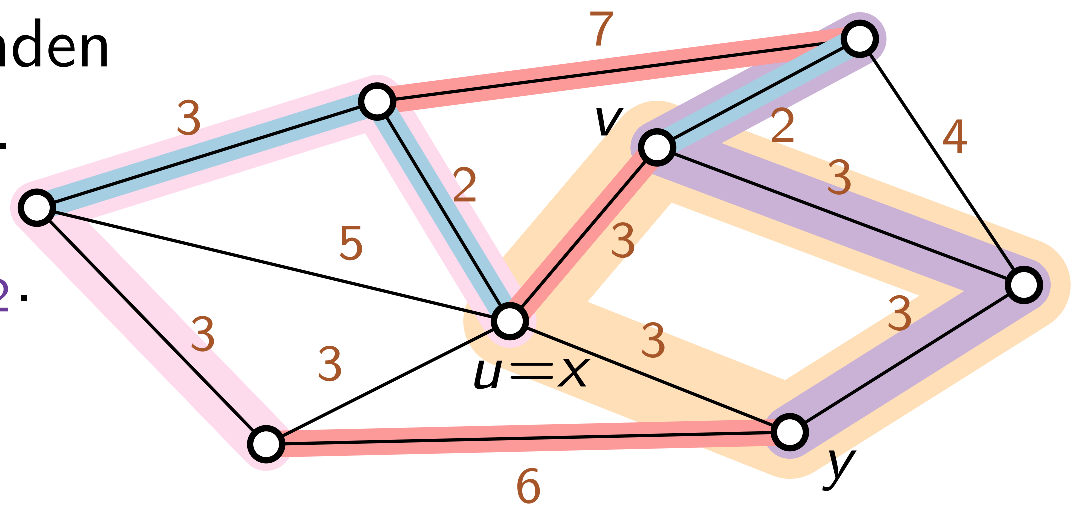
Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.
Sei $u \in T_1$, $v \in T_2$.

$xy \notin E(T)$

$\Rightarrow$ Es gibt Kante $xy \neq uv$ in K mit $x \in T_1, y \in T_2$.

größte ungefärbte Kante $\Rightarrow$ $w(xy) \leq w(uv)$



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

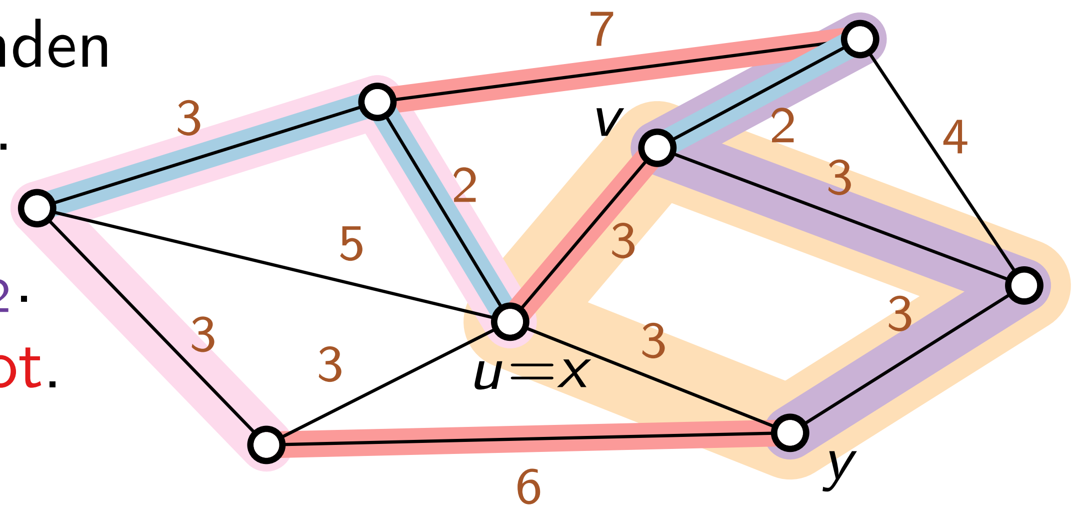
Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.
Sei $u \in T_1$, $v \in T_2$.

$xy \notin E(T)$

$\Rightarrow$ Es gibt Kante $xy \neq uv$ in K mit $x \in T_1, y \in T_2$.

größte ungefärbte Kante $\Rightarrow w(xy) \leq w(uv)$ ohne rote Kante $\Rightarrow xy$ nicht rot.



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

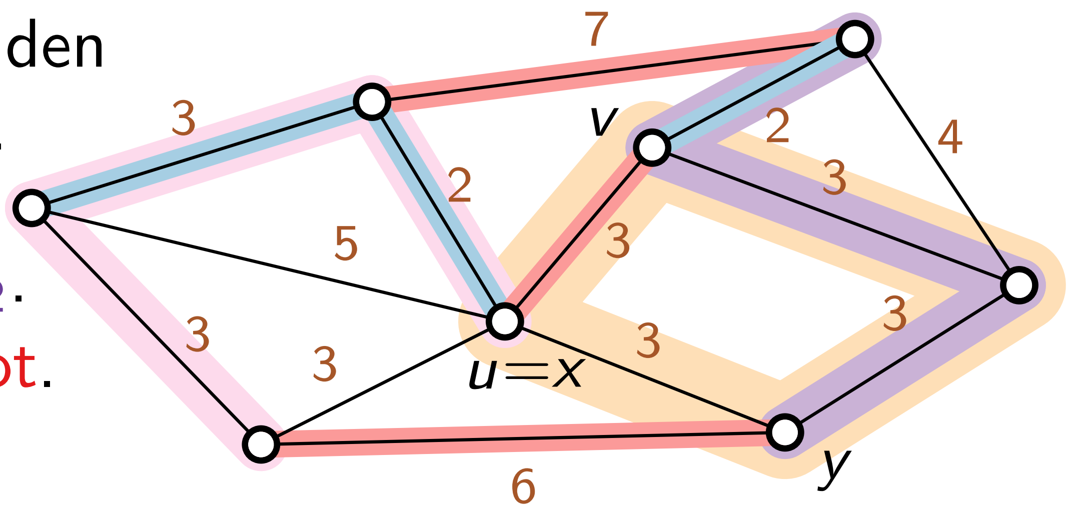
1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.
Sei $u \in T_1$, $v \in T_2$.

$xy \notin E(T)$

$\Rightarrow$ Es gibt Kante $xy \neq uv$ in K mit $x \in T_1, y \in T_2$.

größte ungefärbte Kante $\Rightarrow w(xy) \leq w(uv)$ ohne rote Kante $\Rightarrow xy$ nicht rot.

Wähle $E' = T(E) \cup \{xy\} \setminus \{uv\}$.



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

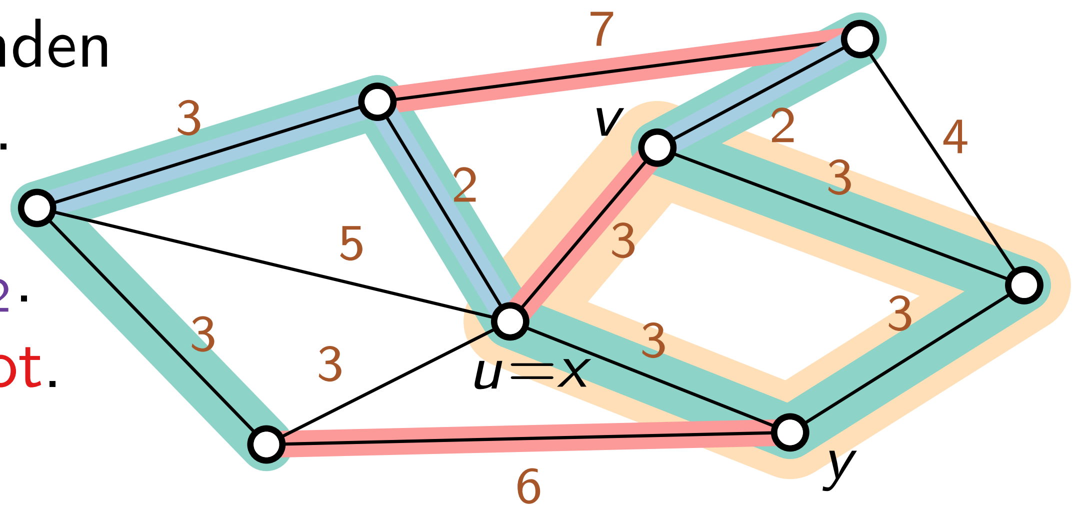
1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.
Sei $u \in T_1$, $v \in T_2$.

$xy \notin E(T)$

$\Rightarrow$ Es gibt Kante $xy \neq uv$ in K mit $x \in T_1, y \in T_2$.

größte ungefärbte Kante $\Rightarrow w(xy) \leq w(uv)$ ohne rote Kante $\Rightarrow xy$ nicht rot.

Wähle $E' = T(E) \cup \{xy\} \setminus \{uv\}$.



Beweis der roten Regel

Farbinvariante (FI): Es gibt einen MSB T:
- T enthält alle blauen Kanten
- T enthält keine rote Kante

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Lemma. Die rote Regel hält die Farbinvariante aufrecht.

Beweis. Sei T min. Spannbaum, der FI bezeugt.
Sei K von roter Regel ausgewählter Kreis.
Sei $uv \in E$ von roter Regel gefärbte Kante.

1. Fall: $uv \notin E(T)$ $\Rightarrow$ FI bleibt erhalten.
2. Fall: $uv \in E(T)$ $\Rightarrow$ $E(T) \setminus \{uv\}$ bildet aufspannenden Wald mit zwei Bäumen T_1, T_2.

$xy \notin E(T)$

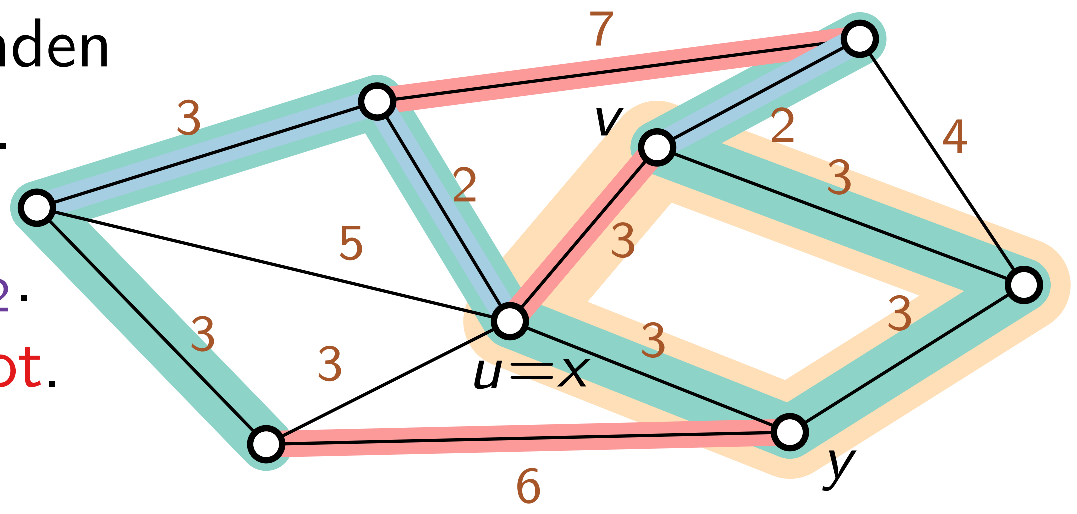
Sei $u \in T_1$, $v \in T_2$.

$\Rightarrow$ Es gibt Kante $xy \neq uv$ in K mit $x \in T_1, y \in T_2$.

größte ungefärbte Kante $\Rightarrow w(xy) \leq w(uv)$ ohne rote Kante $\Rightarrow xy$ nicht rot.

Wähle $E' = T(E) \cup \{xy\} \setminus \{uv\}$.

$\Rightarrow T' = (V(T), E')$ ist MSB, der FI bezeugt. $\square$



Alle Kanten werden gefärbt

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

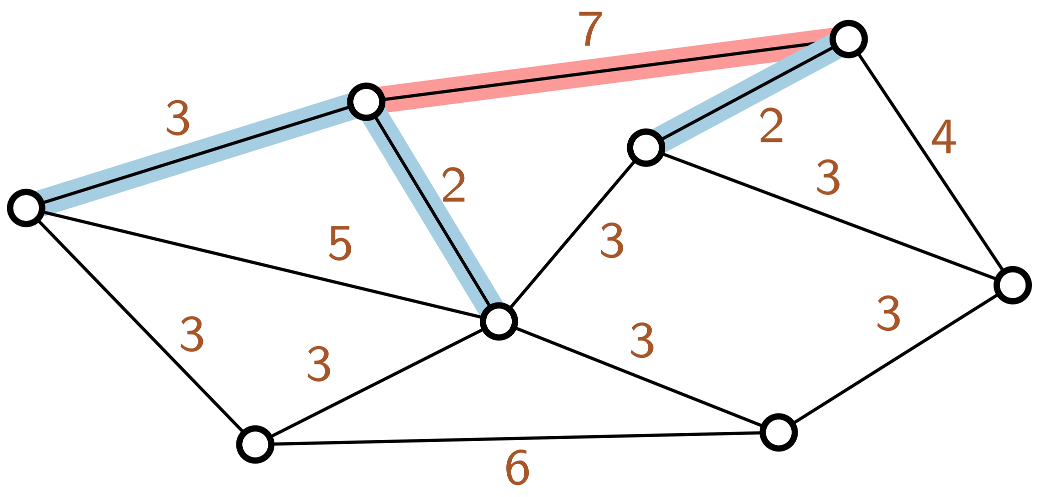
Alle Kanten werden gefärbt

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis.



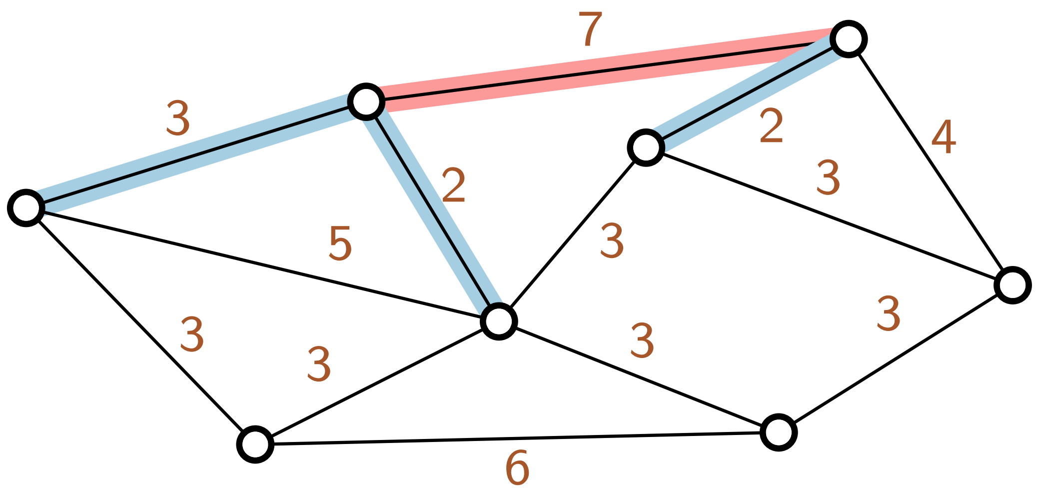
Alle Kanten werden gefärbt

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)



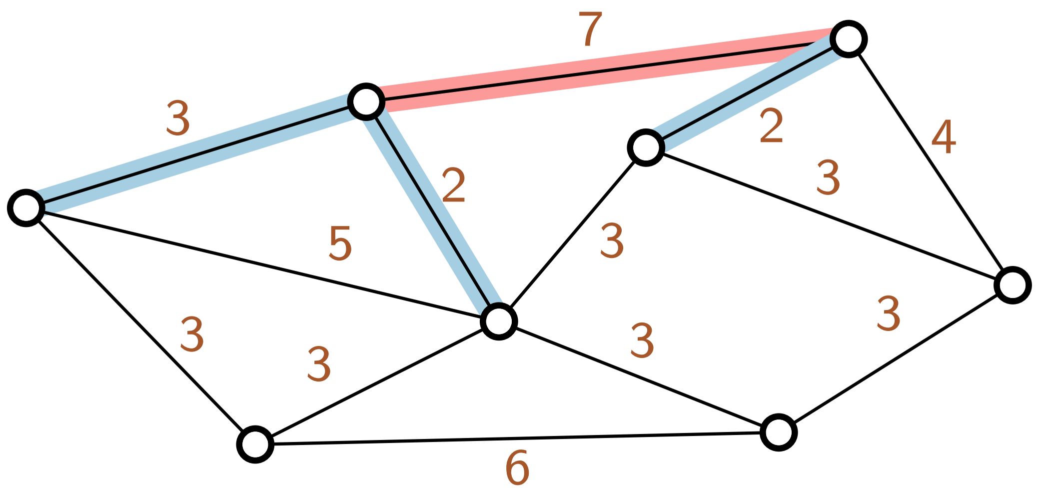
Alle Kanten werden gefärbt

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante.



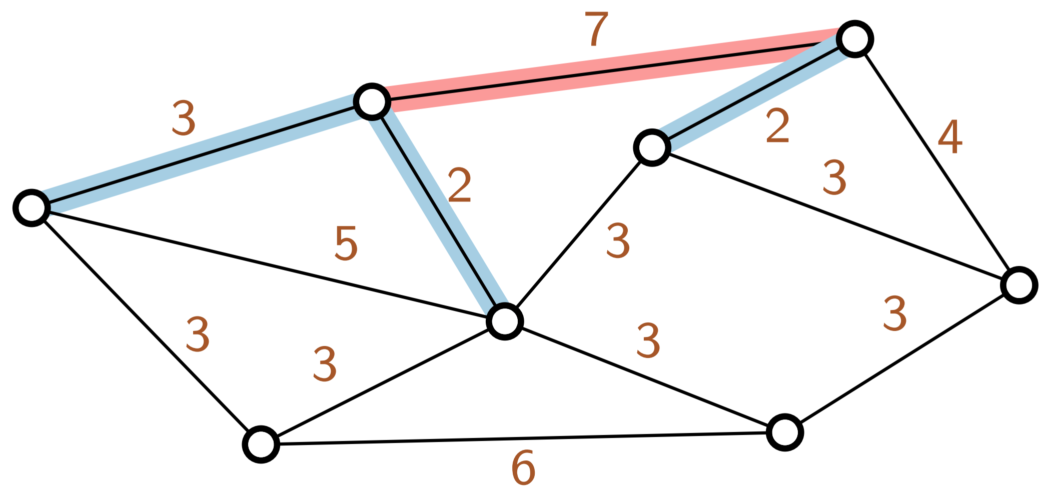
Alle Kanten werden gefärbt

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.



Alle Kanten werden gefärbt

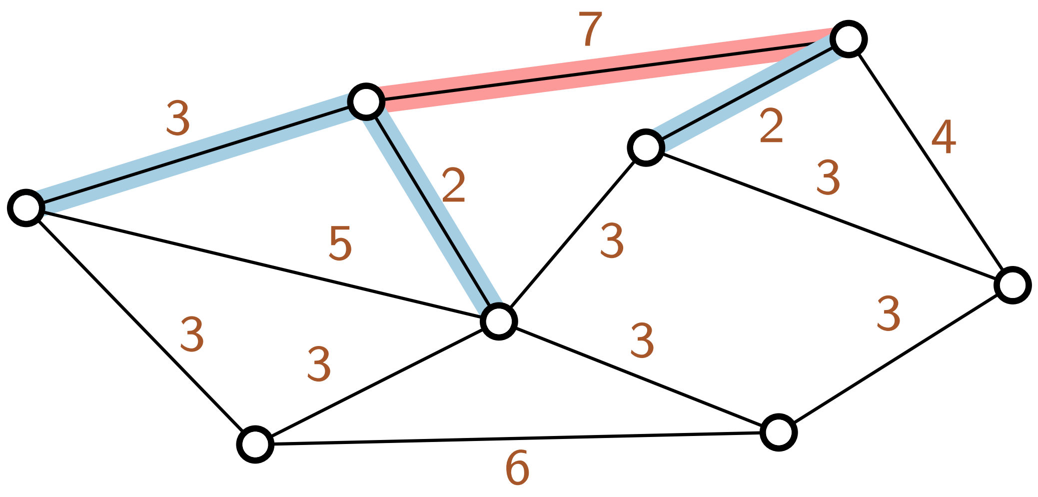
Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.



Alle Kanten werden gefärbt

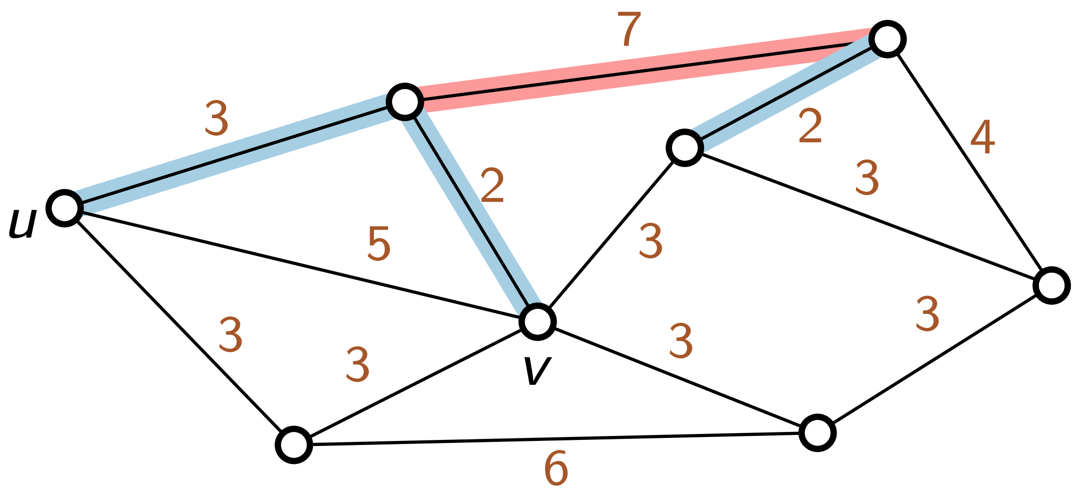
Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.



Alle Kanten werden gefärbt

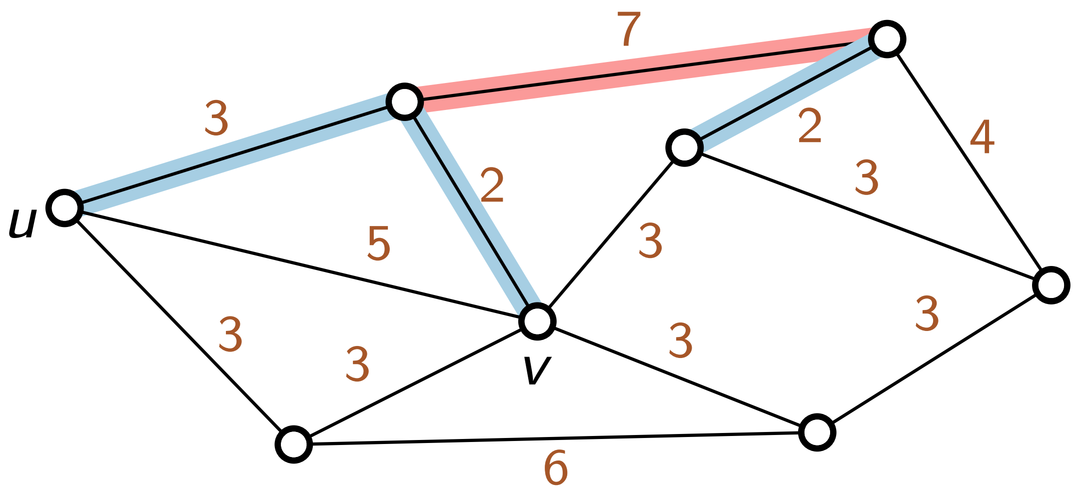
Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.
Wähle Kreis C: Pfad in B von v zu u + Kante uv



Alle Kanten werden gefärbt

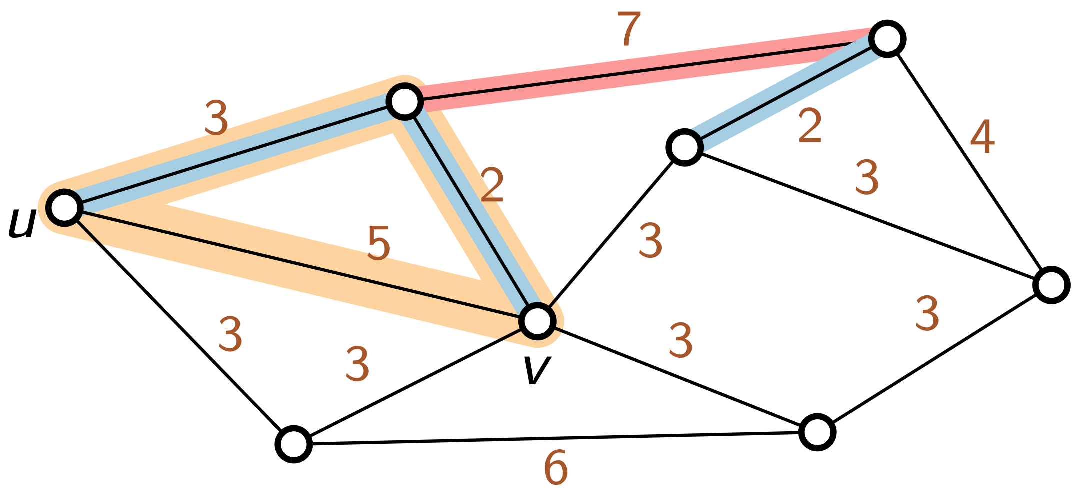
Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.
Wähle Kreis C: Pfad in B von v zu u + Kante uv



Alle Kanten werden gefärbt

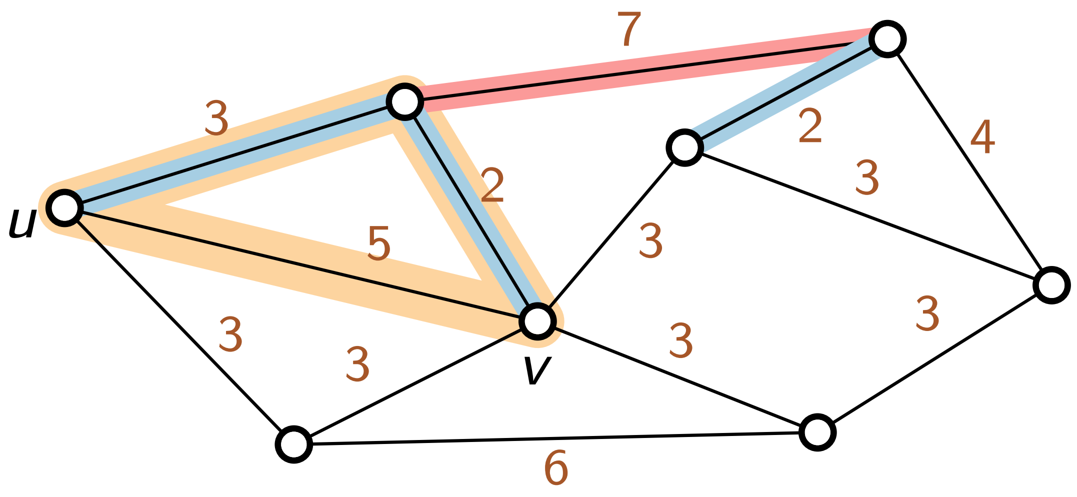
Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.
Wähle Kreis C: Pfad in B von v zu u + Kante uv
$\Rightarrow$ Kanten auf C alle blau bis auf uv.



Alle Kanten werden gefärbt

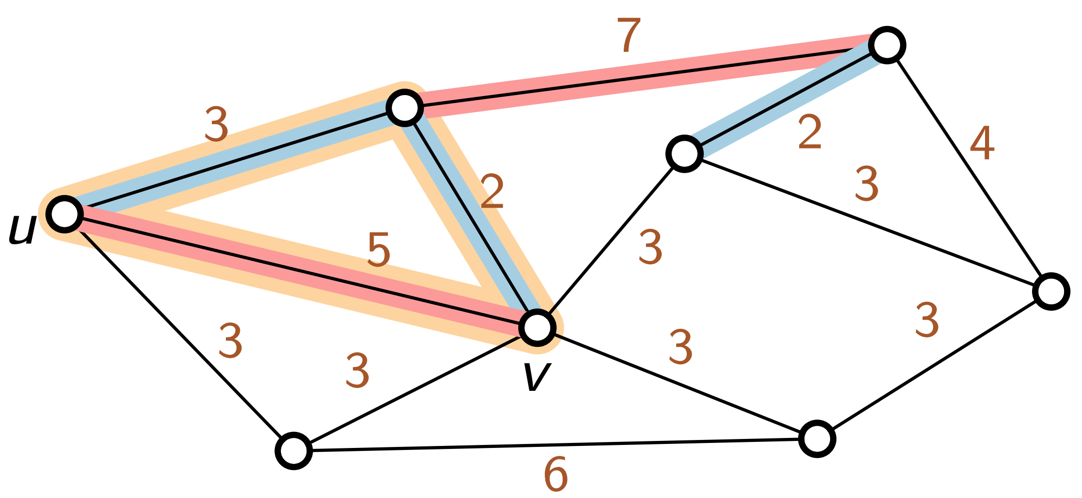
Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.
Wähle Kreis C: Pfad in B von v zu u + Kante uv
$\Rightarrow$ Kanten auf C alle blau bis auf uv.
$\Rightarrow$ Rote Regel anwendbar auf uv.



Alle Kanten werden gefärbt

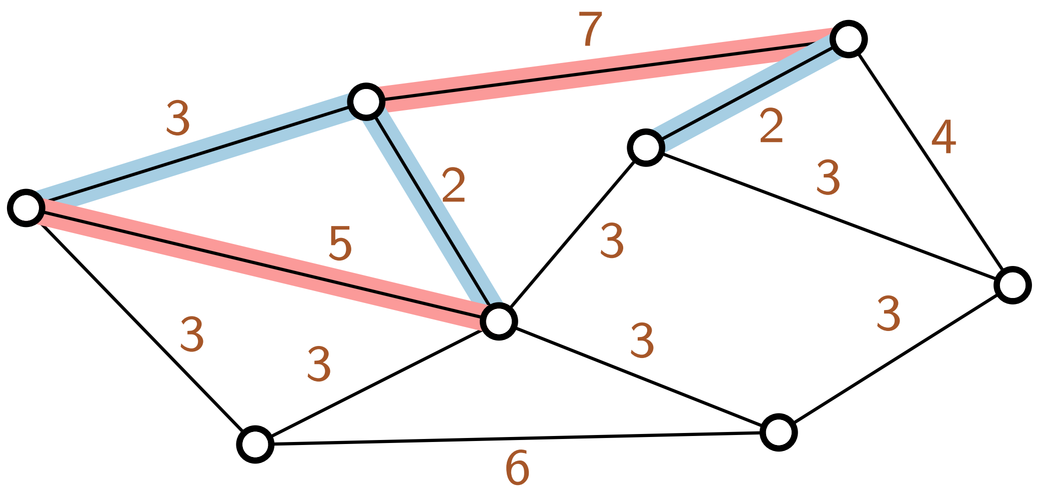
Rote Regel:	Blaue Regel:
Wähle Kreis ohne rote Kante. Färbe größte ungefärbte Kante auf Kreis rot.	Wähle Schnitt, den keine blaue Kante kreuzt. Färbe leichte Kante blau.

Lemma. GREEDYSPANNBAUM färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.
   Wähle Kreis C: Pfad in B von v zu u + Kante uv
   $\Rightarrow$ Kanten auf C alle blau bis auf uv.
   $\Rightarrow$ Rote Regel anwendbar auf uv.

2. Fall: uv verbindet *unterschiedliche* Bäume aus B.



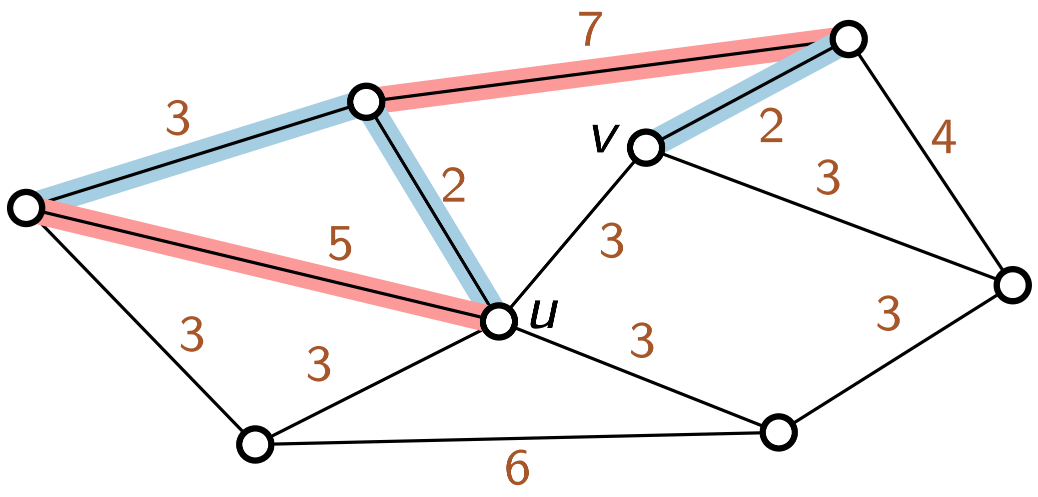
Alle Kanten werden gefärbt

Rote Regel:	Blaue Regel:
Wähle Kreis ohne rote Kante. Färbe größte ungefärbte Kante auf Kreis rot.	Wähle Schnitt, den keine blaue Kante kreuzt. Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.
   Wähle Kreis C: Pfad in B von v zu u + Kante uv
   $\Rightarrow$ Kanten auf C alle blau bis auf uv.
   $\Rightarrow$ Rote Regel anwendbar auf uv.
2. Fall: uv verbindet *unterschiedliche* Bäume aus B.



Alle Kanten werden gefärbt

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

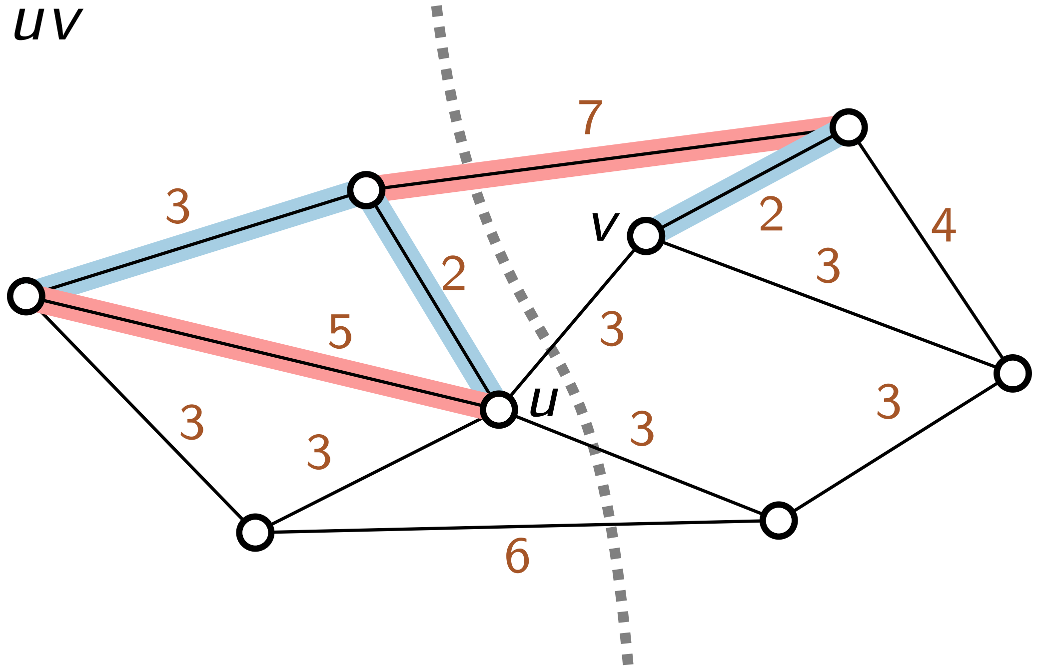
Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.
   Wähle Kreis C: Pfad in B von v zu u + Kante uv
   $\Rightarrow$ Kanten auf C alle blau bis auf uv.
   $\Rightarrow$ Rote Regel anwendbar auf uv.

2. Fall: uv verbindet *unterschiedliche* Bäume aus B.
   $\Rightarrow$ Es gibt Schnitt ohne blaue Kanten



Alle Kanten werden gefärbt

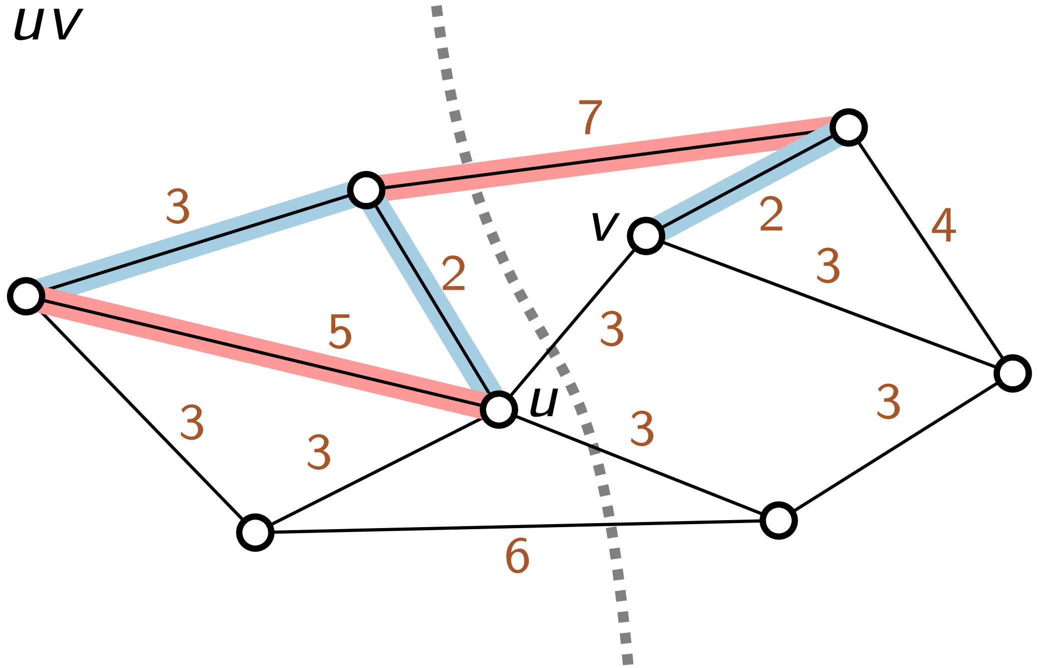
Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.
   Wähle Kreis C: Pfad in B von v zu u + Kante uv
   $\Rightarrow$ Kanten auf C alle blau bis auf uv.
   $\Rightarrow$ Rote Regel anwendbar auf uv.
2. Fall: uv verbindet *unterschiedliche* Bäume aus B.
   $\Rightarrow$ Es gibt Schnitt ohne blaue Kanten
   $\Rightarrow$ Blaue Regel anwendbar auf *eine* Kante, die den Schnitt kreuzt.



Alle Kanten werden gefärbt

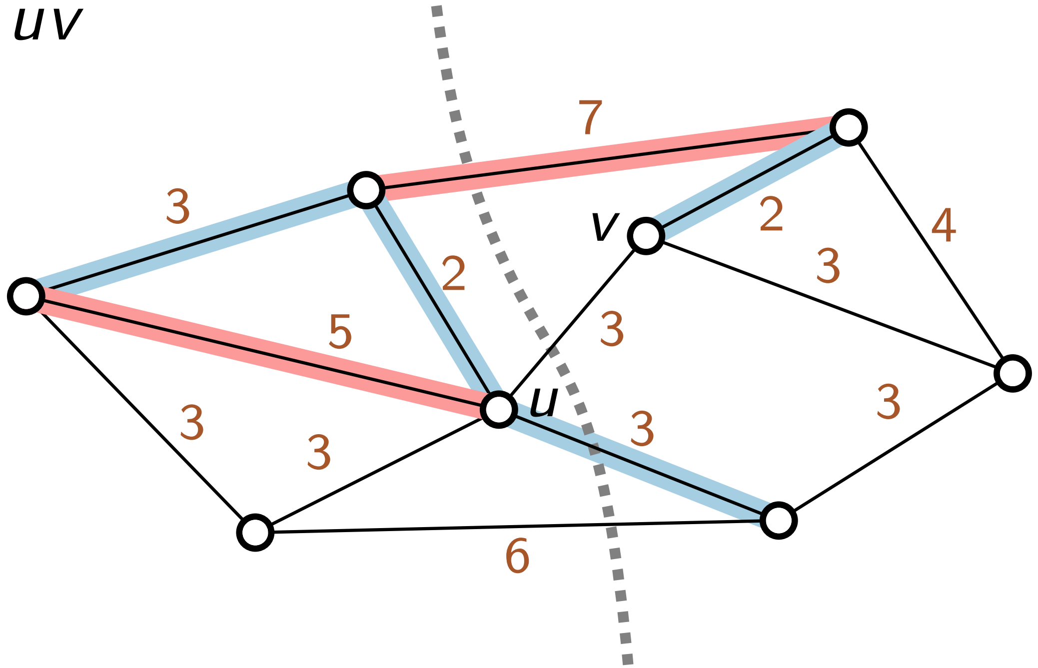
Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GREEDYSPANNBAUM färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.
   Wähle Kreis C: Pfad in B von v zu u + Kante uv
   $\Rightarrow$ Kanten auf C alle blau bis auf uv.
   $\Rightarrow$ Rote Regel anwendbar auf uv.
2. Fall: uv verbindet *unterschiedliche* Bäume aus B.
   $\Rightarrow$ Es gibt Schnitt ohne blaue Kanten
   $\Rightarrow$ Blaue Regel anwendbar auf *eine* Kante, die den Schnitt kreuzt.



Alle Kanten werden gefärbt

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.

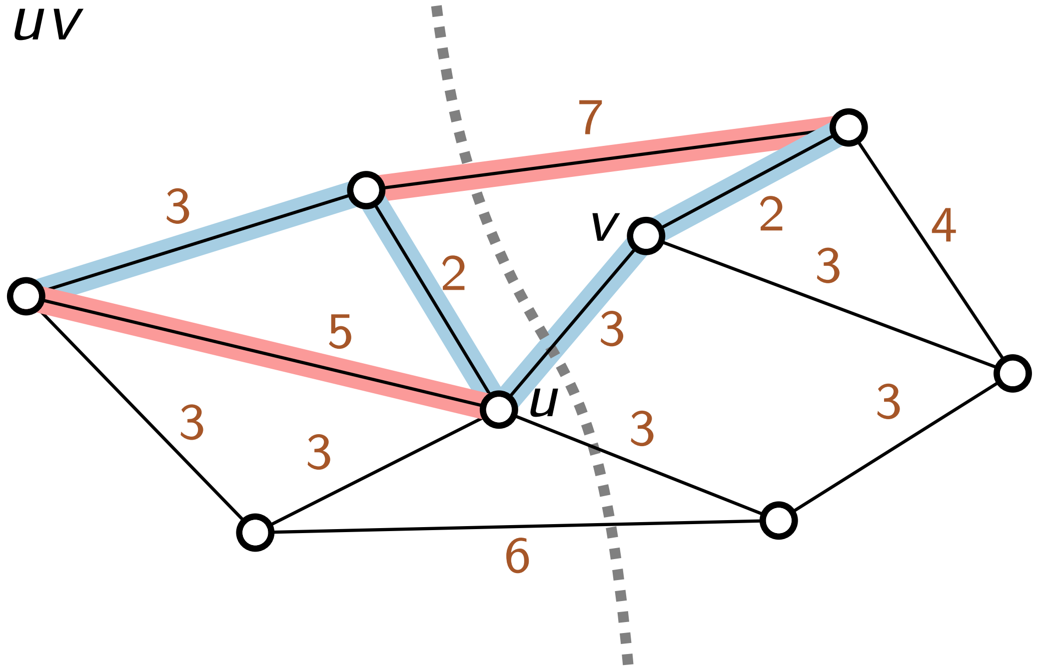
Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Lemma. GreedySpannbaum färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

1. Fall: uv verbindet Knoten *eines* Baumes aus B.
   Wähle Kreis C: Pfad in B von v zu u + Kante uv
   $\Rightarrow$ Kanten auf C alle blau bis auf uv.
   $\Rightarrow$ Rote Regel anwendbar auf uv.

2. Fall: uv verbindet *unterschiedliche* Bäume aus B.
   $\Rightarrow$ Es gibt Schnitt ohne blaue Kanten
   $\Rightarrow$ Blaue Regel anwendbar auf *eine* Kante, die den Schnitt kreuzt.



Alle Kanten werden gefärbt

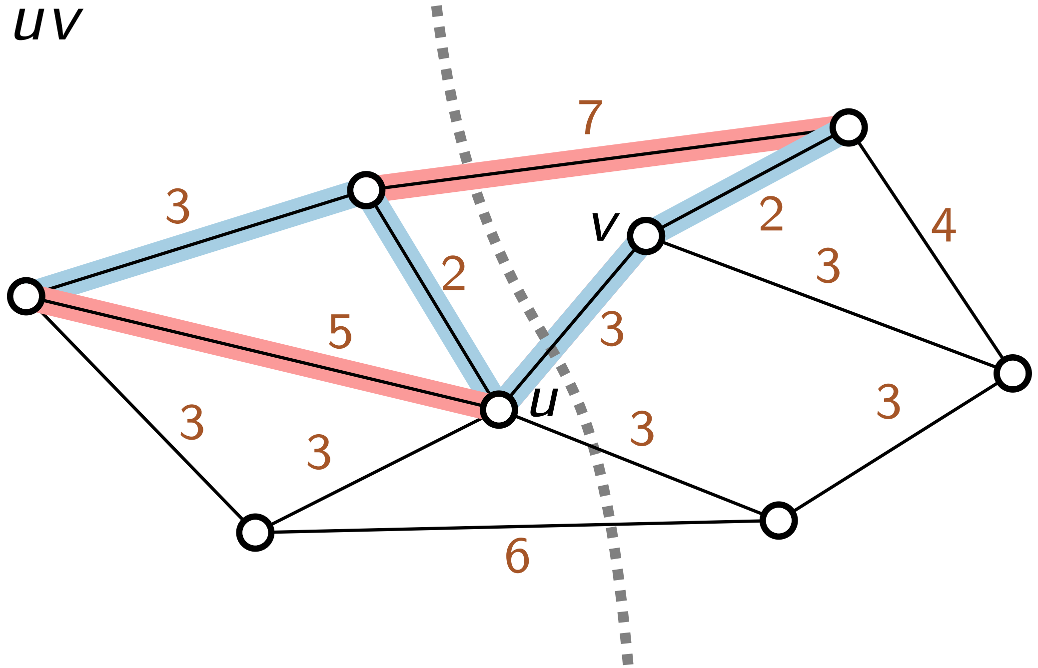
Rote Regel:	Blaue Regel:
Wähle Kreis ohne rote Kante. Färbe größte ungefärbte Kante auf Kreis rot.	Wähle Schnitt, den keine blaue Kante kreuzt. Färbe leichte Kante blau.

Lemma. GREEDYSPANNBAUM färbt alle Kanten.

Beweis. Blaue Kanten bilden Wald B (ggfs. isolierte Knoten)
Sei uv ungefärbte Kante. Zu zeigen: uv (oder eine andere Kante!) wird gefärbt.

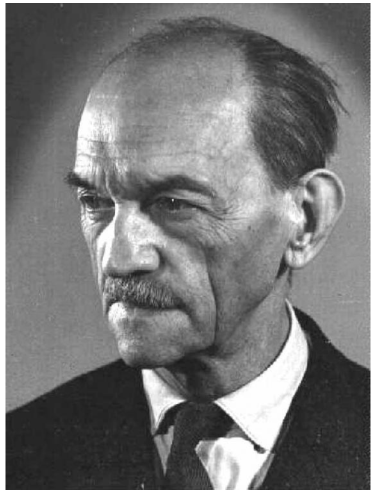
1. Fall: uv verbindet Knoten *eines* Baumes aus B.
   Wähle Kreis C: Pfad in B von v zu u + Kante uv
   $\Rightarrow$ Kanten auf C alle blau bis auf uv.
   $\Rightarrow$ Rote Regel anwendbar auf uv.

2. Fall: uv verbindet *unterschiedliche* Bäume aus B.
   $\Rightarrow$ Es gibt Schnitt ohne blaue Kanten
   $\Rightarrow$ Blaue Regel anwendbar auf *eine* Kante, die den Schnitt kreuzt.
   $\Rightarrow$ Greedy macht Fortschritt. □



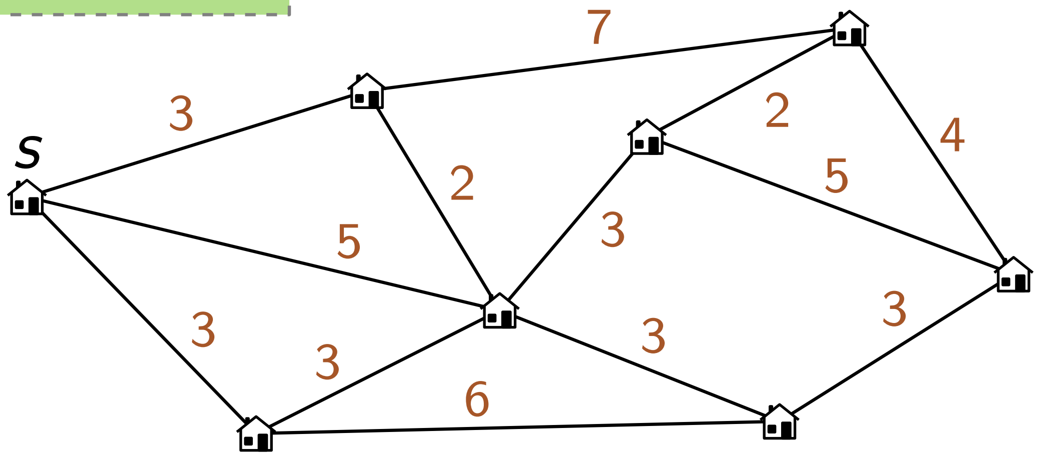
Der Algorithmus von Jarník-Prim (1930/1957)

Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)



Vojtěch Jarník
$*$ 1897 Prag
† 1970 Prag

Robert C. Prim
$*$ 1921 Sweetwater, TX
† 2021 San Clemente, CA

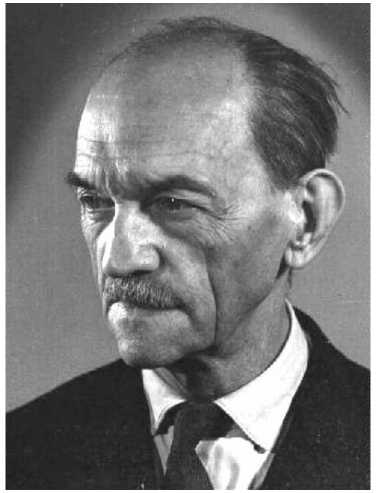


Der Algorithmus von Jarník-Prim (1930/1957)

Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

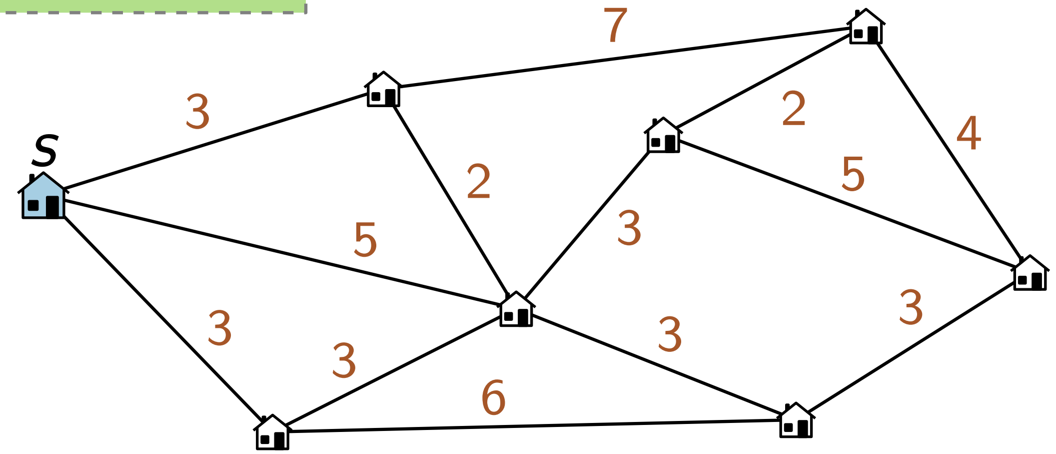
$S = \{s\}$

$E' = \emptyset$



Vojtěch Jarník
$*$ 1897 Prag
† 1970 Prag

Robert C. Prim
$*$ 1921 Sweetwater, TX
† 2021 San Clemente, CA



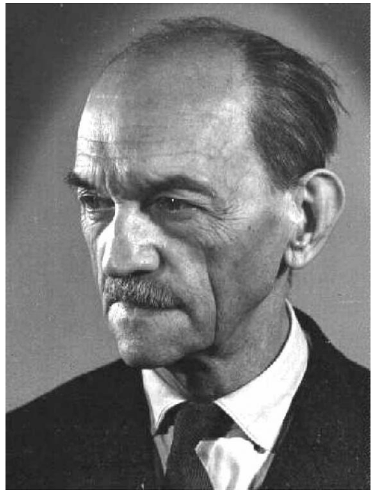
Der Algorithmus von Jarník-Prim (1930/1957)

JARNÍK-PRIM(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$

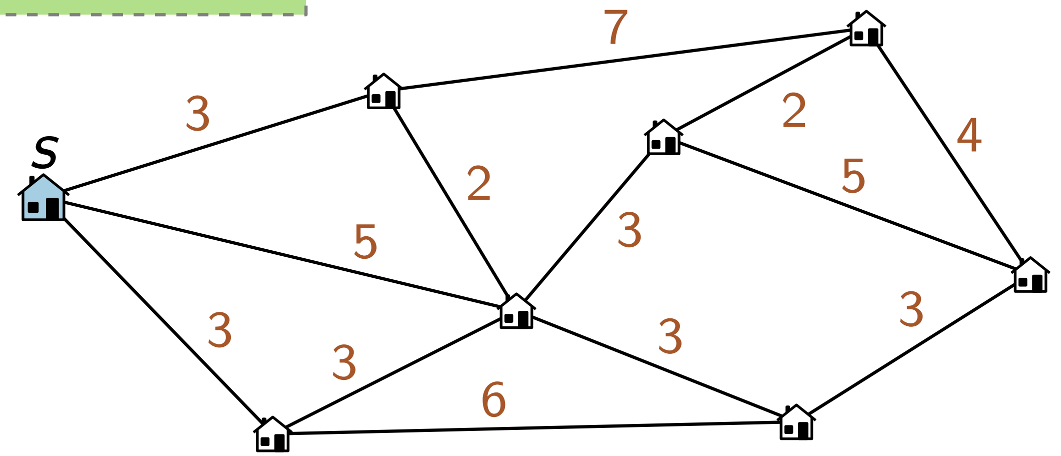
$E' = \emptyset$

while not $S == V(G)$ **do**



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

JARNÍK-PRIM(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

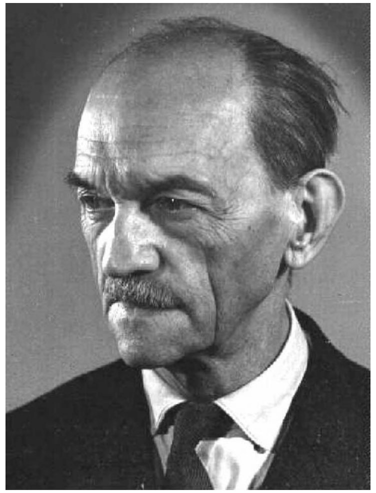
$S = \{s\}$

$E' = \emptyset$

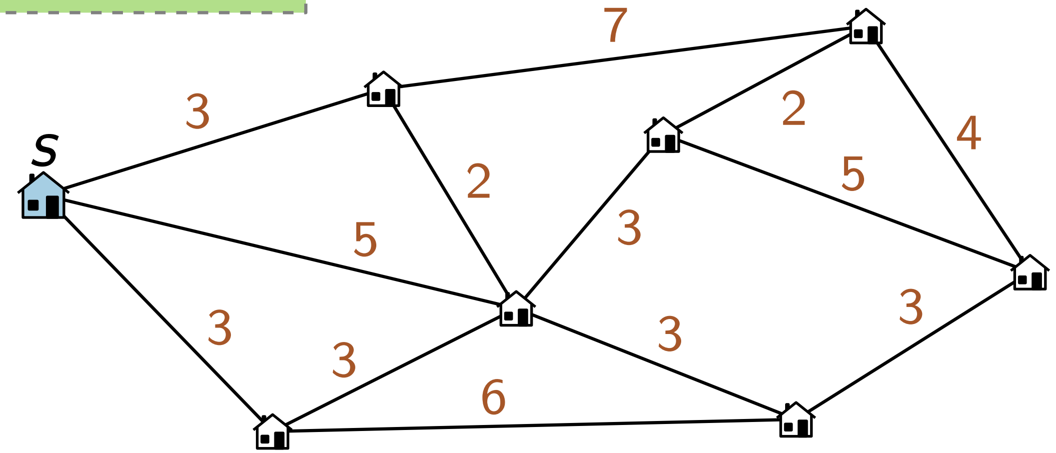
while not $S == V(G)$ **do**

Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel

Vojtěch Jarník
$*$ 1897 Prag
† 1970 Prag



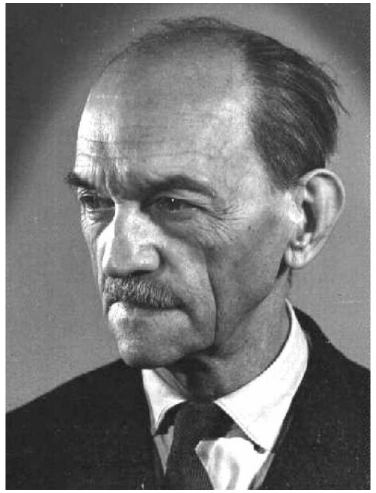
Robert C. Prim
$*$ 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

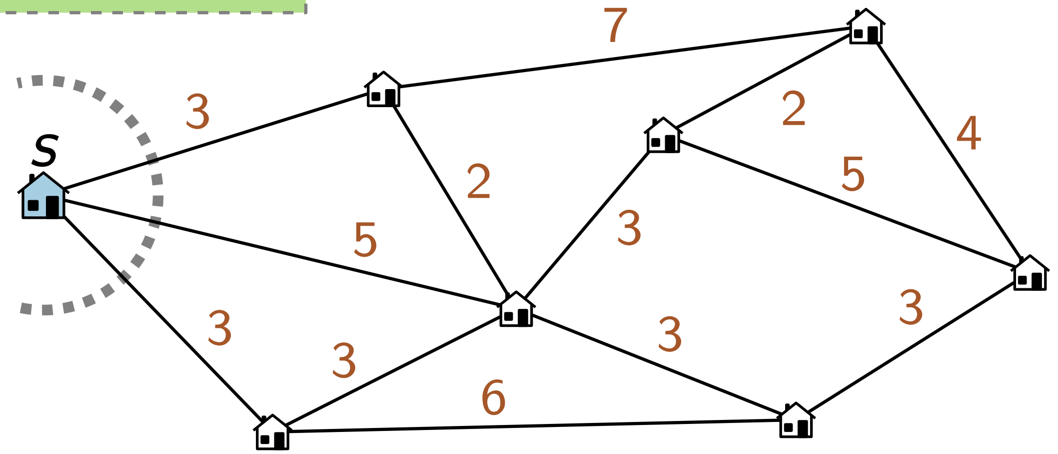
Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
    Wähle Schnitt $(S, V(G) \setminus S)$.   Blaue Regel



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

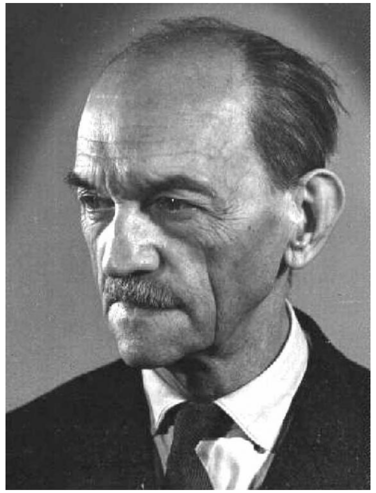
$S = \{s\}$

$E' = \emptyset$

while not $S == V(G)$ **do**

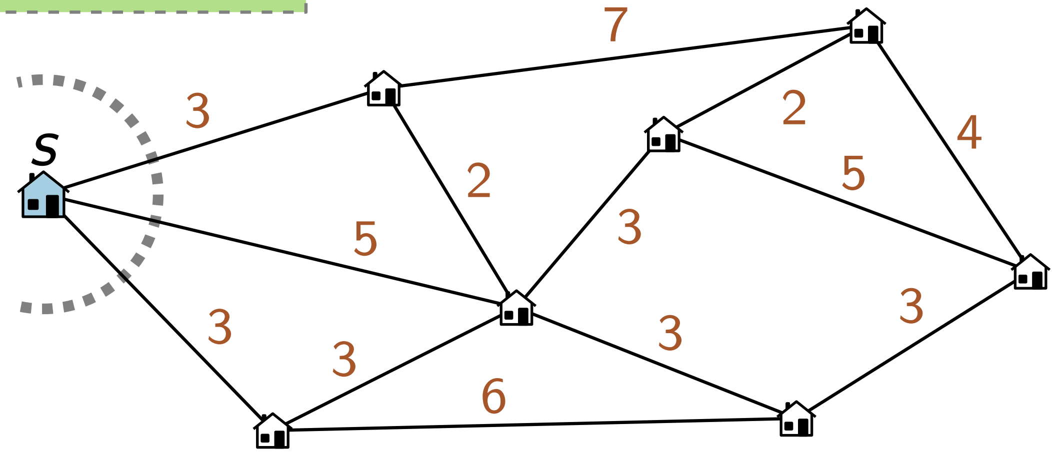
  Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel

  Färbe leichte Kante uv blau ($u \in S, v \in V(G) \setminus S$).



Vojtěch Jarník
$*$ 1897 Prag
† 1970 Prag

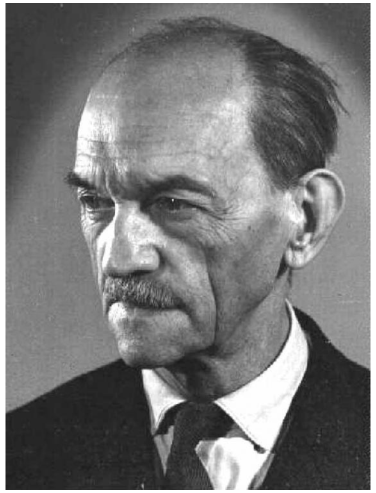
Robert C. Prim
$*$ 1921 Sweetwater, TX
† 2021 San Clemente, CA



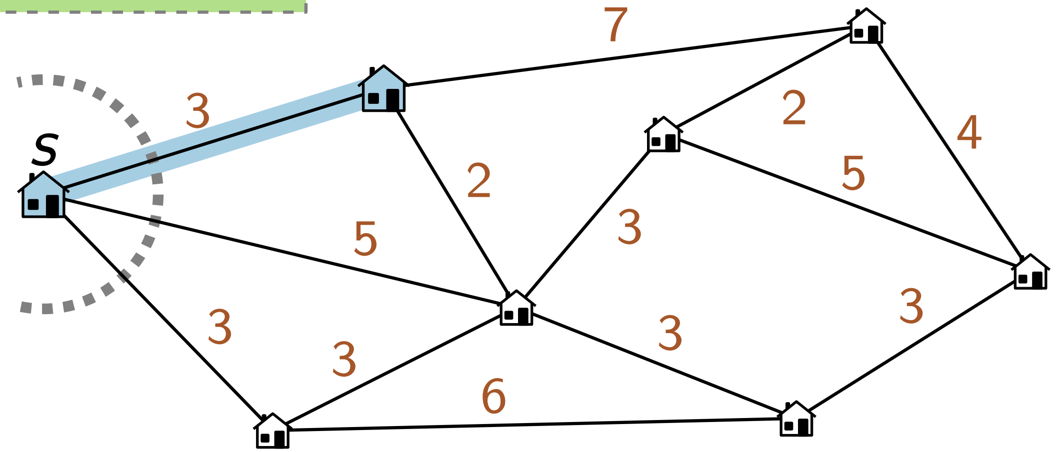
Der Algorithmus von Jarník-Prim (1930/1957)

JARNÍK-PRIM(Graph G, Weights $w \colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)
- $S = \{s\}$
- $E' = \emptyset$
- **while not** $S == V(G)$ **do**
  - Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
  - Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.

Vojtěch Jarník
* 1897 Prag
† 1970 Prag



Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA

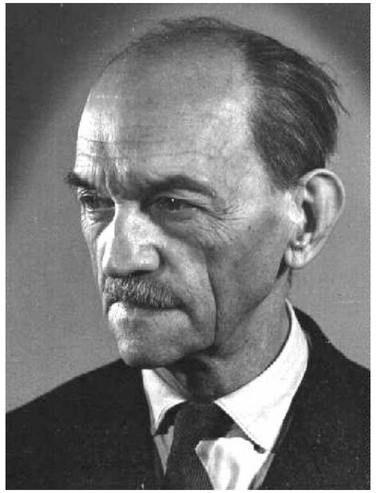


Der Algorithmus von Jarník-Prim (1930/1957)

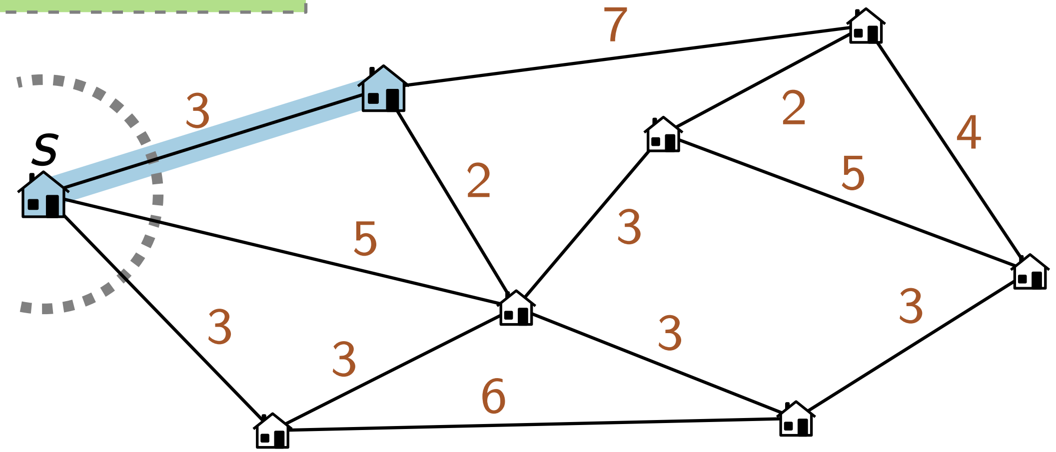
JARNÍK-PRIM(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$

Vojtěch Jarník
* 1897 Prag
† 1970 Prag



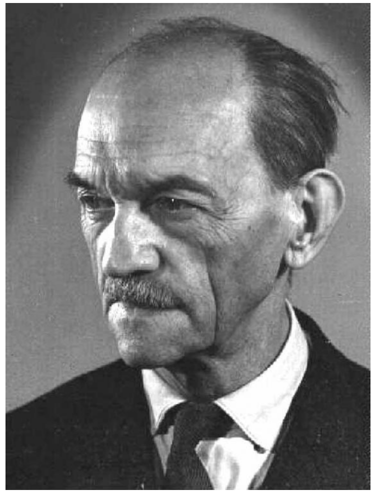
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

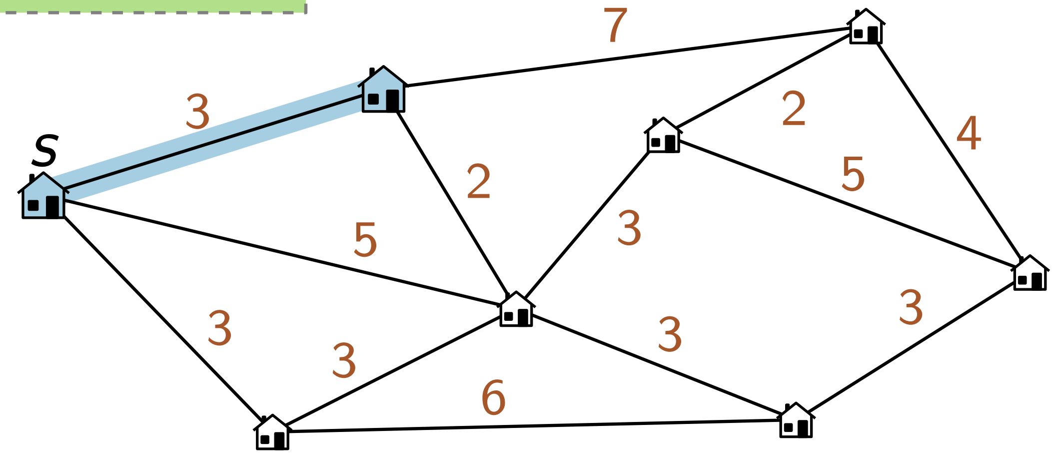
Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

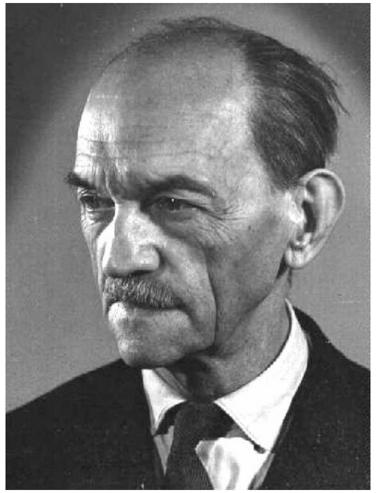
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA

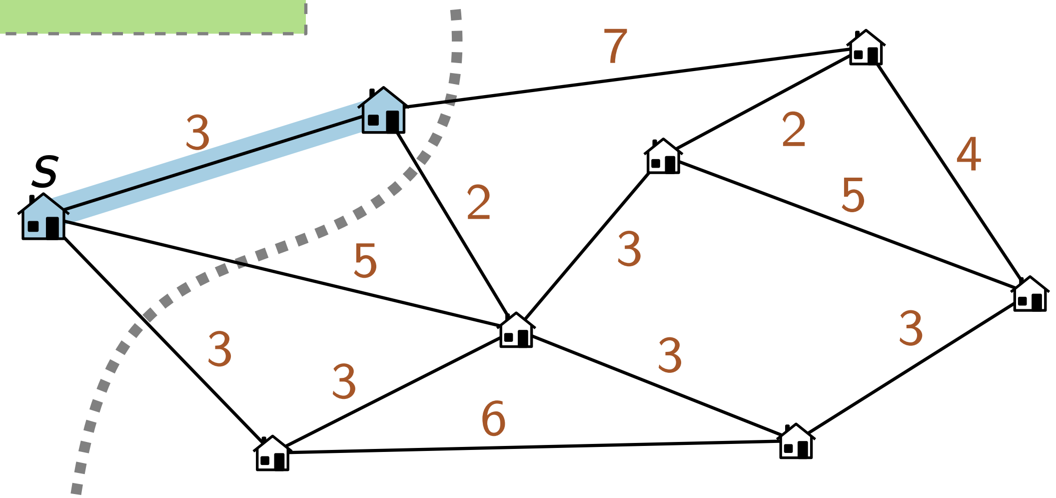


Der Algorithmus von Jarník-Prim (1930/1957)

$\textsc{Jarník-Prim}$(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$

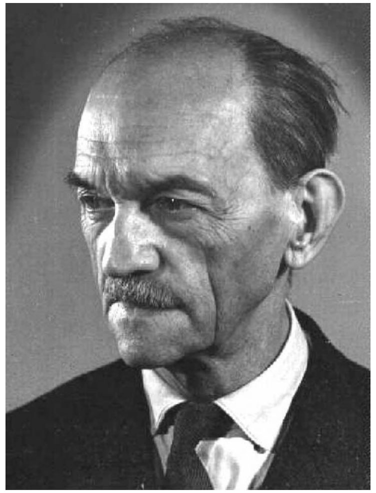




Der Algorithmus von Jarník-Prim (1930/1957)

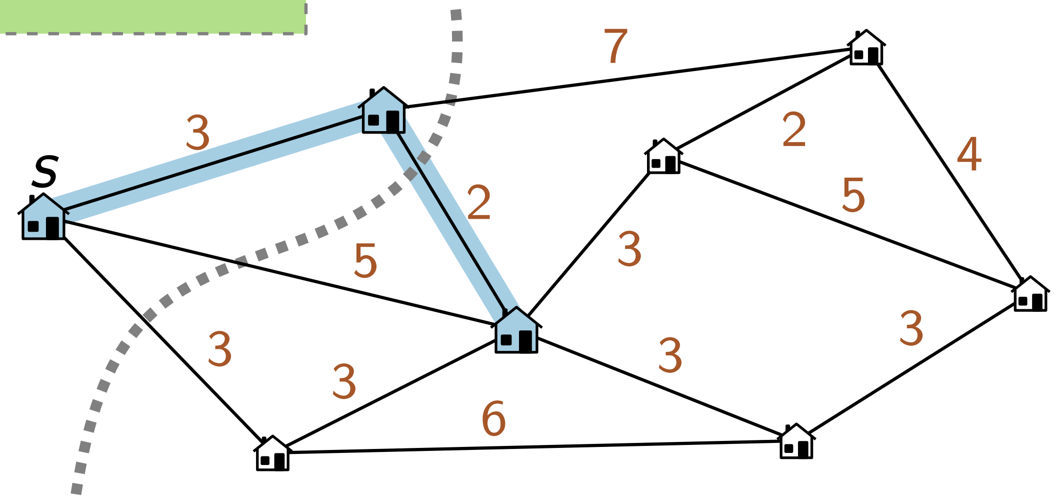
$\textsc{Jarník-Prim}$(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

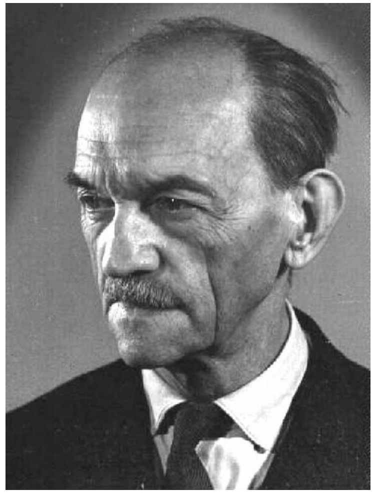
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

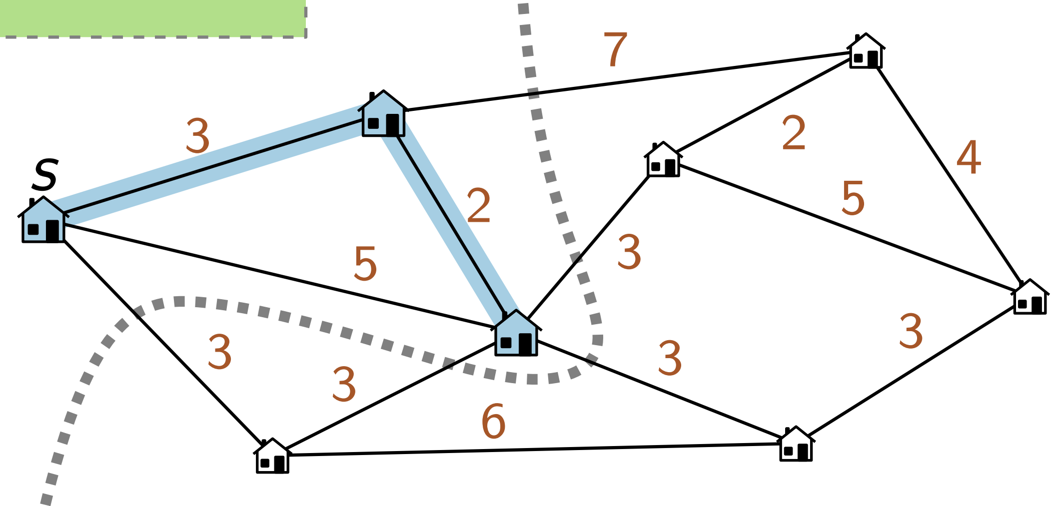
JARNÍK-PRIM(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

  $S = \{s\}$
  $E' = \emptyset$
  while not $S == V(G)$ **do**
    Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
    Färbe leichte Kante uv blau ($u \in S, v \in V(G) \setminus S$).
    $S = S \cup \{v\}$
    $E' = E' \cup \{uv\}$



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

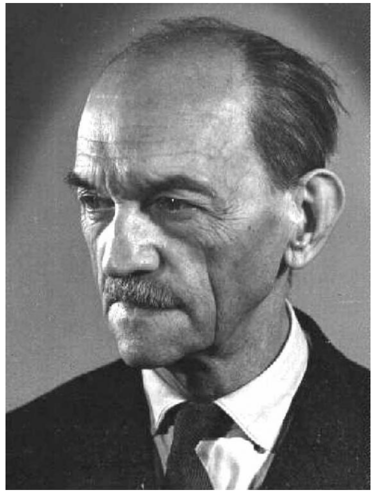
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

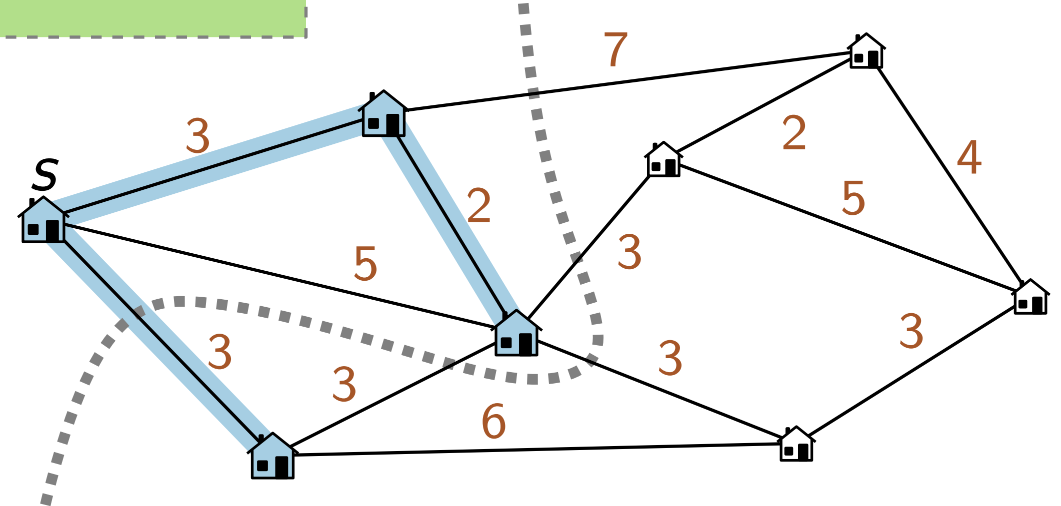
Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

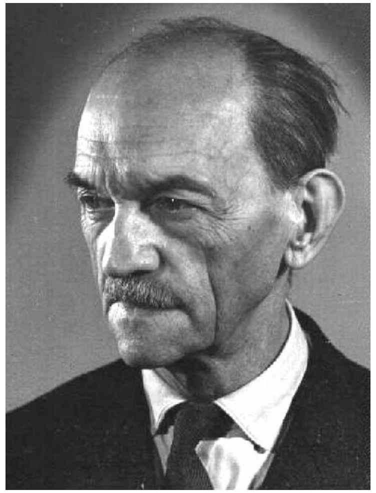
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

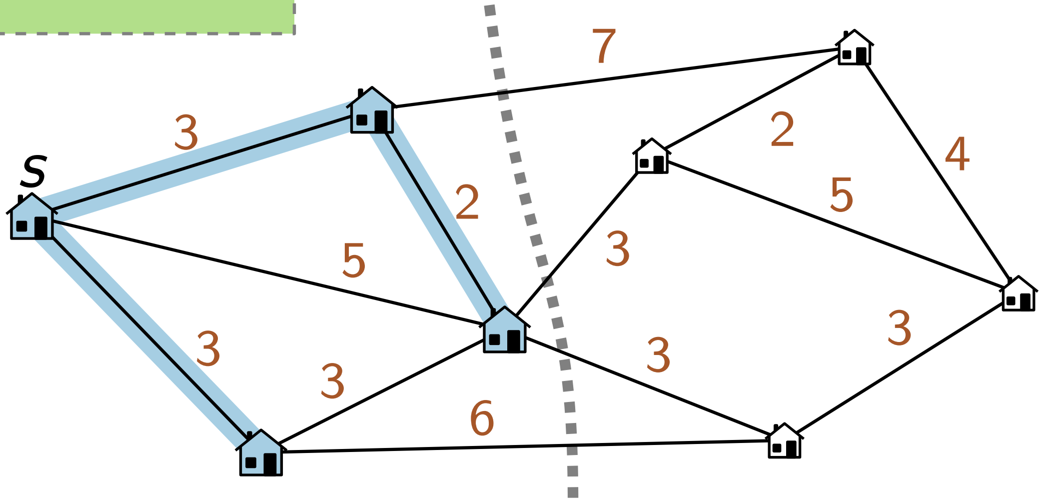
Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

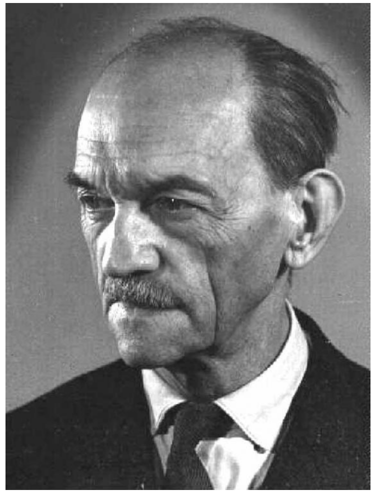
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

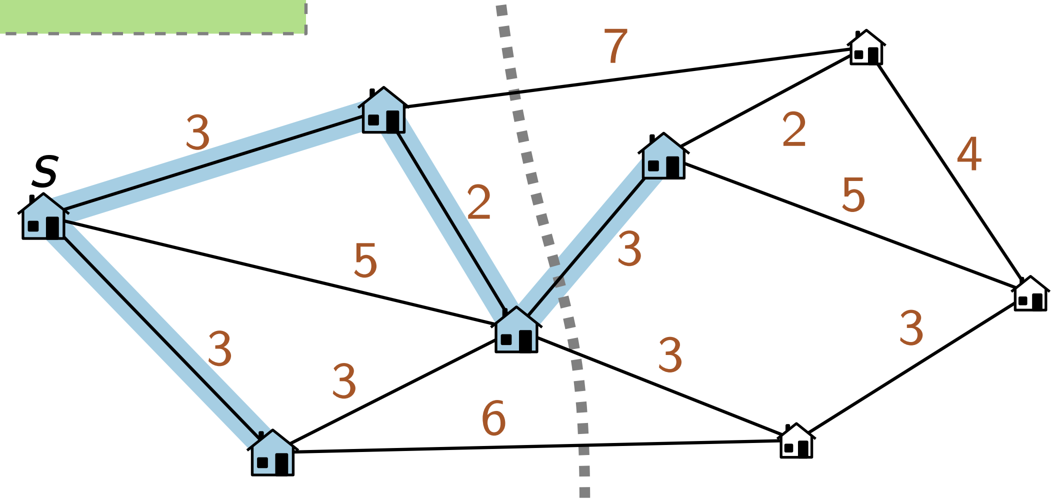
JARNÍK-PRIM(Graph G, Weights $w \colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

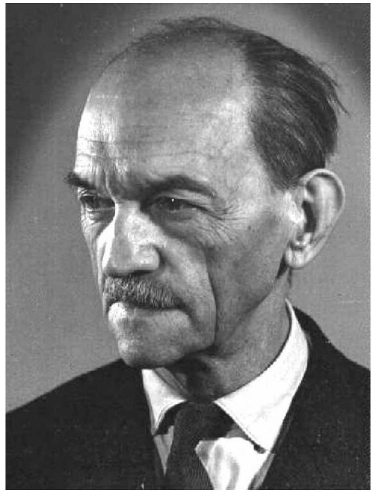
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

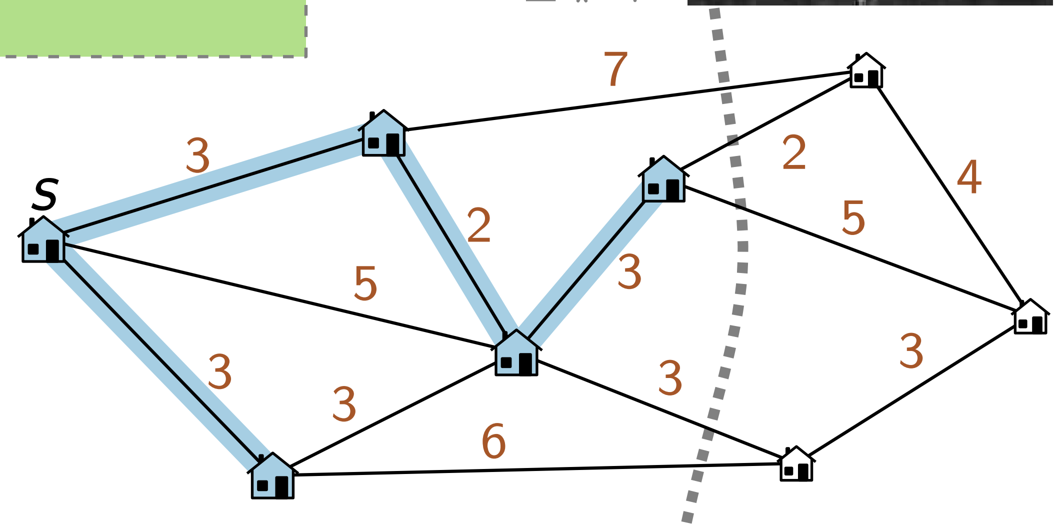
$\textsc{Jarník-Prim}(\text{Graph } G, \text{ Weights } w\colon E(G) \to \mathbb{R}_{\geq 0}), \text{ Vertex } s)$

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
- Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel
- Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
- $S = S \cup \{v\}$
- $E' = E' \cup \{uv\}$



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

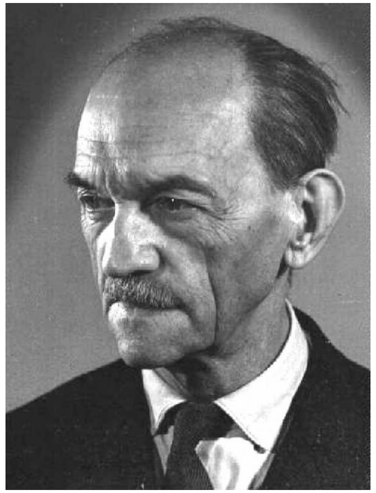
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

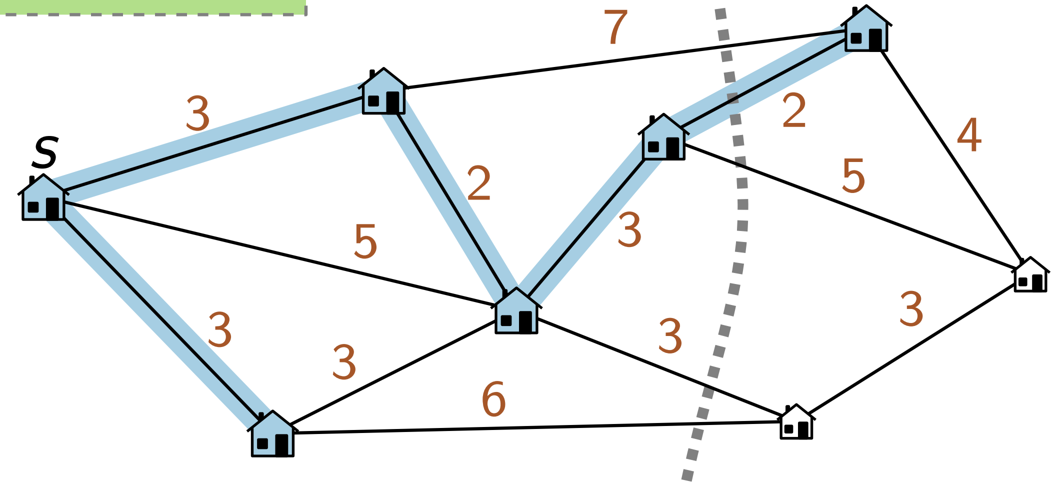
Jarník-Prim(Graph G, Weights $w \colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
 Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
 Färbe leichte Kante uv blau ($u \in S, v \in V(G) \setminus S$).
 $S = S \cup \{v\}$
 $E' = E' \cup \{uv\}$



Vojtěch Jarník
$*$ 1897 Prag
† 1970 Prag

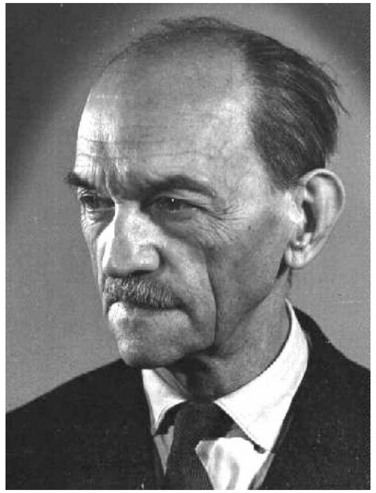
Robert C. Prim
$*$ 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

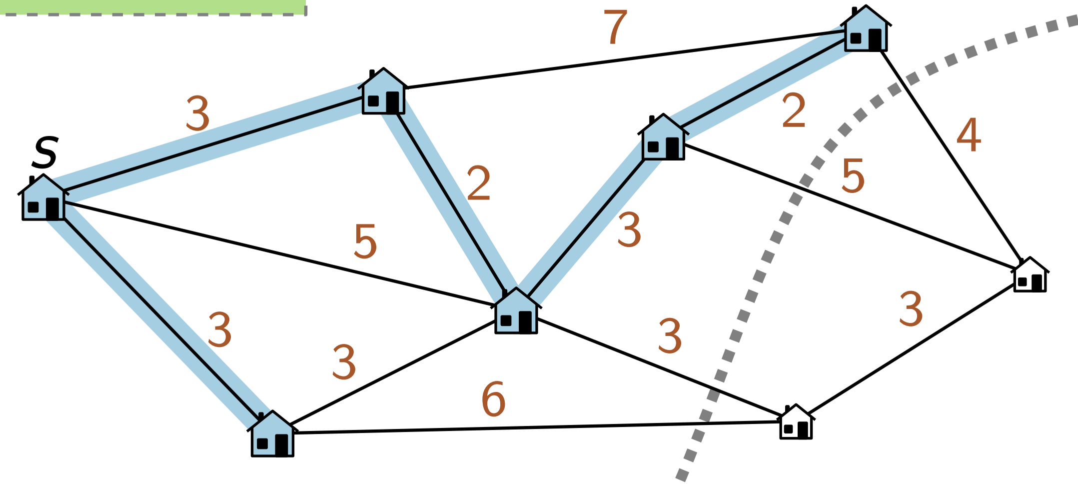
Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

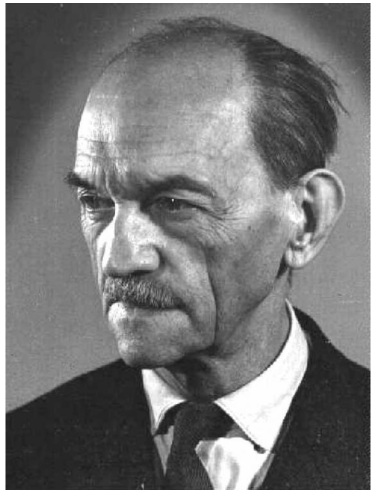
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

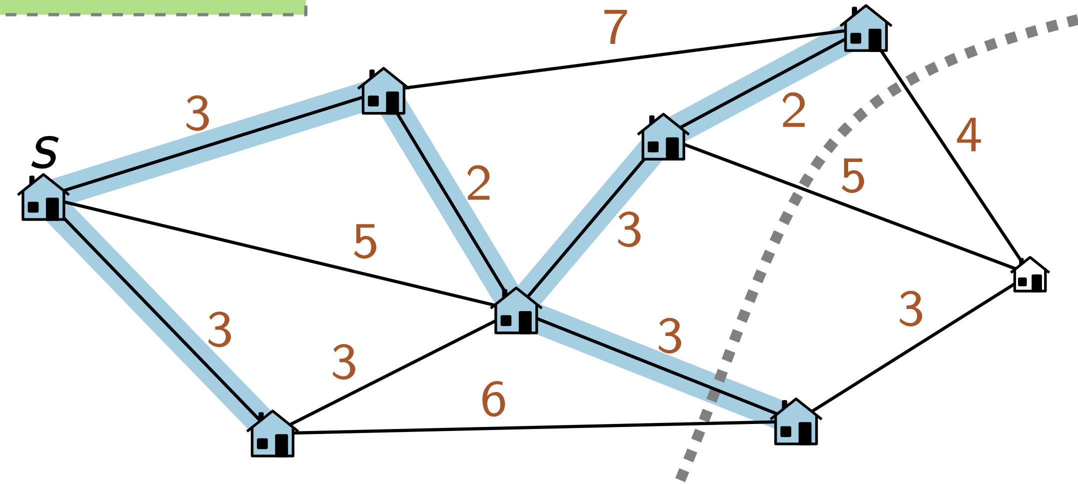
Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$



Vojtěch Jarník
∗ 1897 Prag
† 1970 Prag

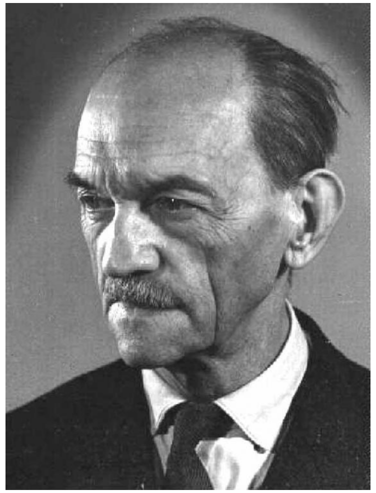
Robert C. Prim
∗ 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

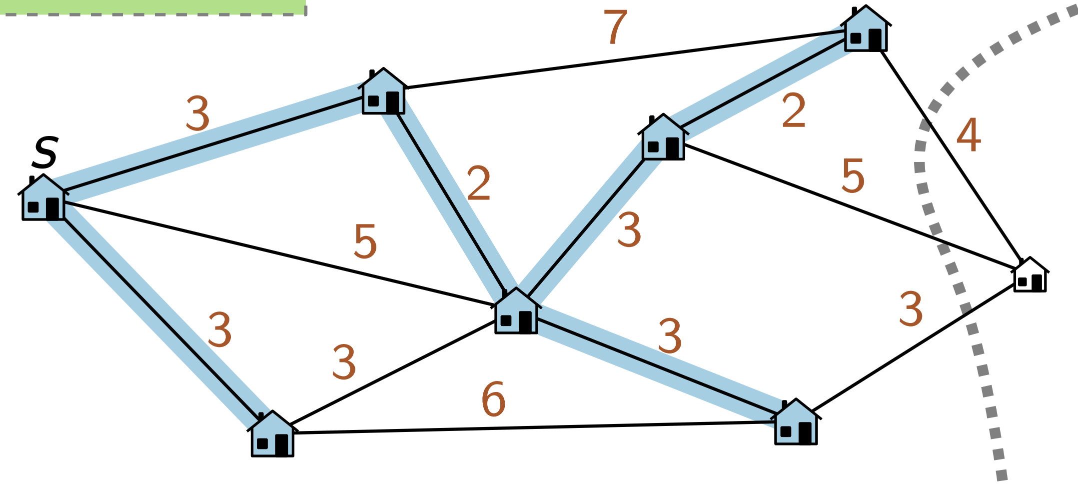
$\textsc{Jarník-Prim}$(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$



Vojtěch Jarník
∗ 1897 Prag
† 1970 Prag

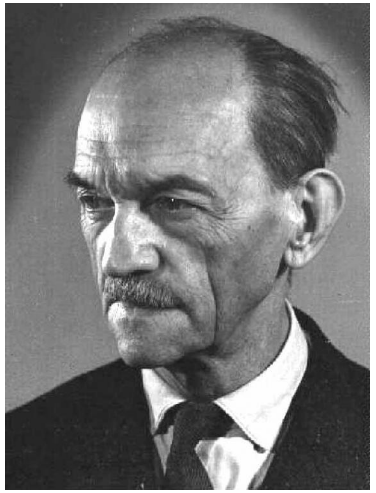
Robert C. Prim
∗ 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

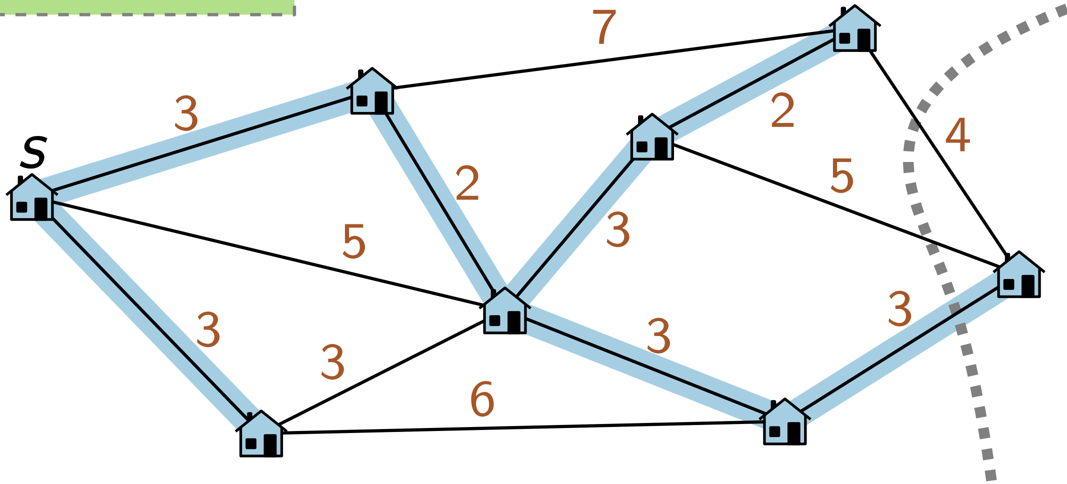
$\textsc{Jarník-Prim}$(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

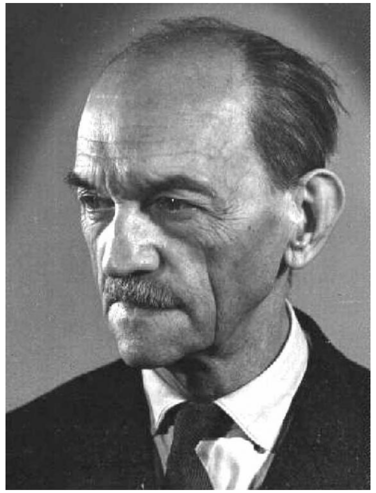
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

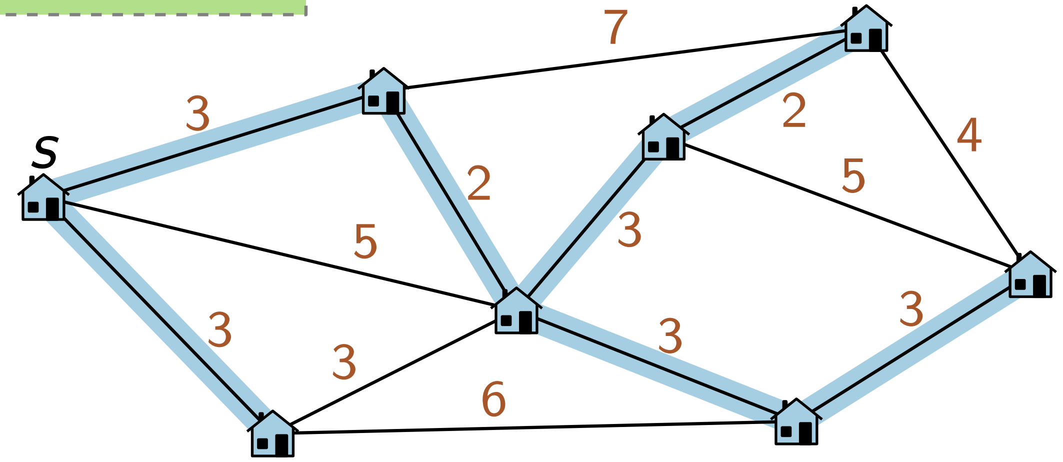
$\textsc{Jarník-Prim}$(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel
  Färbe leichte Kante uv blau ($u \in S, v \in V(G) \setminus S$).
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$



Vojtěch Jarník
$*$ 1897 Prag
† 1970 Prag

Robert C. Prim
$*$ 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$

$E' = \emptyset$

while not $S == V(G)$ **do**

  Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel

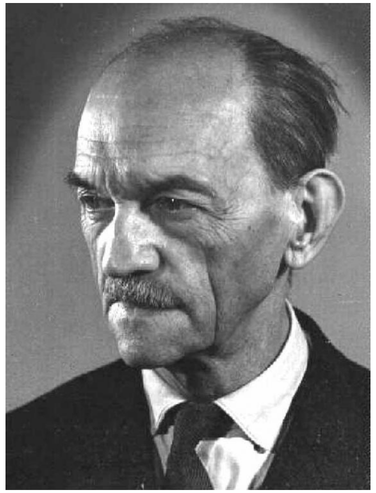
  Färbe leichte Kante uv blau ($u \in S, v \in V(G) \setminus S$).

  $S = S \cup \{v\}$

  $E' = E' \cup \{uv\}$

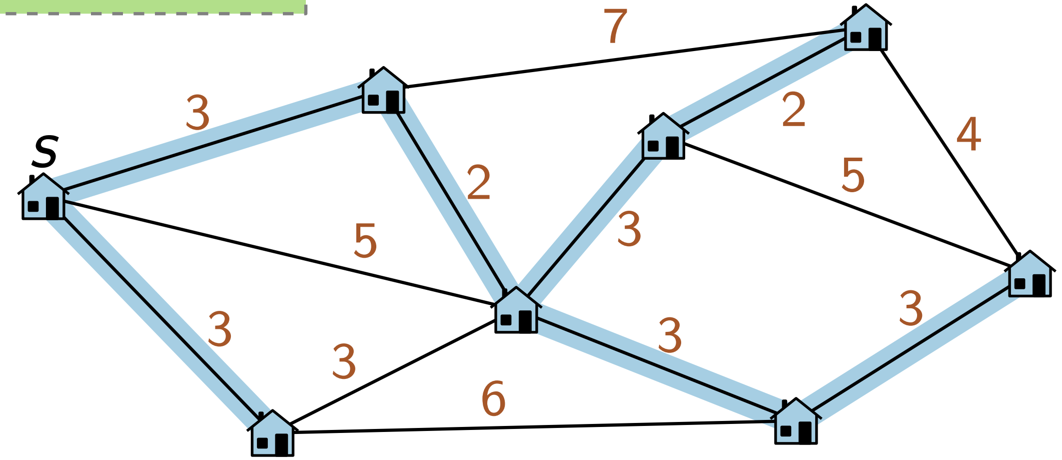
Färbe alle anderen Kanten rot. — Rote Regel

return E'



Vojtěch Jarník
$*$ 1897 Prag
† 1970 Prag

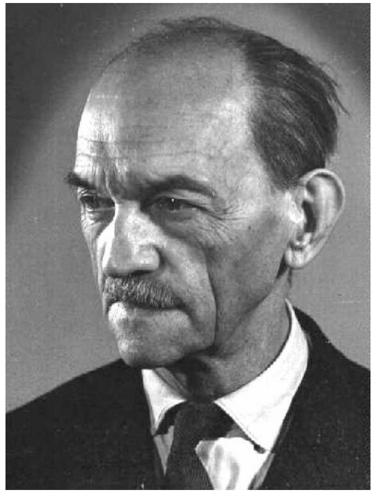
Robert C. Prim
$*$ 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

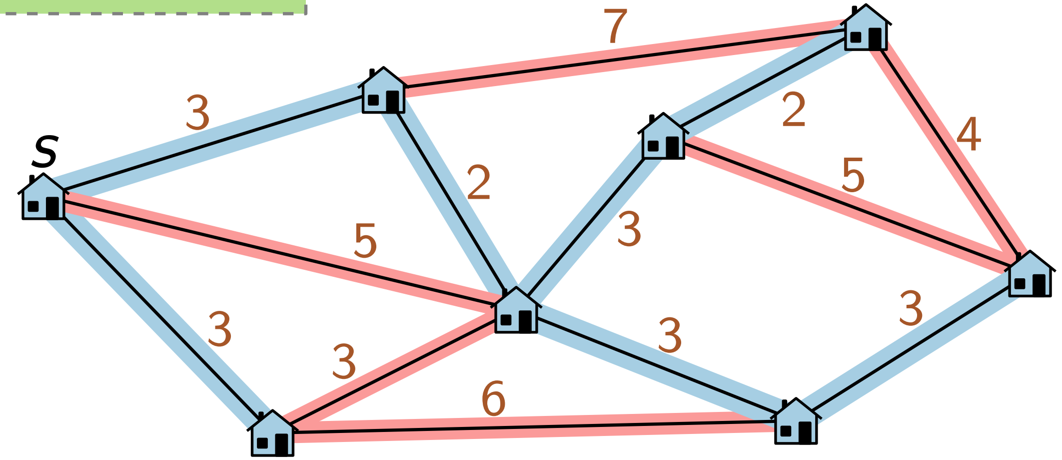
$\textsc{Jarník-Prim}$(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel
  Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$
Färbe alle anderen Kanten rot. — Rote Regel
return E'



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

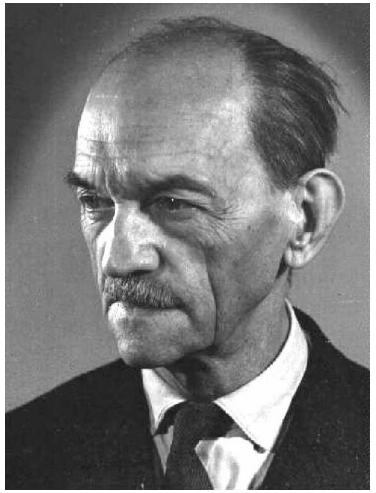
Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA



Der Algorithmus von Jarník-Prim (1930/1957)

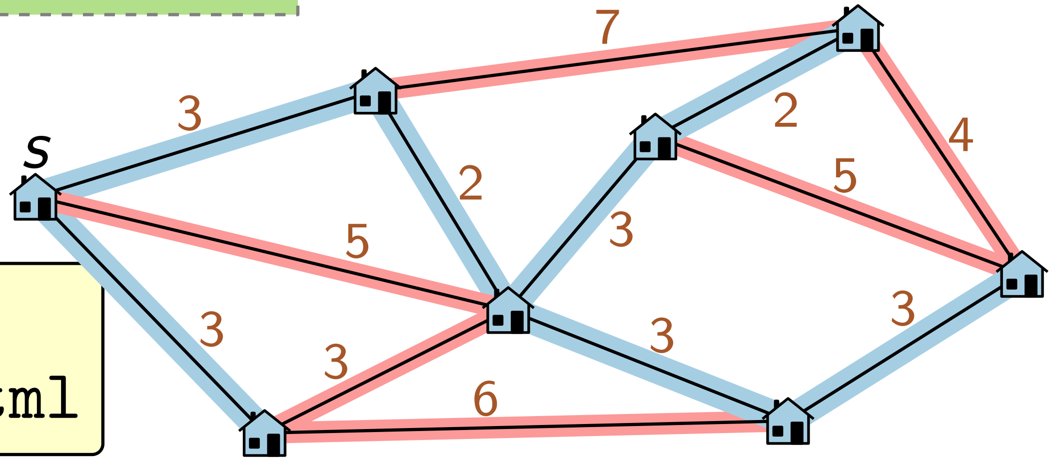
Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel
  Färbe leichte Kante uv blau ($u \in S, v \in V(G) \setminus S$).
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$
Färbe alle anderen Kanten rot. — Rote Regel
return E'



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA

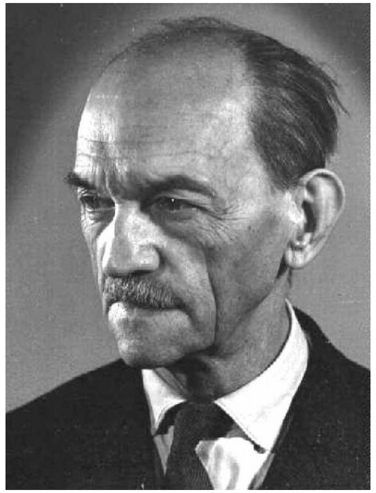


Demo.
https://algo.uni-trier.de/demos/spanningtree.html

Der Algorithmus von Jarník-Prim (1930/1957)

$\textsc{Jarník-Prim}$(Graph G, Weights $w \colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

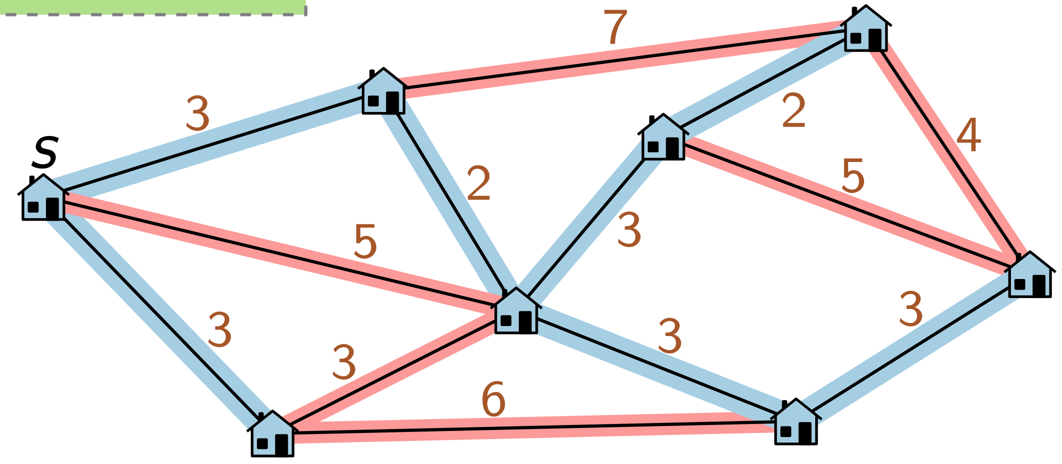
  $S = \{s\}$
  $E' = \emptyset$
  while not $S == V(G)$ **do**
    Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel
    Färbe leichte Kante uv blau ($u \in S, v \in V(G) \setminus S$).
    $S = S \cup \{v\}$
    $E' = E' \cup \{uv\}$
  Färbe alle anderen Kanten rot. — Rote Regel
  return E'



Vojtěch Jarník
* 1897 Prag
† 1970 Prag

Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA

Laufzeit?



Der Algorithmus von Jarník-Prim (1930/1957)

Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$

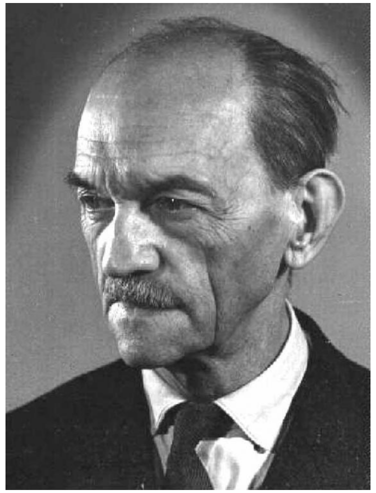
$E' = \emptyset$

while not $S == V(G)$ **do**

- Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel
- Färbe leichte Kante uv blau ($u \in S, v \in V(G) \setminus S$).
- $S = S \cup \{v\}$
- $E' = E' \cup \{uv\}$

Färbe alle anderen Kanten rot. — Rote Regel

return E'

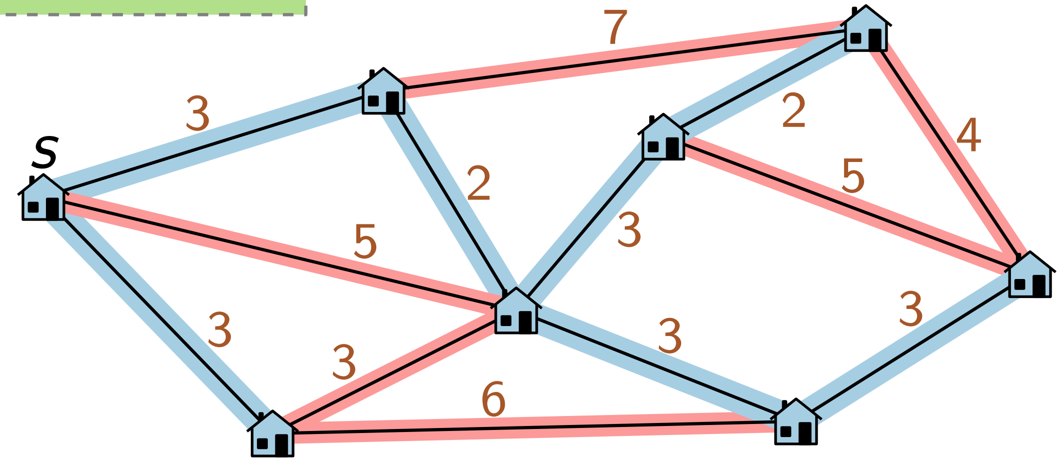


Vojtěch Jarník
* 1897 Prag
† 1970 Prag

Robert C. Prim
* 1921 Sweetwater, TX
† 2021 San Clemente, CA

Laufzeit?

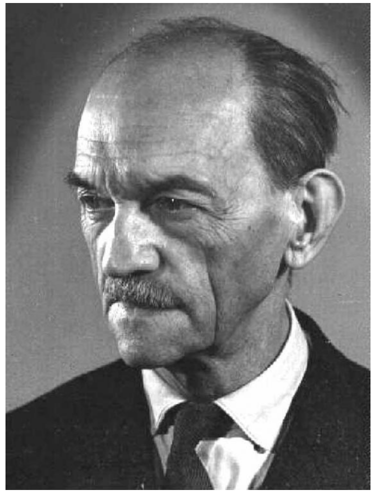
Wie Dijkstra!



Der Algorithmus von Jarník-Prim (1930/1957)

$\textsc{Jarník-Prim}$(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
  Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel
  Färbe leichte Kante uv blau ($u \in S, v \in V(G) \setminus S$).
  $S = S \cup \{v\}$
  $E' = E' \cup \{uv\}$
Färbe alle anderen Kanten rot. — Rote Regel
return E'



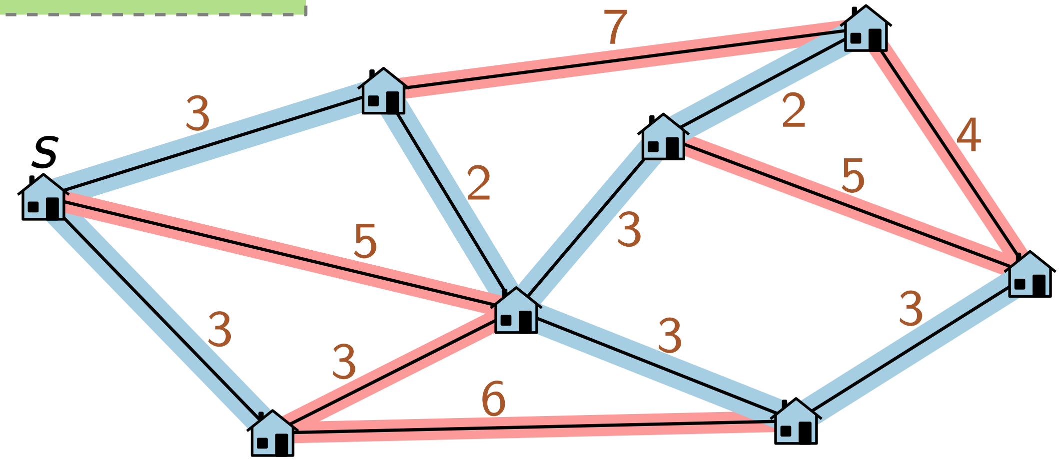
Vojtěch Jarník
$*$ 1897 Prag
† 1970 Prag

Robert C. Prim
$*$ 1921 Sweetwater, TX
† 2021 San Clemente, CA

Laufzeit?

Wie $\textsc{Dijkstra}$!

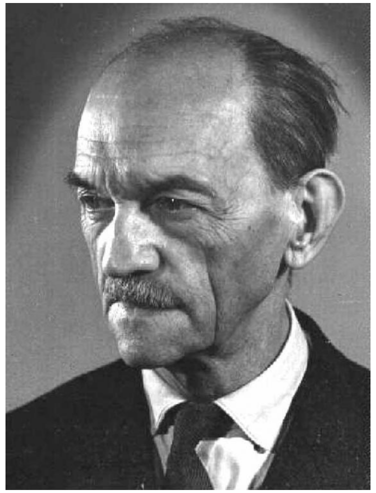
$\Rightarrow \mathcal{O}((E + V) \log V)$ $\boxed{\textsc{Heap/RS-Baum}}$



Der Algorithmus von Jarník-Prim (1930/1957)

Jarník-Prim(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$), Vertex s)

$S = \{s\}$
$E' = \emptyset$
while not $S == V(G)$ **do**
    Wähle Schnitt $(S, V(G) \setminus S)$. — Blaue Regel
    Färbe leichte Kante uv blau $(u \in S, v \in V(G) \setminus S)$.
    $S = S \cup \{v\}$
    $E' = E' \cup \{uv\}$
Färbe alle anderen Kanten rot. — Rote Regel
return E'



Vojtěch Jarník
$*$ 1897 Prag
† 1970 Prag

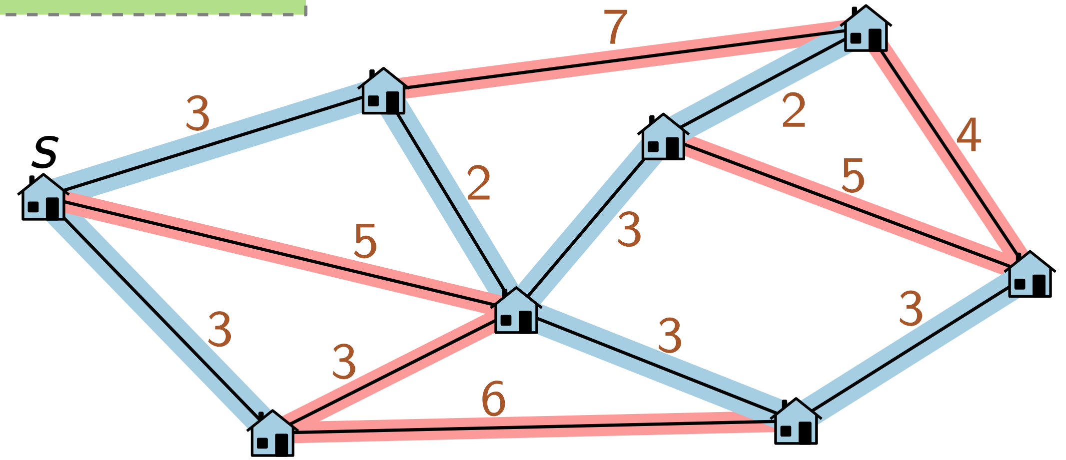
Robert C. Prim
$*$ 1921 Sweetwater, TX
† 2021 San Clemente, CA

Laufzeit?

Wie Dijkstra!

$\Rightarrow \mathcal{O}((E + V)\log V)$ — Heap/RS-Baum

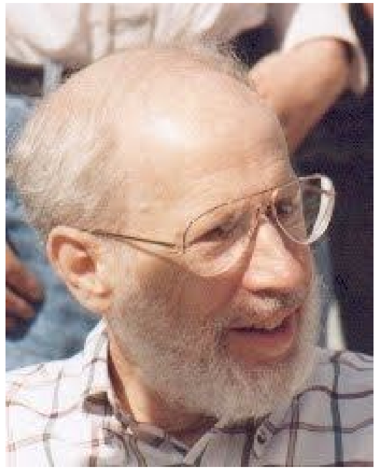
$\Rightarrow \mathcal{O}(E + V \log V)$ — FibonacciHeap

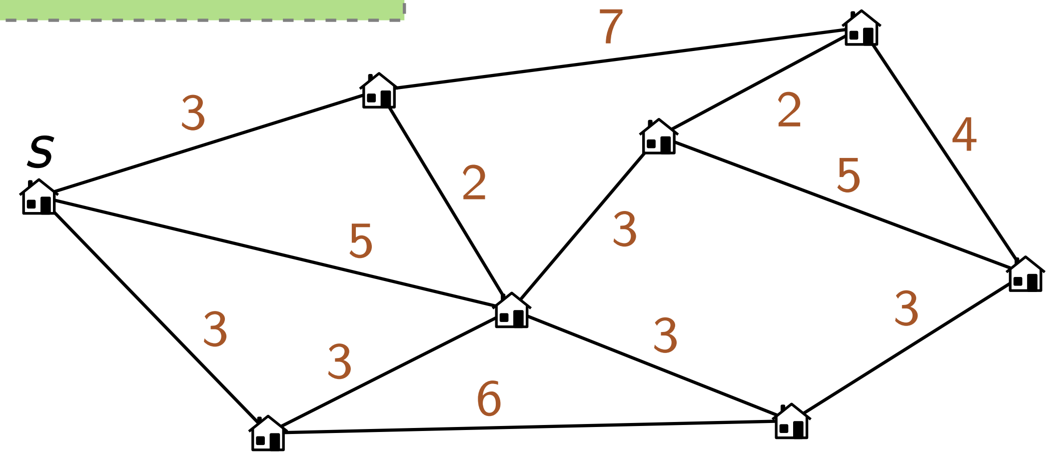


Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w\colon E(G) \to \mathbb{R}_{\geq 0}))$

Joseph Bernard Kruskal, Jr.
$*$ 1928 New York, NY
† 2010 Maplewood, NJ



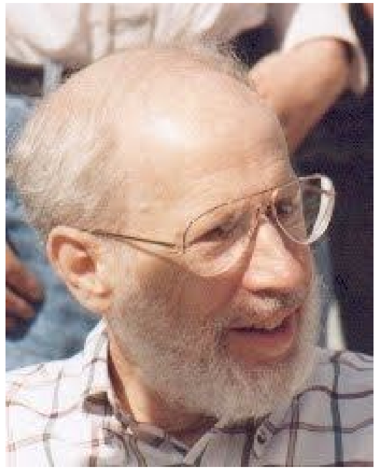


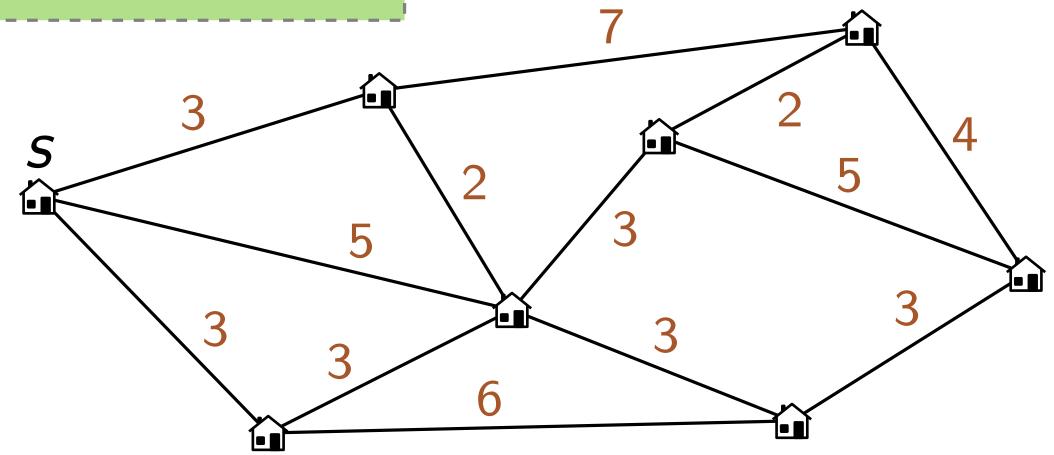
Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{ Weights } w \colon E(G) \to \mathbb{R}_{\geq 0})$

$E' = \emptyset$

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





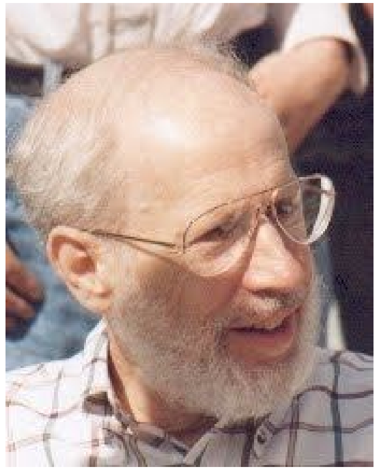
Der Algorithmus von Kruskal (1956)

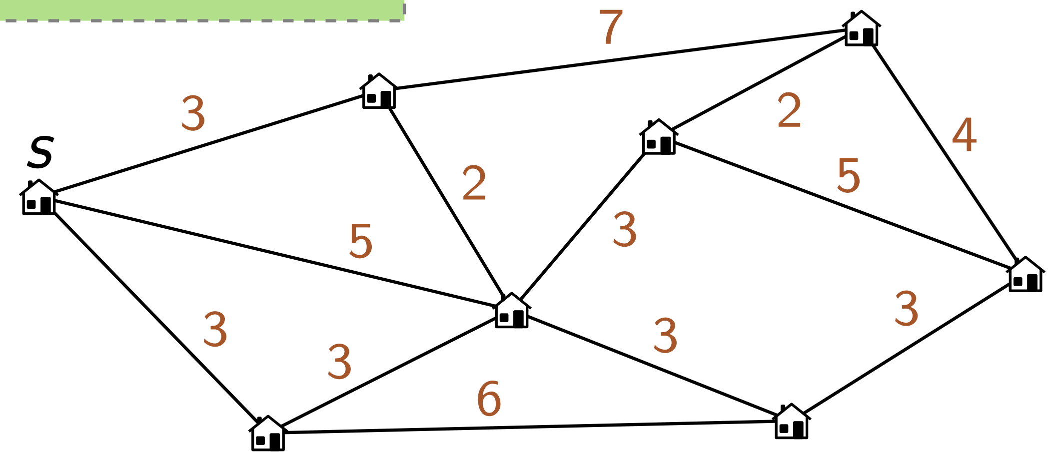
KRUSKAL(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$)

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

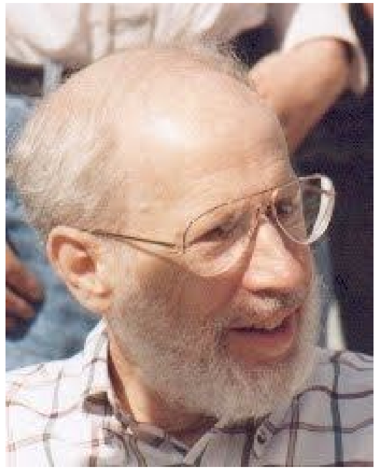
$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

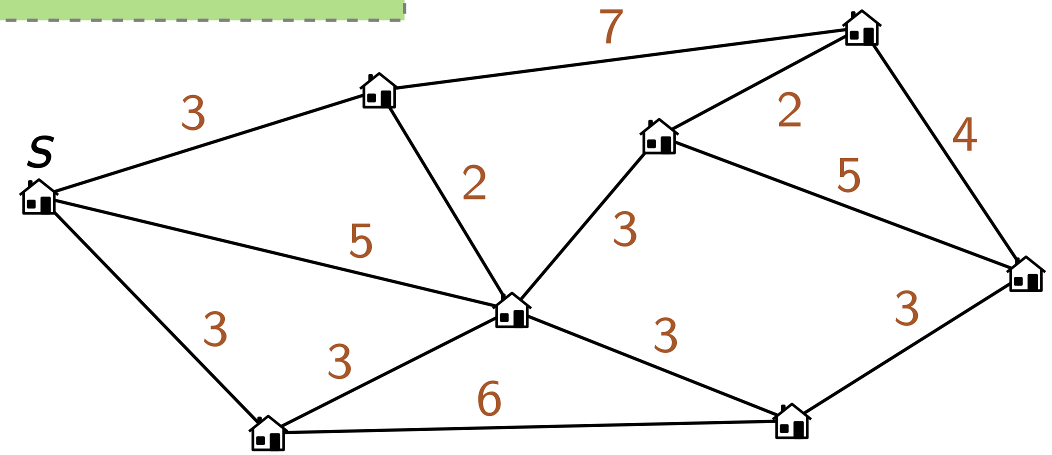
$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do**

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

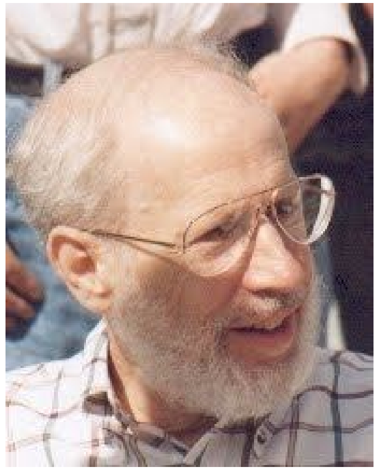
$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

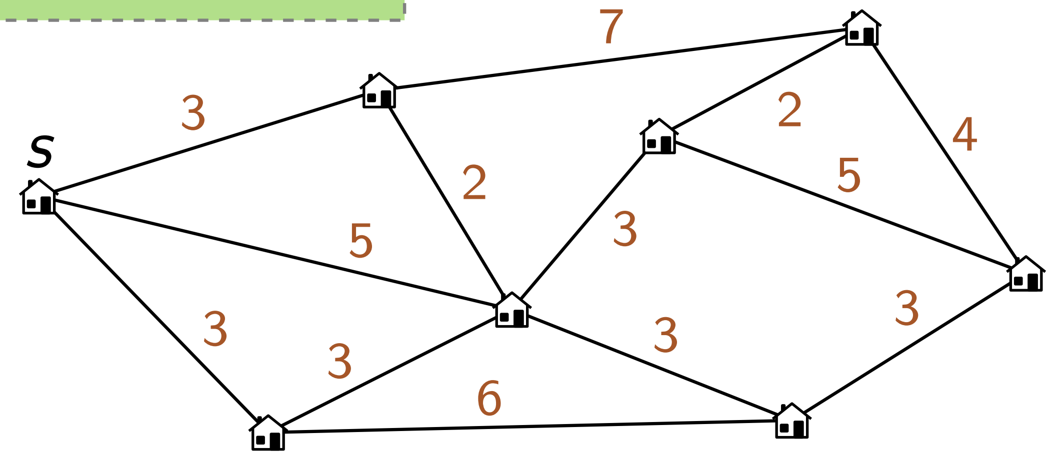
$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

KRUSKAL(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$)

$E' = \emptyset$

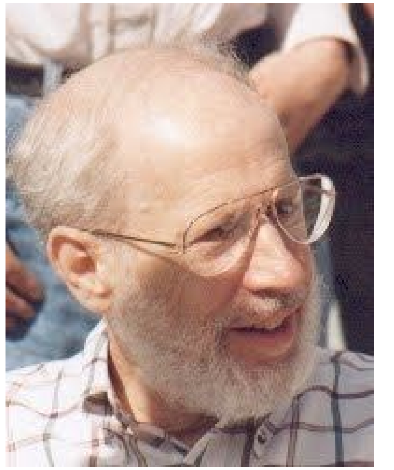
Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

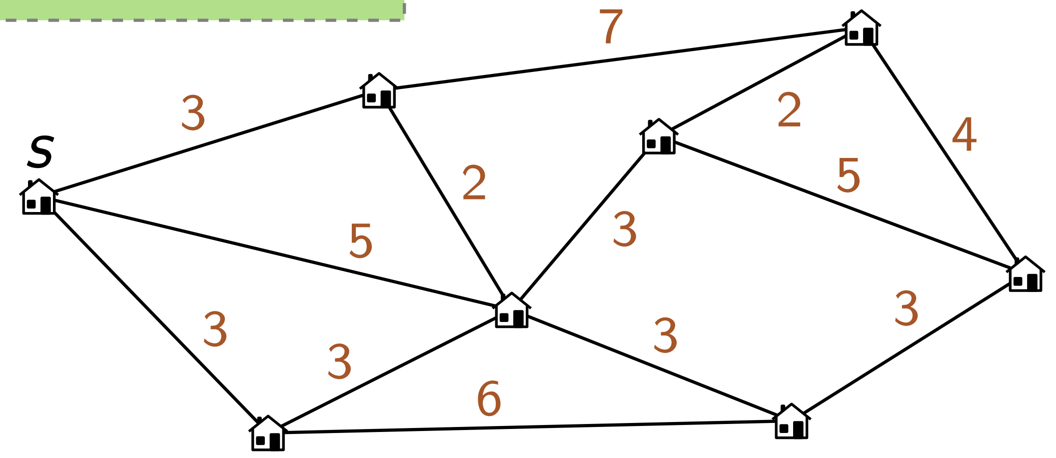
foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

  else

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

Kruskal(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$))

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

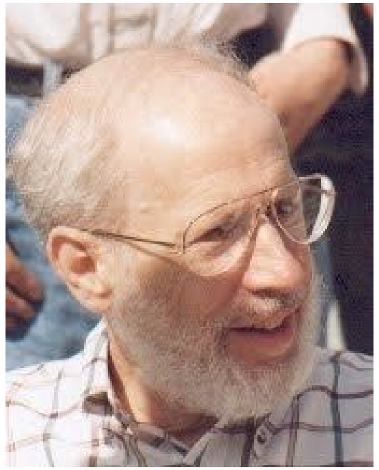
    Färbe uv blau.

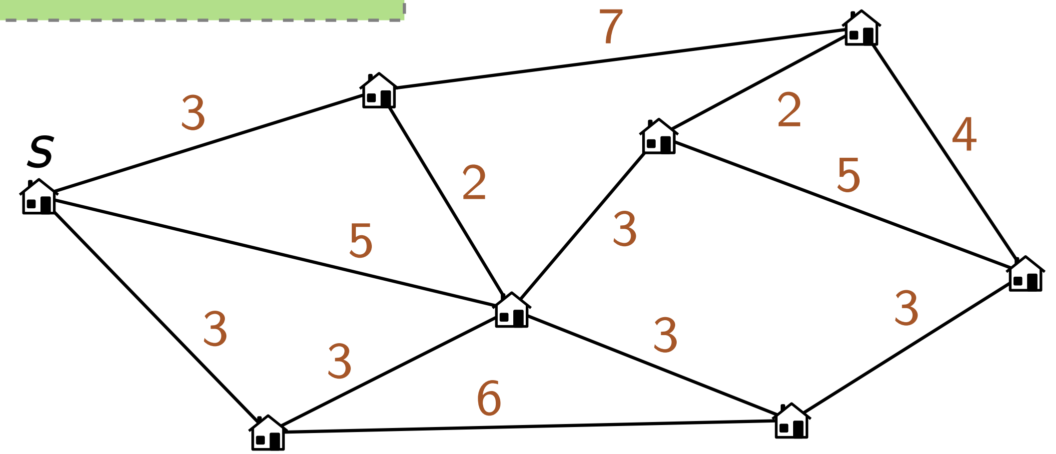
    $E' = E' \cup \{uv\}$

  else

Blaue Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

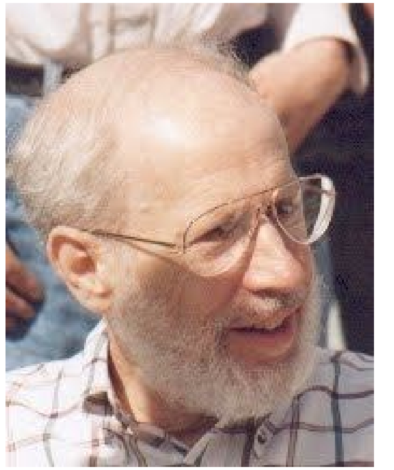
    Färbe uv blau.

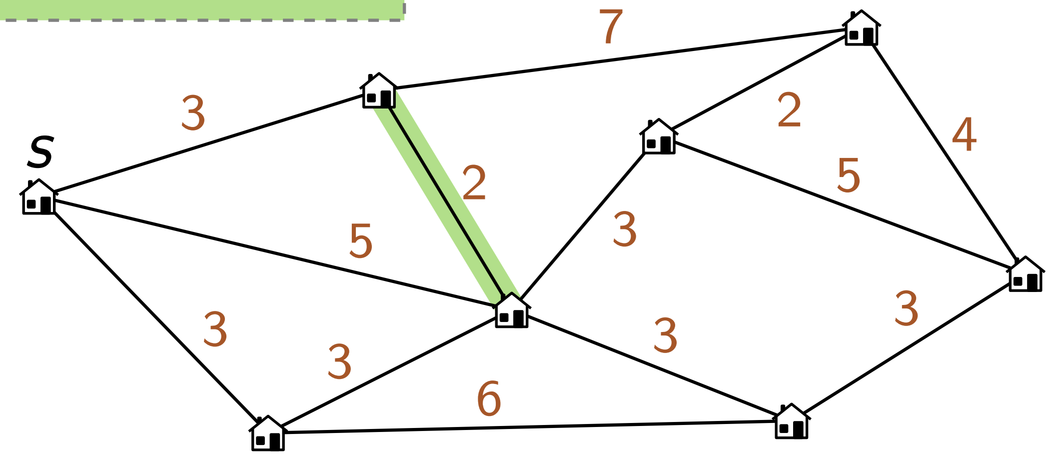
    $E' = E' \cup \{uv\}$

  else

Blaue Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





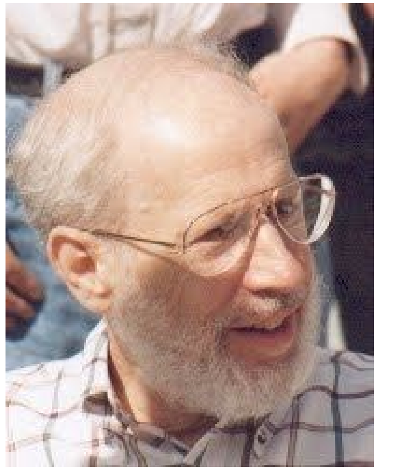
Der Algorithmus von Kruskal (1956)

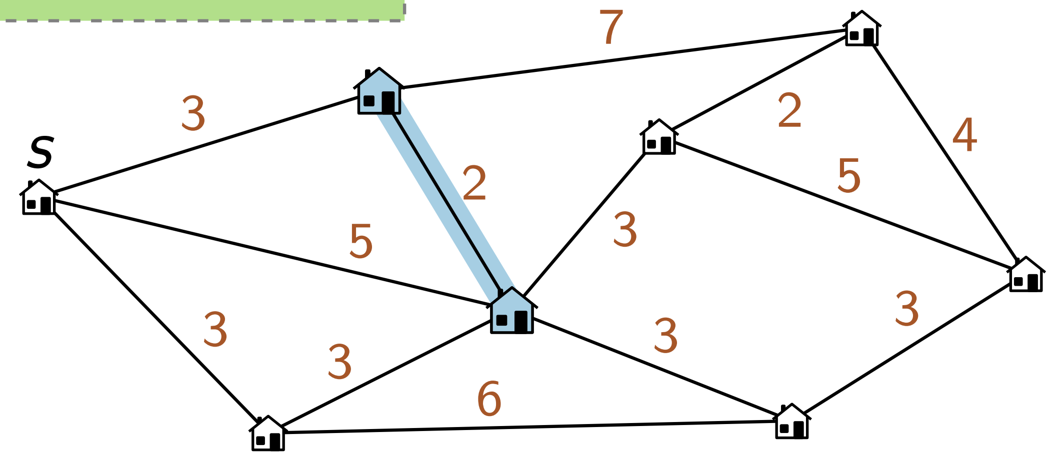
Kruskal(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$)

$E' = \emptyset$
Sortiere $E(G)$ nicht-absteigend nach Gewicht w.
foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)
  if $E' \cup \{uv\}$ enthält keinen Kreis **then**
    Färbe uv blau.
    $E' = E' \cup \{uv\}$
  else

Blaue Regel

Joseph Bernard Kruskal, Jr.
$*$ 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

Kruskal(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$)

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

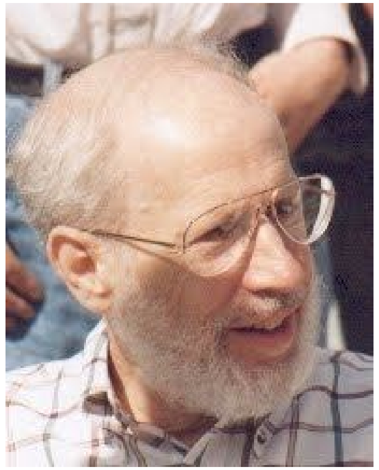
    Färbe uv blau.

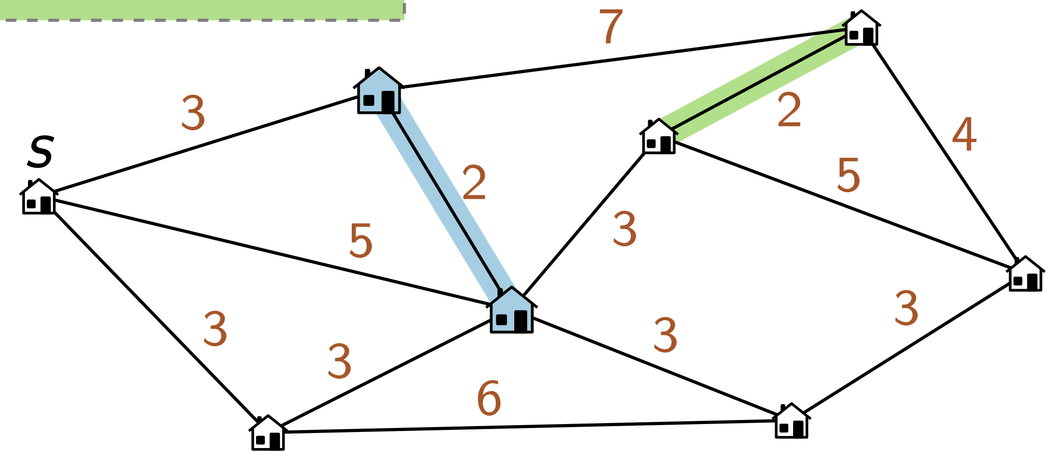
    $E' = E' \cup \{uv\}$

  else

Blaue Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

KRUSKAL(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$)

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

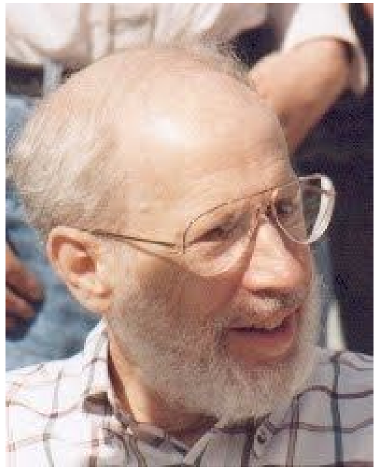
    Färbe uv blau.

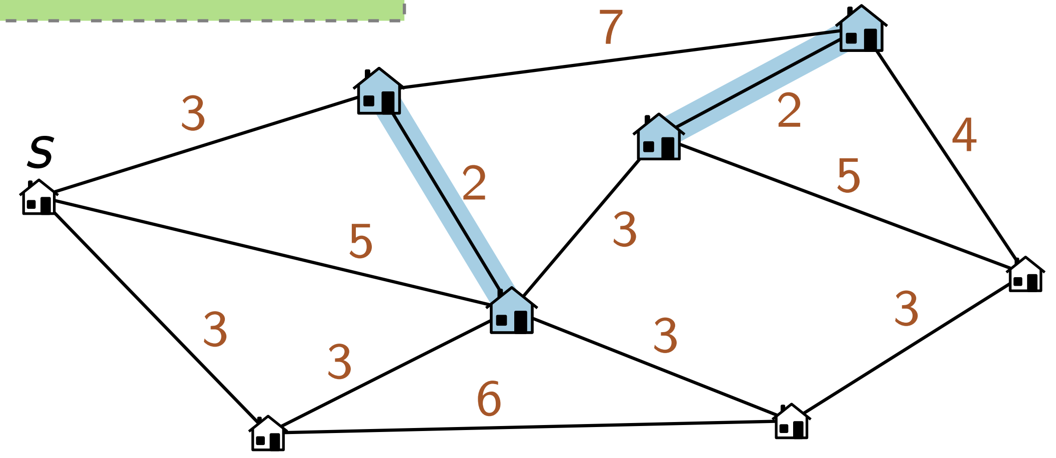
    $E' = E' \cup \{uv\}$

  else

Blaue Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w\colon E(G) \to \mathbb{R}_{\geq 0}))$

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

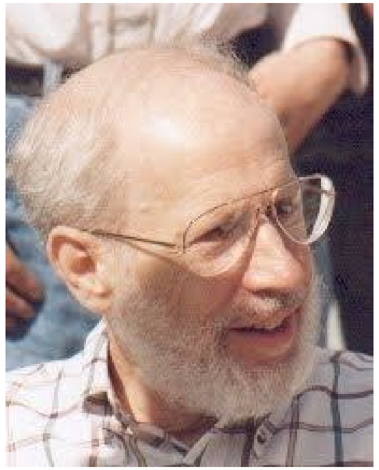
    Färbe uv blau.

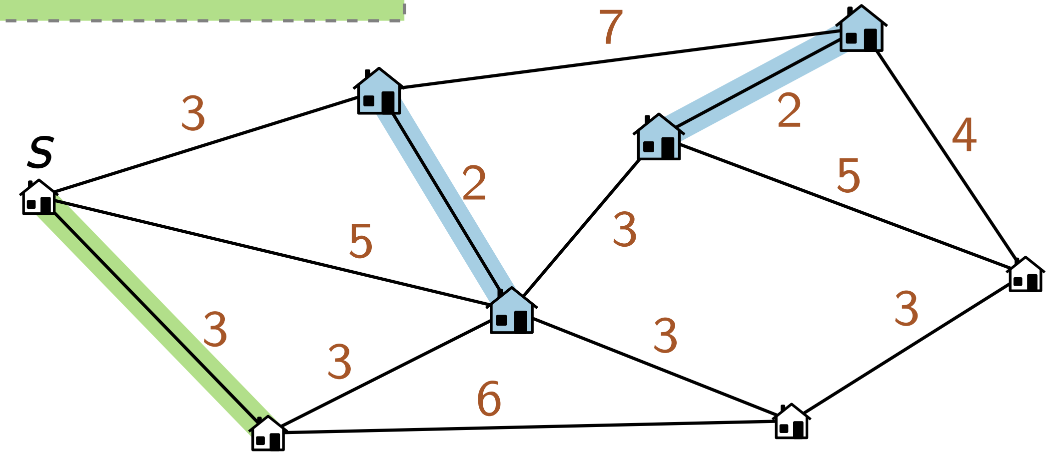
    $E' = E' \cup \{uv\}$

  else

Blaue Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{ Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

if $E' \cup \{uv\}$ enthält keinen Kreis **then**

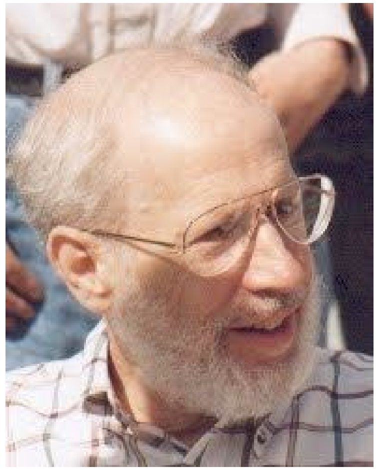
Färbe uv blau.

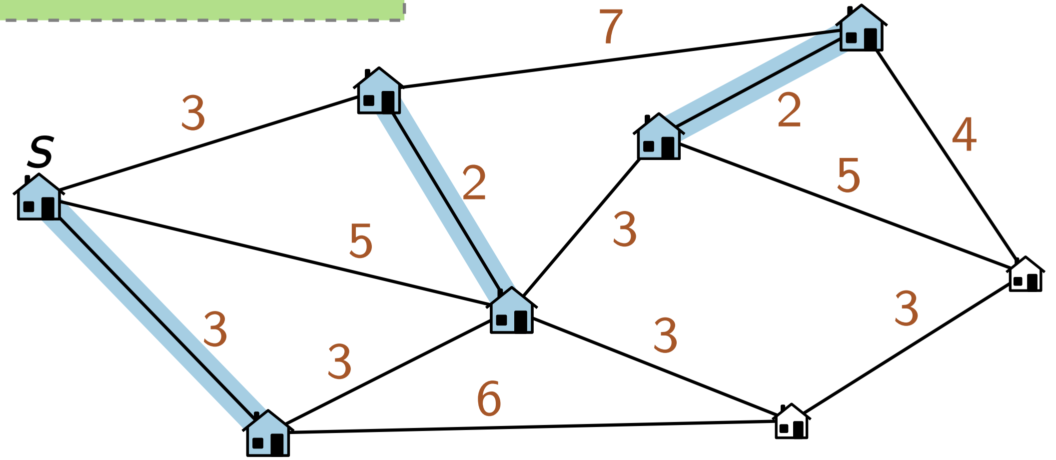
$E' = E' \cup \{uv\}$

Blaue Regel

else

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ

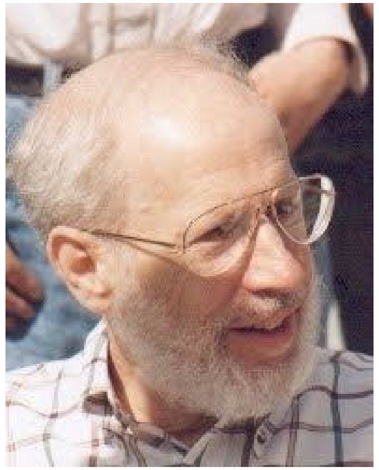


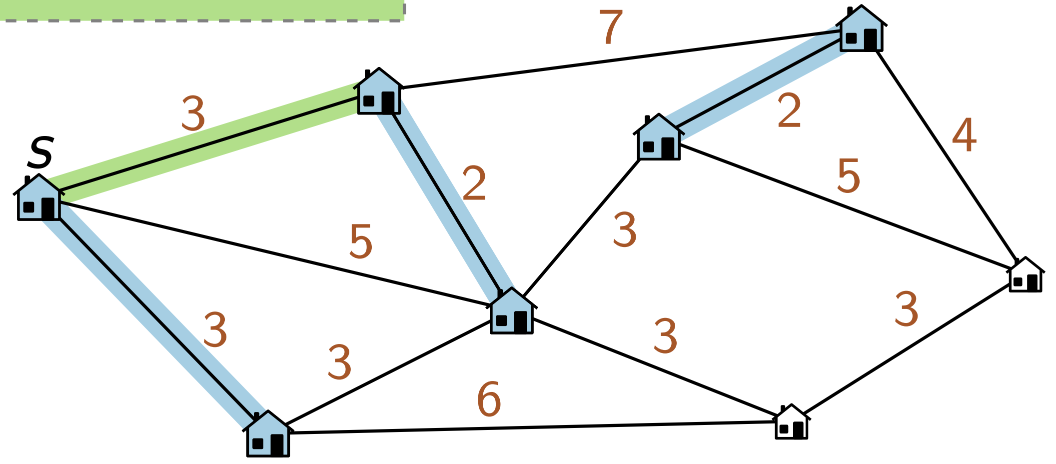


Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$
  $E' = \emptyset$
  Sortiere $E(G)$ nicht-absteigend nach Gewicht w.
  foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)
    if $E' \cup \{uv\}$ enthält keinen Kreis **then**
      Färbe uv blau. Blaue Regel
      $E' = E' \cup \{uv\}$
    else

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





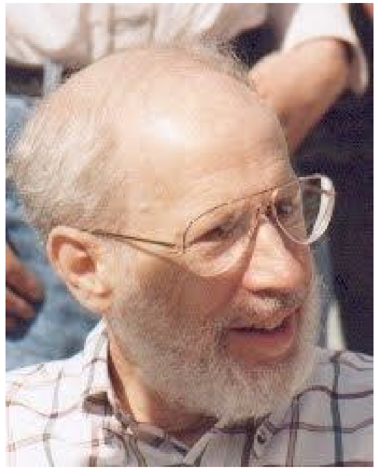
Der Algorithmus von Kruskal (1956)

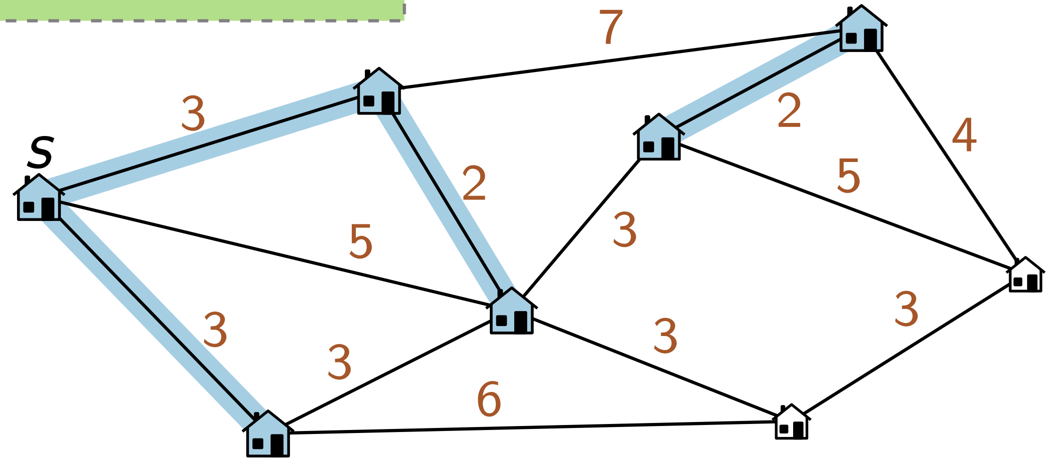
Kruskal(Graph G, Weights $w \colon E(G) \to \mathbb{R}_{\geq 0}$))

$E' = \emptyset$
Sortiere $E(G)$ nicht-absteigend nach Gewicht w.
foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)
  if $E' \cup \{uv\}$ enthält keinen Kreis **then**
    Färbe uv blau.
    $E' = E' \cup \{uv\}$
  else

Blaue Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{ Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

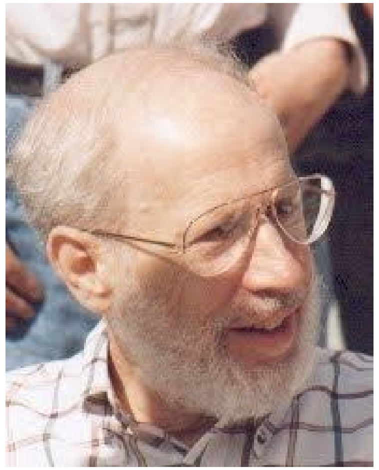
  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

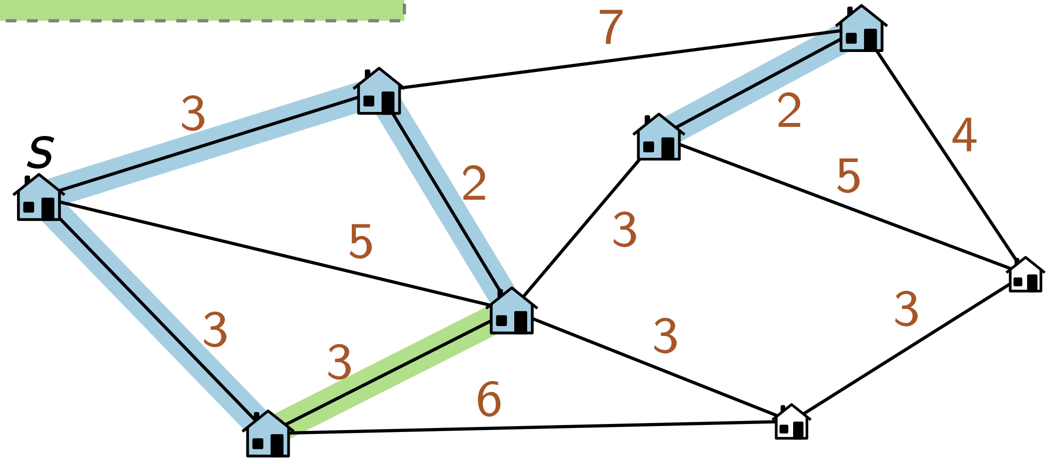
    Färbe uv blau. — Blaue Regel

    $E' = E' \cup \{uv\}$

  else

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ



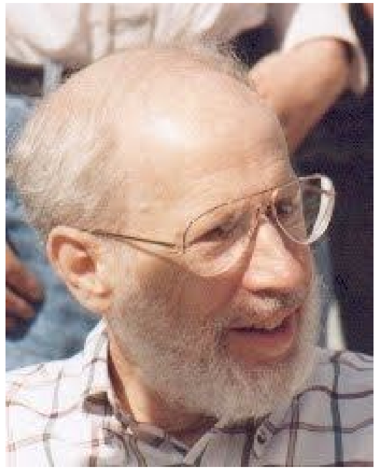


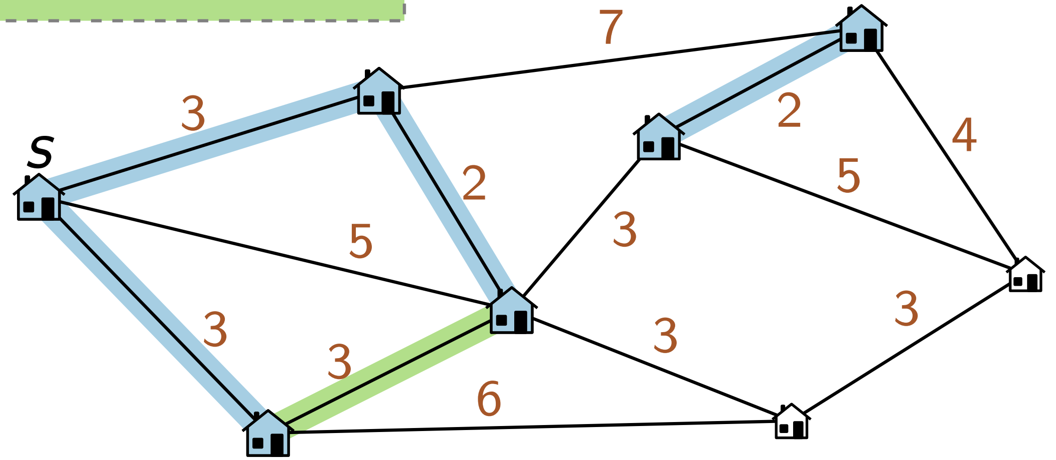
Der Algorithmus von Kruskal (1956)

Kruskal(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$)

```
E' = ∅
Sortiere E(G) nicht-absteigend nach Gewicht w.
foreach uv ∈ E(G) do   (in sortierter Reihenfolge)
    if E' ∪ {uv} enthält keinen Kreis then
        Färbe uv blau.                          Blaue Regel
        E' = E' ∪ {uv}
    else
        Färbe uv rot.                           Rote Regel
```

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

KRUSKAL(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$))

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

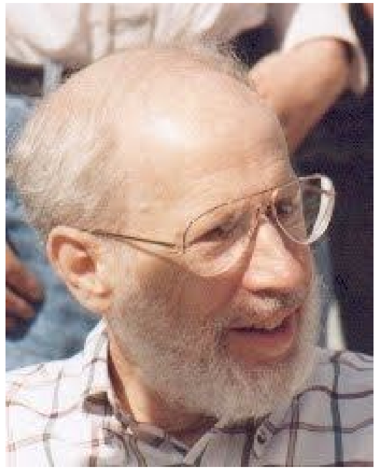
    Färbe uv blau. — Blaue Regel

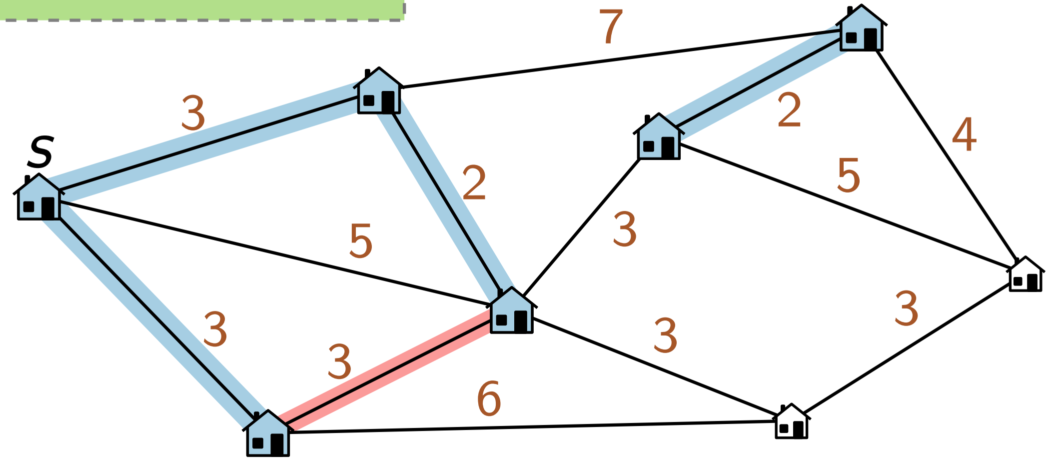
    $E' = E' \cup \{uv\}$

  else

    Färbe uv rot. — Rote Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

KRUSKAL(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$))

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

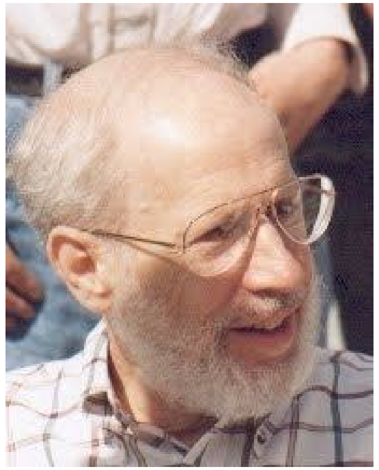
    Färbe uv blau. — Blaue Regel

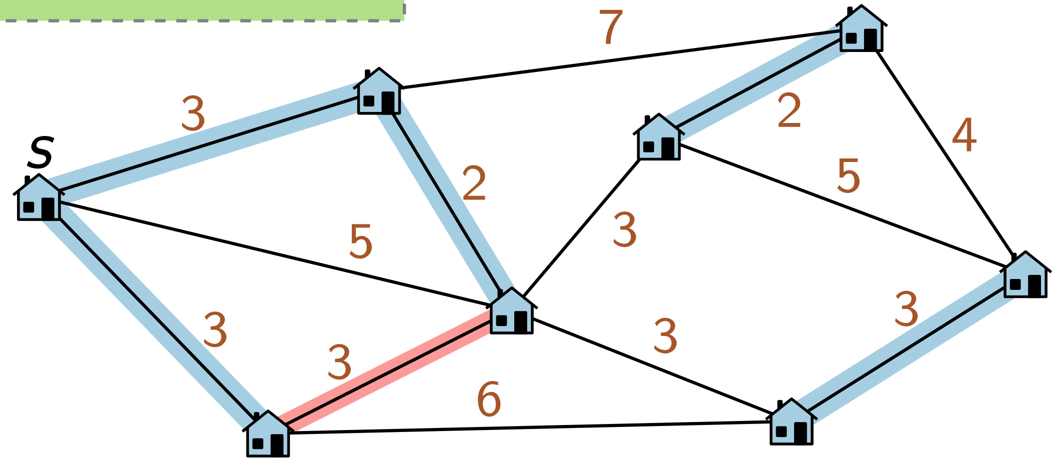
    $E' = E' \cup \{uv\}$

  else

    Färbe uv rot. — Rote Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

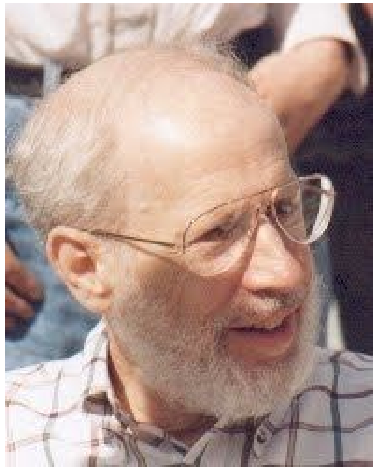
$\textsc{Kruskal}(\text{Graph } G, \text{ Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

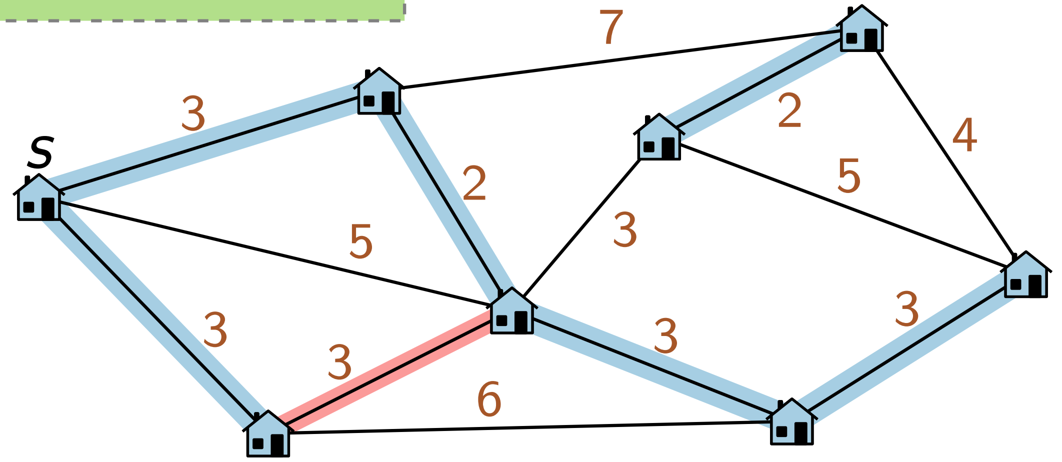
- $E' = \emptyset$
- Sortiere $E(G)$ nicht-absteigend nach Gewicht w.
- **foreach** $uv \in E(G)$ **do** (in sortierter Reihenfolge)
  - **if** $E' \cup \{uv\}$ enthält keinen Kreis **then**
    - Färbe uv blau.
    - $E' = E' \cup \{uv\}$

    Blaue Regel
  - **else**
    - Färbe uv rot.

    Rote Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{ Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

    Färbe uv blau.

    $E' = E' \cup \{uv\}$

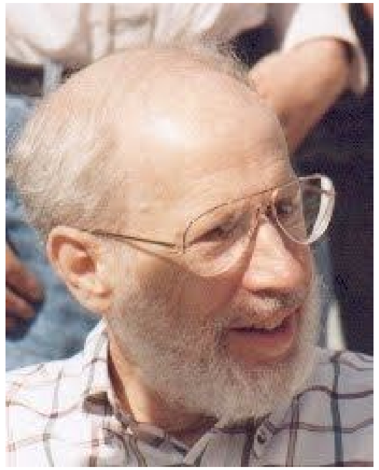
Blaue Regel

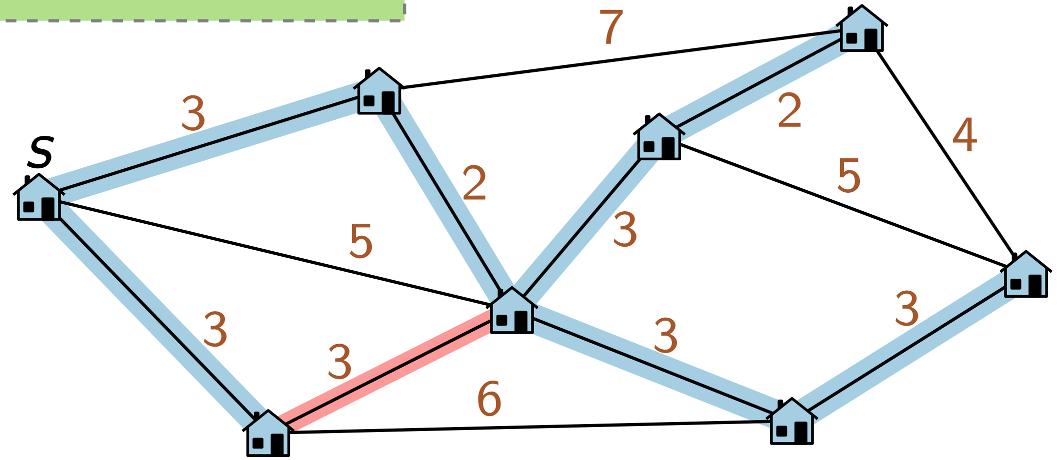
  else

    Färbe uv rot.

Rote Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ



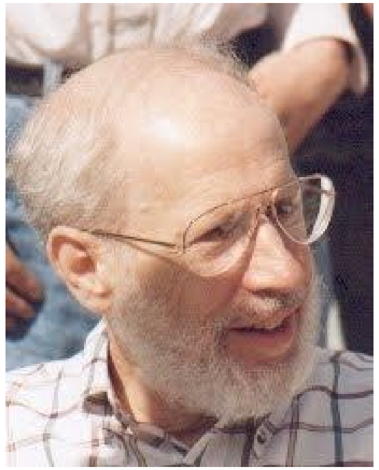


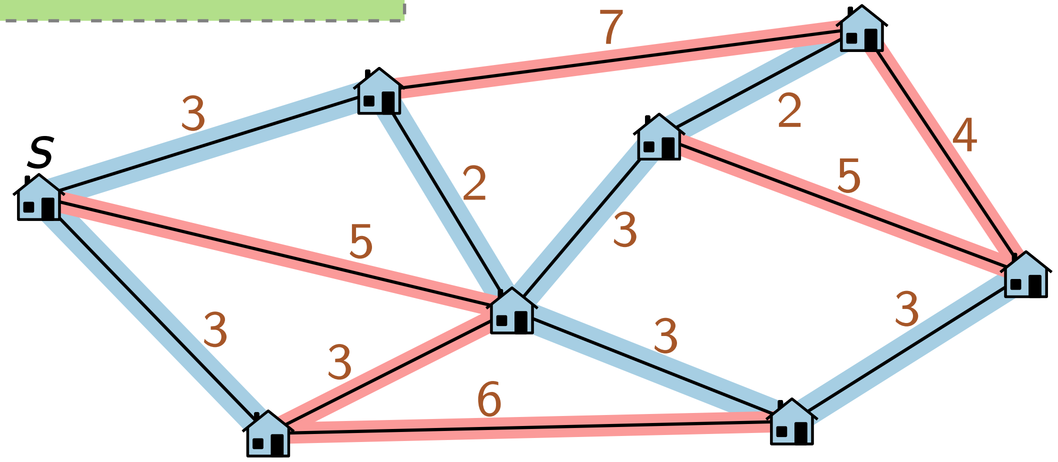
Der Algorithmus von Kruskal (1956)

KRUSKAL(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$))

$E' = \emptyset$
Sortiere $E(G)$ nicht-absteigend nach Gewicht w.
foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)
  if $E' \cup \{uv\}$ enthält keinen Kreis **then**
    Färbe uv blau. — Blaue Regel
    $E' = E' \cup \{uv\}$
  else
    Färbe uv rot. — Rote Regel

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ





Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

    Färbe uv blau. Blaue Regel

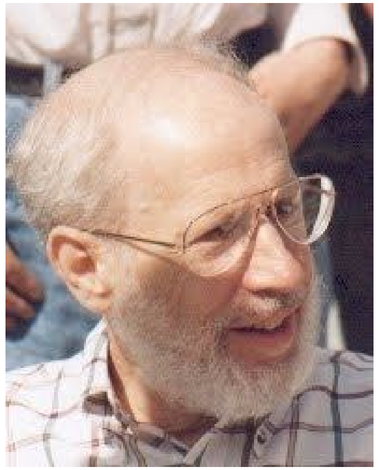
    $E' = E' \cup \{uv\}$

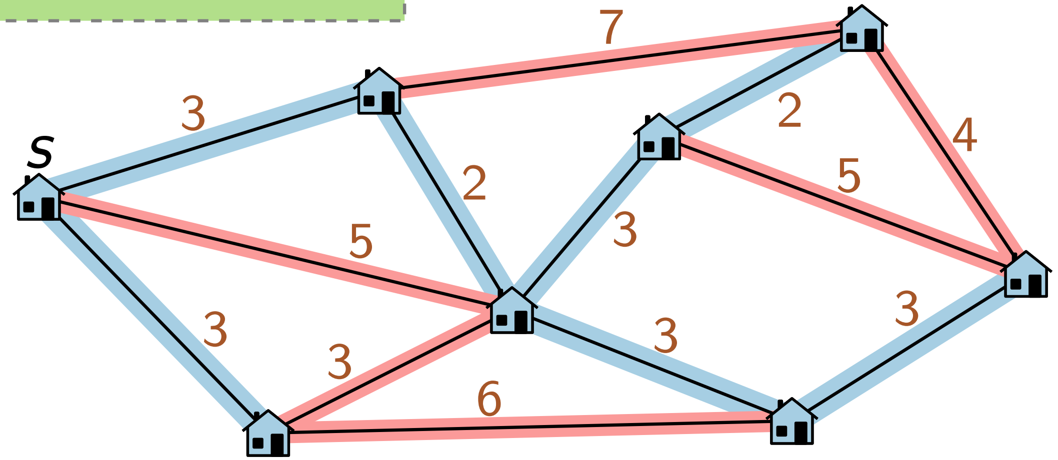
  else

    Färbe uv rot. Rote Regel

return E'

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ

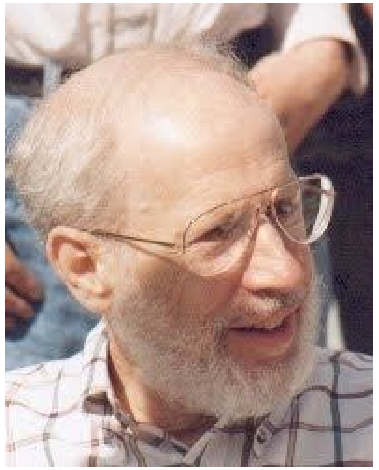




Der Algorithmus von Kruskal (1956)

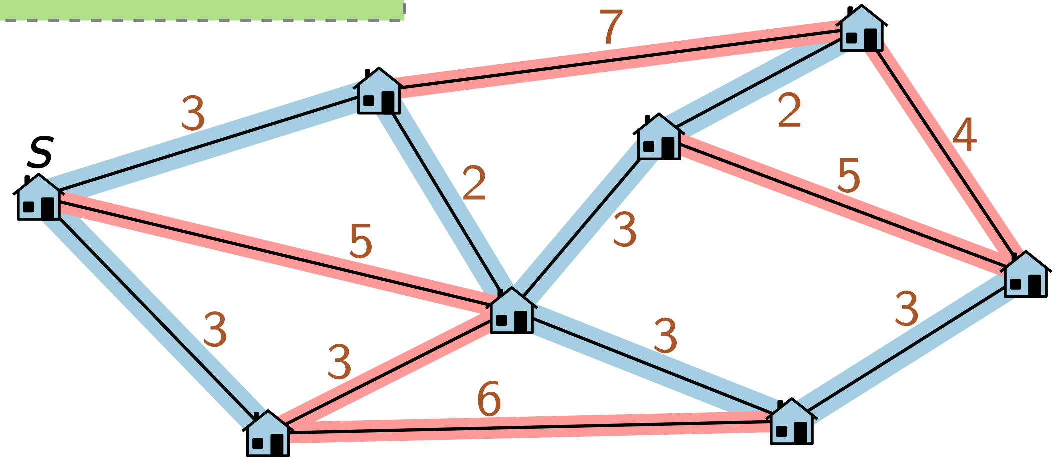
$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w \colon E(G) \to \mathbb{R}_{\geq 0})$

- $E' = \emptyset$
- Sortiere $E(G)$ nicht-absteigend nach Gewicht w.
- **foreach** $uv \in E(G)$ **do** (in sortierter Reihenfolge)
  - **if** $E' \cup \{uv\}$ enthält keinen Kreis **then**
    - Färbe uv blau. — Blaue Regel
    - $E' = E' \cup \{uv\}$
  - **else**
    - Färbe uv rot. — Rote Regel
- **return** E'



Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ

Laufzeit?



Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{ Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

    Färbe uv blau. — Blaue Regel

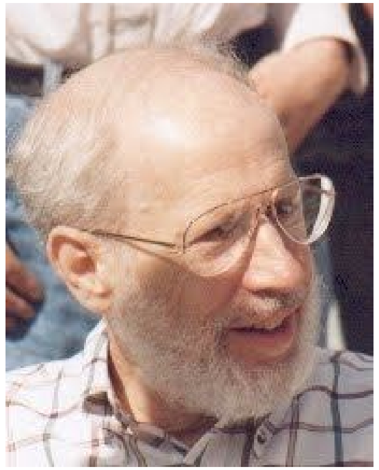
    $E' = E' \cup \{uv\}$

  else

    Färbe uv rot. — Rote Regel

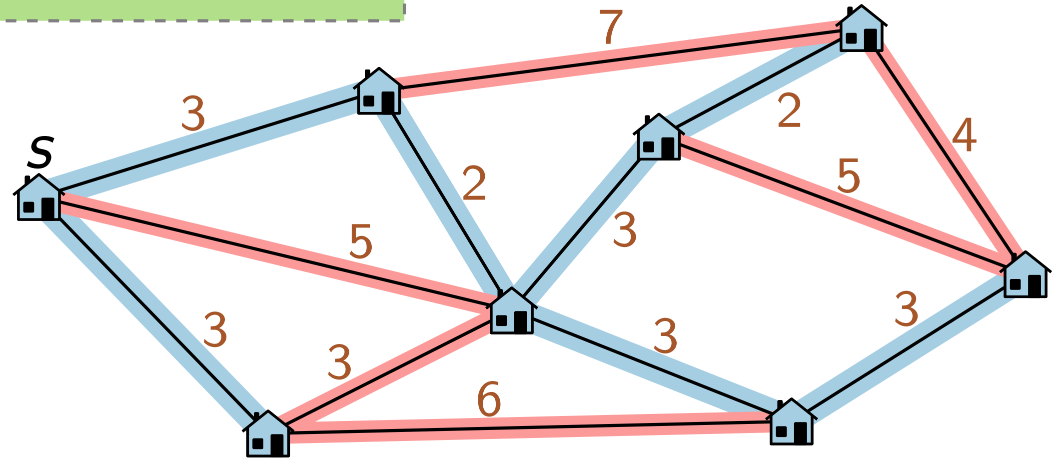
return E'

Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ



Laufzeit?

$\mathcal{O}(E \log V)$

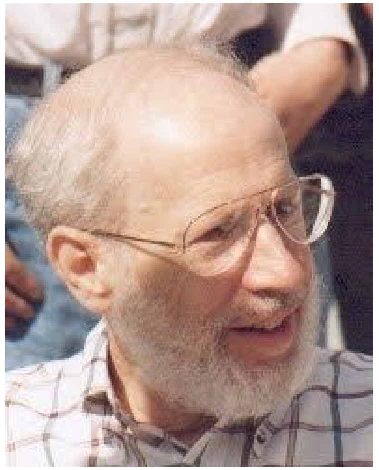


Der Algorithmus von Kruskal (1956)

$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

- $E' = \emptyset$
- Sortiere $E(G)$ nicht-absteigend nach Gewicht w.
- **foreach** $uv \in E(G)$ **do** (in sortierter Reihenfolge)
  - **if** $E' \cup \{uv\}$ enthält keinen Kreis **then**
    - Färbe uv blau. — Blaue Regel
    - $E' = E' \cup \{uv\}$
  - **else**
    - Färbe uv rot. — Rote Regel
- **return** E'

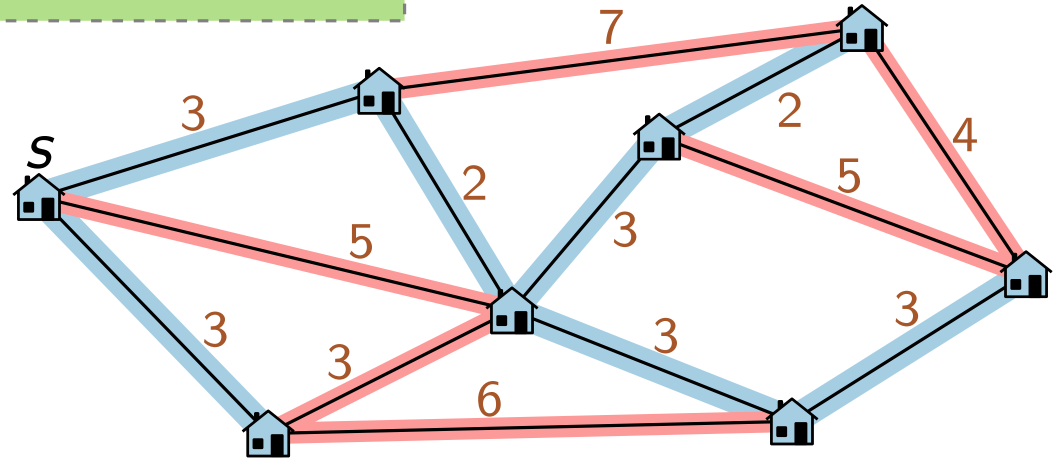
Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ



Laufzeit?

$\mathcal{O}(E \log V)$

$\mathcal{O}(E \cdot \alpha(V))$ falls vorsortiert



Der Algorithmus von Kruskal (1956)

KRUSKAL(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$)

$E' = \emptyset$

Sortiere $E(G)$ nicht-absteigend nach Gewicht w.

foreach $uv \in E(G)$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then**

    Färbe uv blau.   Blaue Regel

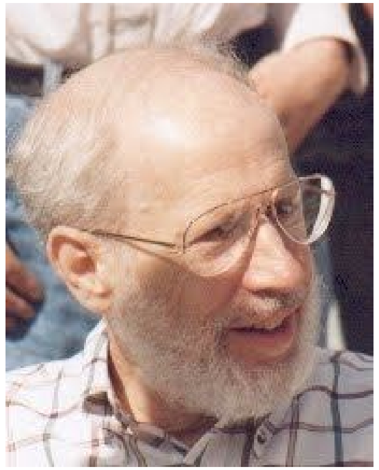
    $E' = E' \cup \{uv\}$

  else

    Färbe uv rot.   Rote Regel

return E'

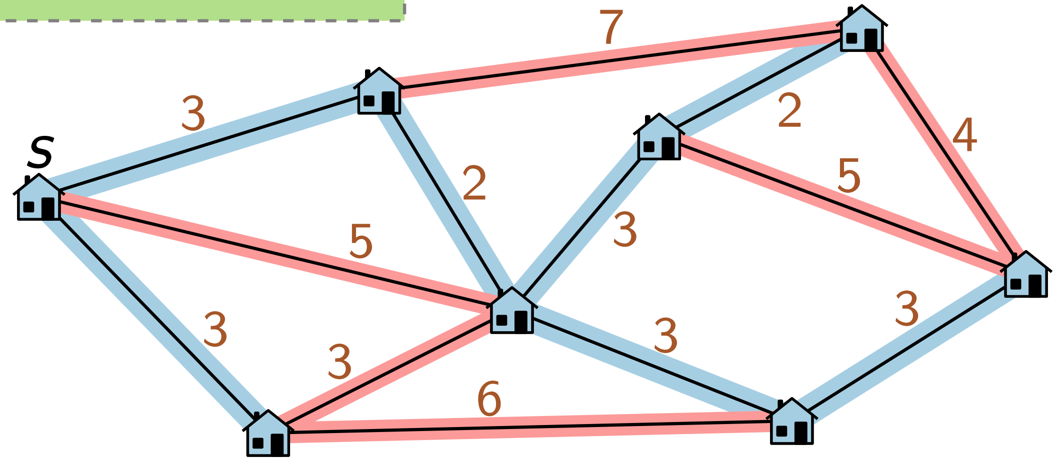
Joseph Bernard Kruskal, Jr.
∗ 1928 New York, NY
† 2010 Maplewood, NJ



Laufzeit?

$\mathcal{O}(E \log V)$

$\mathcal{O}(E \cdot \alpha(V))$ falls vorsortiert



UnionFind-Datenstruktur

UnionFind-Datenstruktur

Datenstruktur für **halbdynamische Mengen**

UnionFind-Datenstruktur

Datenstruktur für **halbdynamische Mengen**

(wachsen nur, schrumpfen nicht)

UnionFind-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X.

UnionFind-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

UNIONFIND-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

UnionFind-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

Make(Element x)

Find(Element x)

Union(Elem. x, Elem. y)

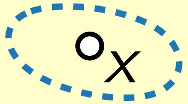
UnionFind-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

Make(Element x) legt die Menge $\{x\}$ an.



Find(Element x)

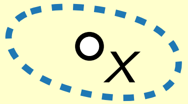
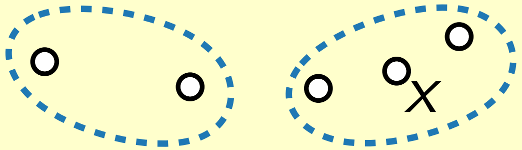
Union(Elem. x, Elem. y)

UnionFind-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

Make(Element x) 	legt die Menge $\{x\}$ an.
Find(Element x) 	liefert (Zeiger auf) die Menge zurück, die momentan x enthält.
Union(Elem. x, Elem. y)	

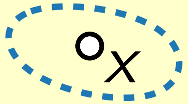
UNIONFIND-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

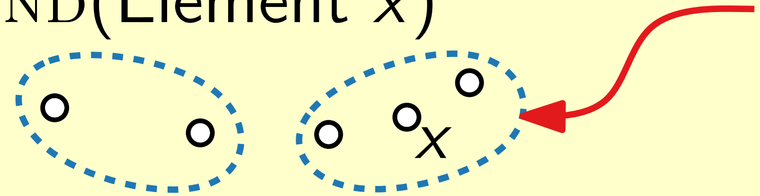
Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

MAKE(Element x) legt die Menge $\{x\}$ an.



FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



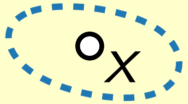
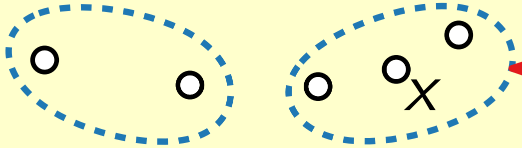
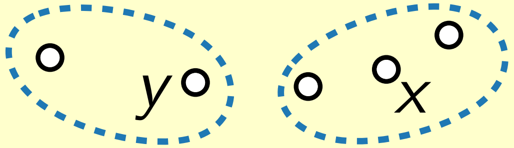
UNION(Elem. x, Elem. y)

UNIONFIND-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

MAKE(Element x) 	legt die Menge $\{x\}$ an.
FIND(Element x) 	liefert (Zeiger auf) die Menge zurück, die momentan x enthält.
UNION(Elem. x, Elem. y) 	vereinigt die Mengen, die momentan x und y enthalten.

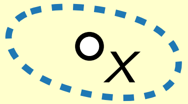
UnionFind-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

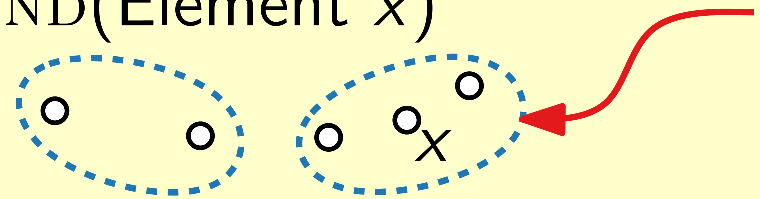
Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

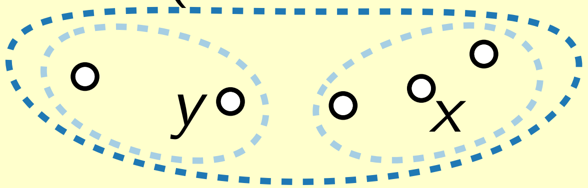
Make(Element x) legt die Menge $\{x\}$ an.



Find(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



Union(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

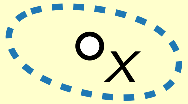
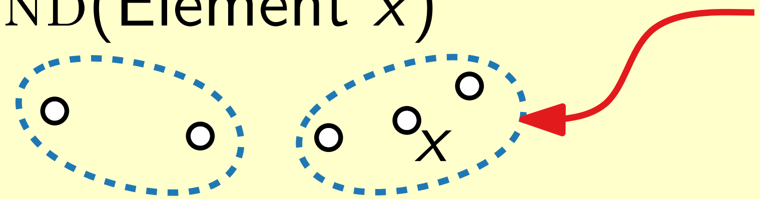
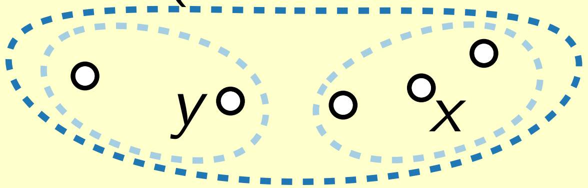


UnionFind-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

Make(Element x) 	legt die Menge $\{x\}$ an.
Find(Element x) 	liefert (Zeiger auf) die Menge zurück, die momentan x enthält.
Union(Elem. x, Elem. y) 	vereinigt die Mengen, die momentan x und y enthalten.

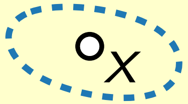
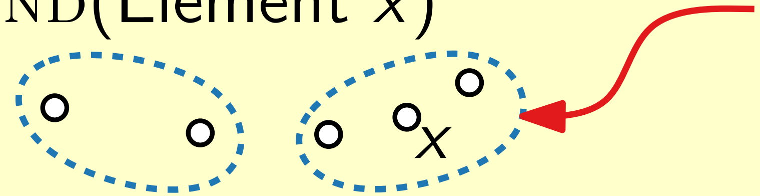
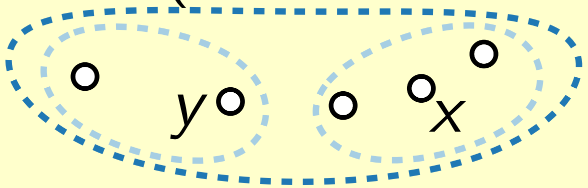
Beispiel.

UnionFind-Datenstruktur

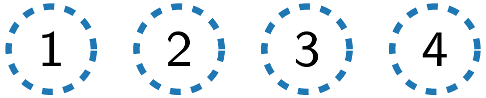
Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

Make(Element x) 	legt die Menge $\{x\}$ an.
Find(Element x) 	liefert (Zeiger auf) die Menge zurück, die momentan x enthält.
Union(Elem. x, Elem. y) 	vereinigt die Mengen, die momentan x und y enthalten.

Beispiel.

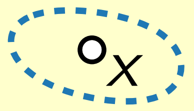
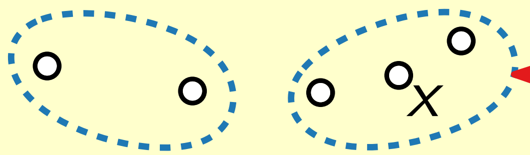
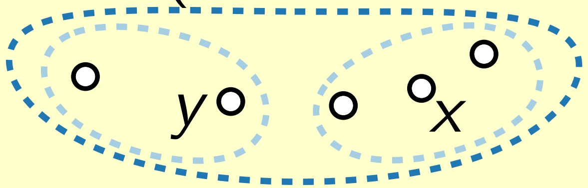


UNIONFIND-Datenstruktur

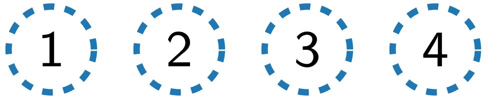
Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

MAKE(Element x) 	legt die Menge $\{x\}$ an.
FIND(Element x) 	liefert (Zeiger auf) die Menge zurück, die momentan x enthält.
UNION(Elem. x, Elem. y) 	vereinigt die Mengen, die momentan x und y enthalten.

Beispiel.



- UNION$(1, 2)$

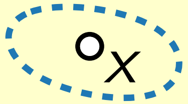
UNIONFIND-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

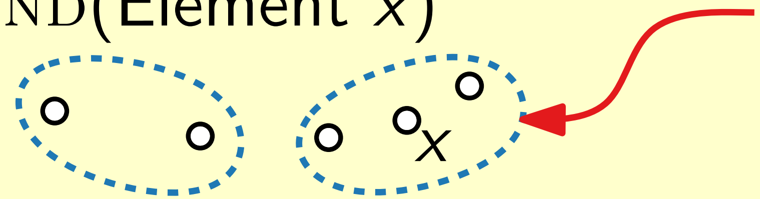
Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

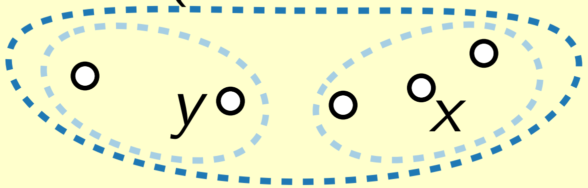
MAKE(Element x) legt die Menge $\{x\}$ an.



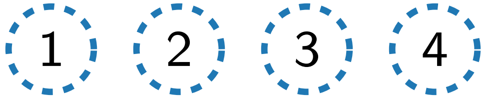
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



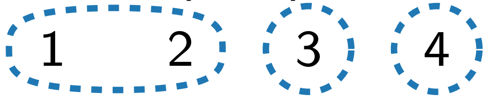
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



Beispiel.



- UNION$(1, 2)$

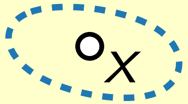
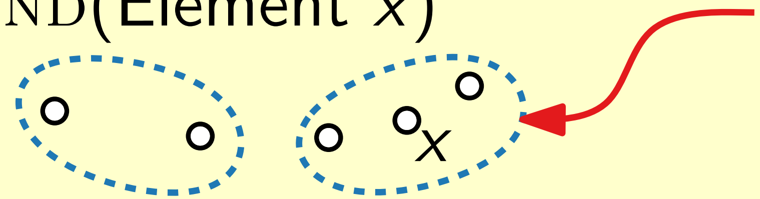
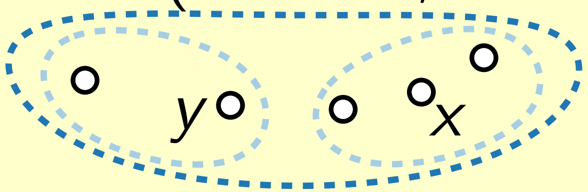


UnionFind-Datenstruktur

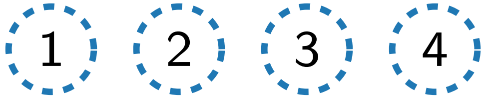
Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

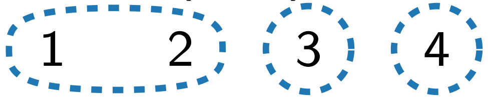
Drei Operationen:

Make(Element x) 	legt die Menge $\{x\}$ an.
Find(Element x) 	liefert (Zeiger auf) die Menge zurück, die momentan x enthält.
Union(Elem. x, Elem. y) 	vereinigt die Mengen, die momentan x und y enthalten.

Beispiel.



- Union$(1, 2)$



- Union$(2, 3)$

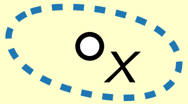
UNIONFIND-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

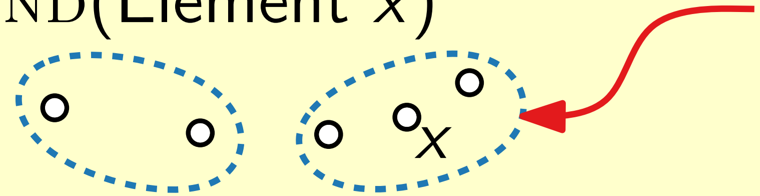
Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

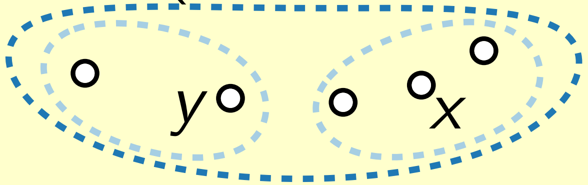
MAKE(Element x) legt die Menge $\{x\}$ an.



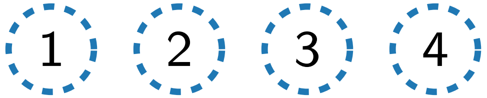
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



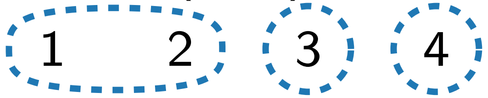
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



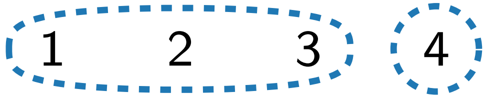
Beispiel.



- UNION$(1, 2)$



- UNION$(2, 3)$



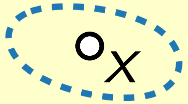
UnionFind-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

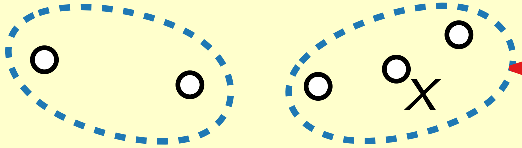
Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

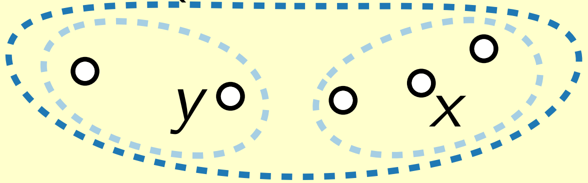
Make(Element x) — legt die Menge $\{x\}$ an.



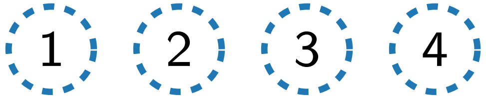
Find(Element x) — liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



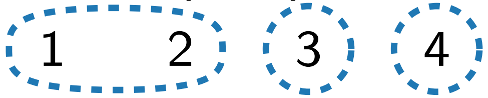
Union(Elem. x, Elem. y) — vereinigt die Mengen, die momentan x und y enthalten.



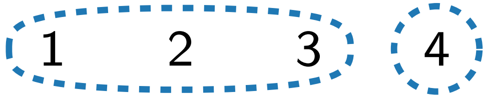
Beispiel.



- Union$(1, 2)$



- Union$(2, 3)$



- Find(1) = Find(3)?

UNIONFIND-Datenstruktur

Datenstruktur für **halbdynamische Mengen**

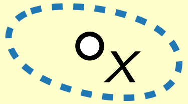
(wachsen nur, schrumpfen nicht)

Die halbdynamischen Mengen zerlegen immer eine Grundmenge X.

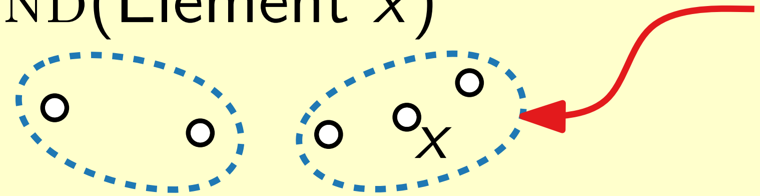
(bei Kruskal: $X = V$)

Drei Operationen:

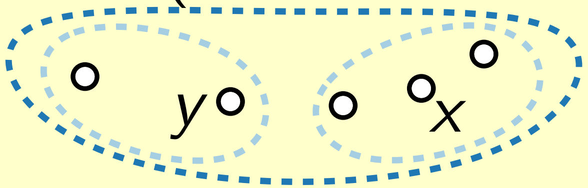
MAKE(Element x) — legt die Menge $\{x\}$ an.



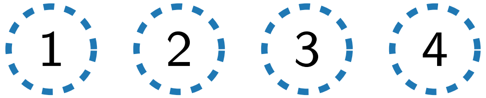
FIND(Element x) — liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



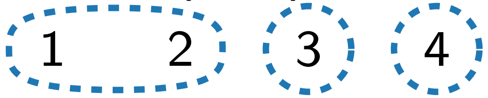
UNION(Elem. x, Elem. y) — vereinigt die Mengen, die momentan x und y enthalten.



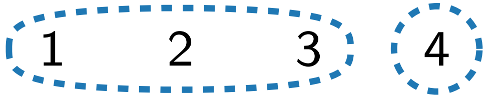
Beispiel.



- UNION$(1, 2)$



- UNION$(2, 3)$



- FIND(1) = FIND(3)?
  ➡ `true`

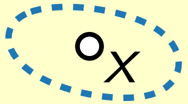
UNIONFIND-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

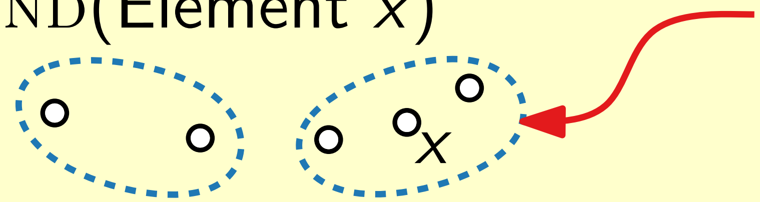
Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

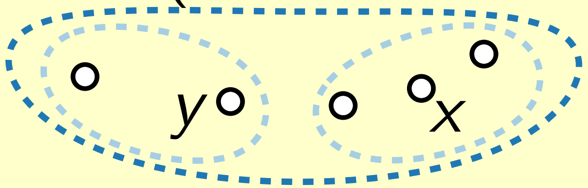
MAKE(Element x) — legt die Menge $\{x\}$ an.



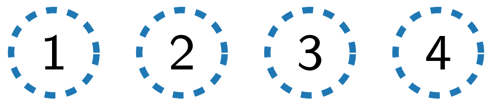
FIND(Element x) — liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



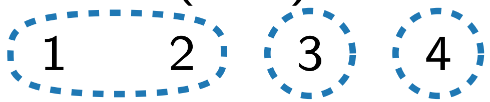
UNION(Elem. x, Elem. y) — vereinigt die Mengen, die momentan x und y enthalten.



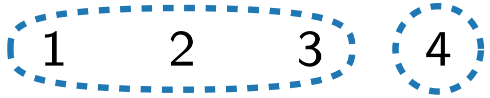
Beispiel.



- UNION$(1, 2)$



- UNION$(2, 3)$



- FIND(1) = FIND(3)?
  ➡ `true`
- FIND(2) = FIND(4)?

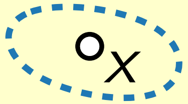
UNIONFIND-Datenstruktur

Datenstruktur für **halbdynamische Mengen** (wachsen nur, schrumpfen nicht)

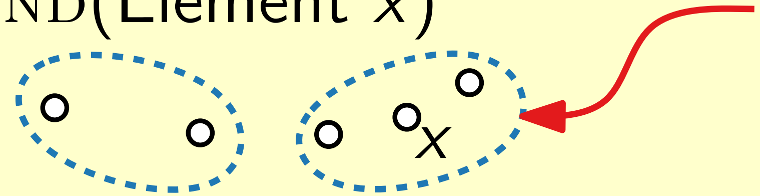
Die halbdynamischen Mengen zerlegen immer eine Grundmenge X. (bei Kruskal: $X = V$)

Drei Operationen:

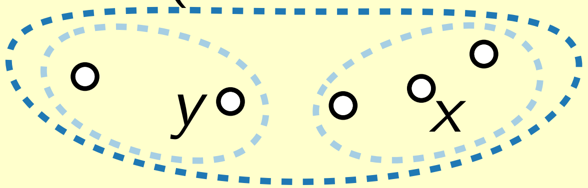
MAKE(Element x) legt die Menge $\{x\}$ an.



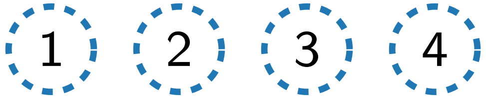
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



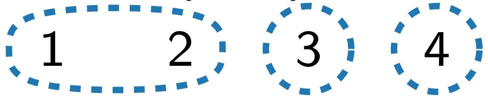
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



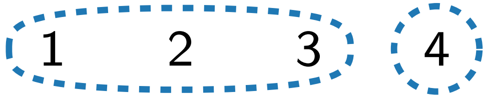
Beispiel.



- UNION$(1, 2)$



- UNION$(2, 3)$



- FIND(1) = FIND(3)?
  ➡ `true`
- FIND(2) = FIND(4)?
  ➡ `false`

Anpassung Kruskal

Kruskal(Graph G, Weights $w\colon E(G) \to \mathbb{R}_{\geq 0}$)

$E' = \emptyset$

Sortiere E nicht-absteigend nach Gewicht w

foreach $uv \in E$ **do** (in sortierter Reihenfolge)

if $E' \cup \{uv\}$ enthält keinen Kreis **then**

Färbe uv blau — Blaue Regel

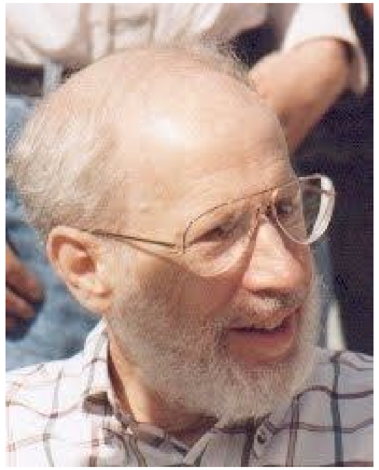
$E' = E' \cup \{uv\}$

else

Färbe uv rot — Rote Regel

return E'

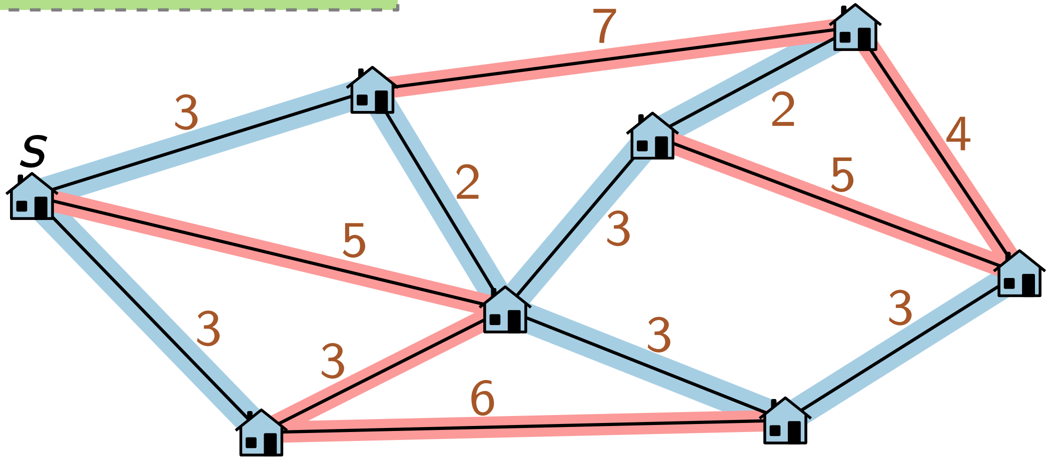
Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ



Laufzeit?

$\mathcal{O}(E \log V)$

$\mathcal{O}(E \cdot \alpha(V))$ falls vorsortiert



Anpassung Kruskal

$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

  $E' = \emptyset$ $\forall v \in V(G) : \textsc{Make}(v)$

  Sortiere E nicht-absteigend nach Gewicht w

  foreach $uv \in E$ **do**   (in sortierter Reihenfolge)

    if $E' \cup \{uv\}$ enthält keinen Kreis **then**

      Färbe uv blau   Blaue Regel

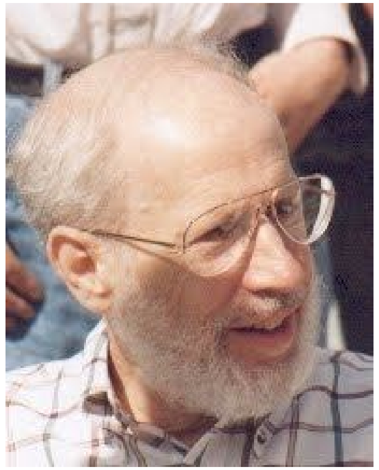
      $E' = E' \cup \{uv\}$

    else

      Färbe uv rot   Rote Regel

  return E'

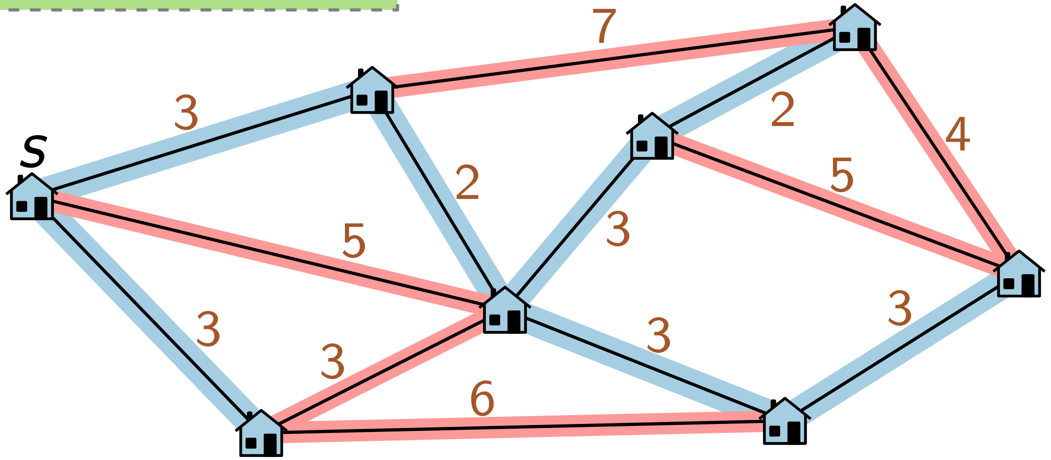
Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ



Laufzeit?

$\mathcal{O}(E \log V)$

$\mathcal{O}(E \cdot \alpha(V))$ falls vorsortiert



Anpassung Kruskal

$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w \colon E(G) \to \mathbb{R}_{\geq 0}))$

$E' = \emptyset$ $\quad\quad$ $\forall v \in V(G) : \textsc{Make}(v)$

Sortiere E nicht-absteigend nach Gewicht w

foreach $uv \in E$ **do** $\quad$ (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then** $\quad\quad$ **if** $\textsc{Find}(u) \neq \textsc{Find}(v)$

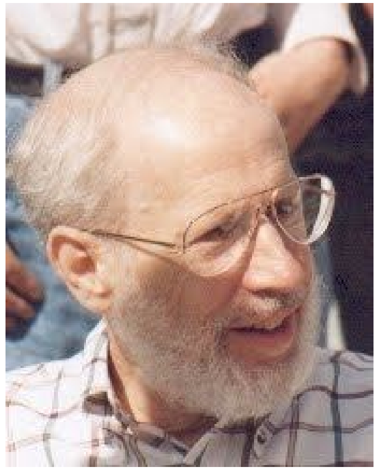
    Färbe uv blau $\quad\quad$ Blaue Regel

    $E' = E' \cup \{uv\}$

  else

    Färbe uv rot $\quad\quad$ Rote Regel

return E'

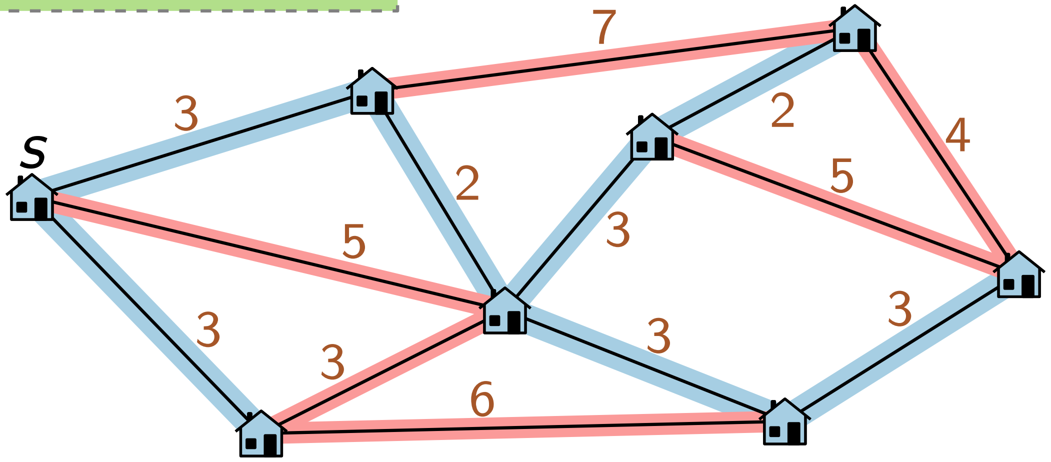


Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ

Laufzeit?

$\mathcal{O}(E \log V)$

$\mathcal{O}(E \cdot \alpha(V))$ falls vorsortiert



Anpassung Kruskal

$\textsc{Kruskal}(\text{Graph } G, \text{Weights } w\colon E(G) \to \mathbb{R}_{\geq 0})$

$E' = \emptyset$ $\qquad$ $\forall v \in V(G) : \textsc{Make}(v)$

Sortiere E nicht-absteigend nach Gewicht w

foreach $uv \in E$ **do** (in sortierter Reihenfolge)

  if $E' \cup \{uv\}$ enthält keinen Kreis **then** $\qquad$ **if** $\textsc{Find}(u) \neq \textsc{Find}(v)$

    Färbe uv blau $\qquad$ Blaue Regel

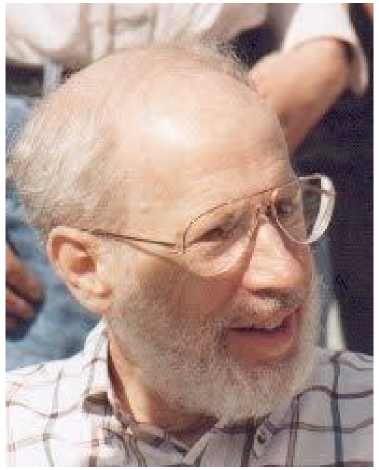
    $E' = E' \cup \{uv\}$ $\qquad$ $\textsc{Union}(u, v)$

  else

    Färbe uv rot $\qquad$ Rote Regel

return E'

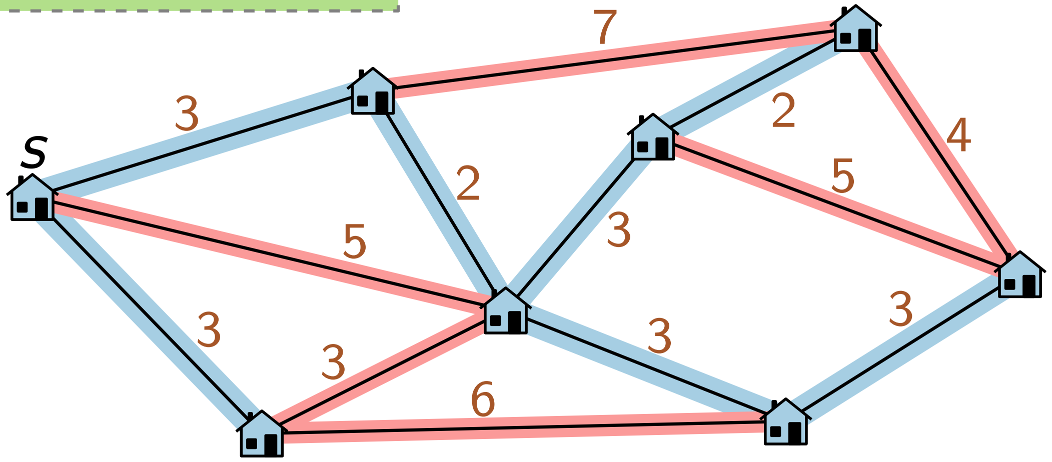
Joseph Bernard Kruskal, Jr.
* 1928 New York, NY
† 2010 Maplewood, NJ



Laufzeit?

$\mathcal{O}(E \log V)$

$\mathcal{O}(E \cdot \boxed{\alpha(V)})$ falls vorsortiert



Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

Realisierung Union-Find-Datenstruktur

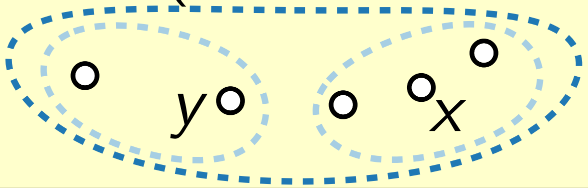
Baumstruktur für jede Menge

Beispiel.

Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

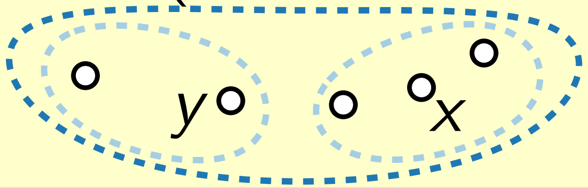


Beispiel.

Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



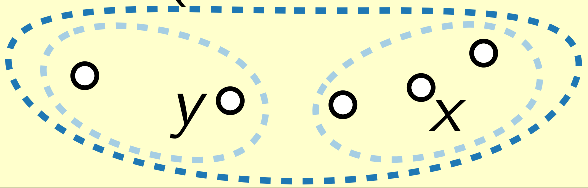
Beispiel.

- UNION$(1, 2)$

Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:

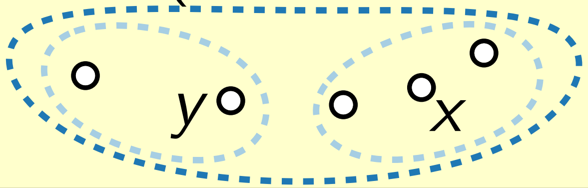
Beispiel.

- UNION(1, 2)

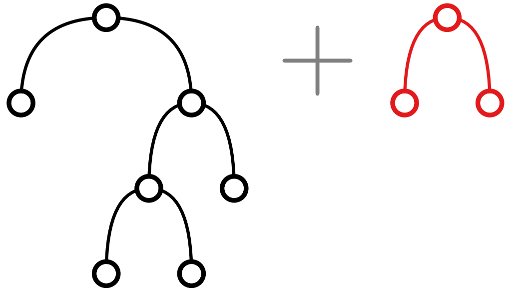
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



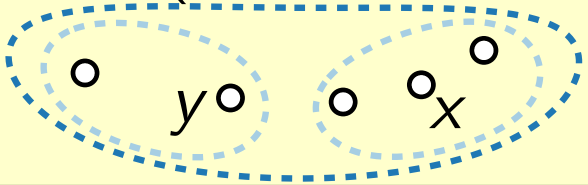
Beispiel.

- UNION$(1, 2)$

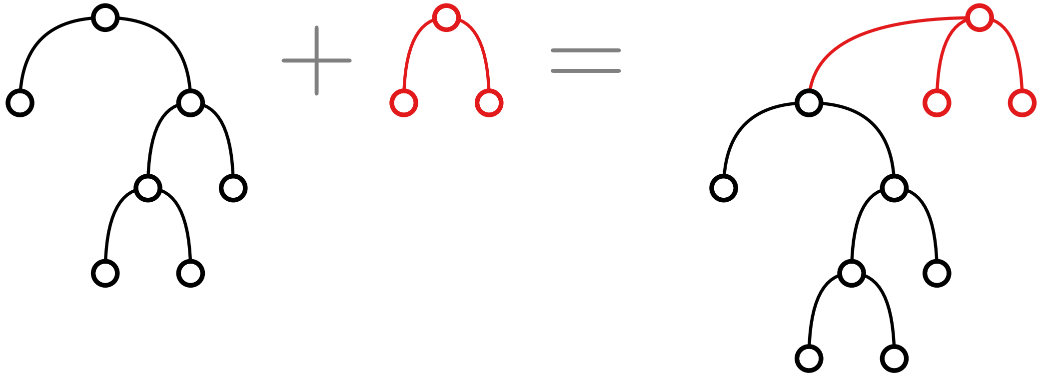
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



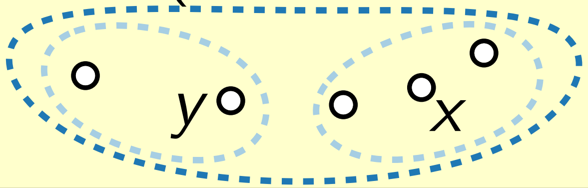
Beispiel.

- UNION$(1, 2)$

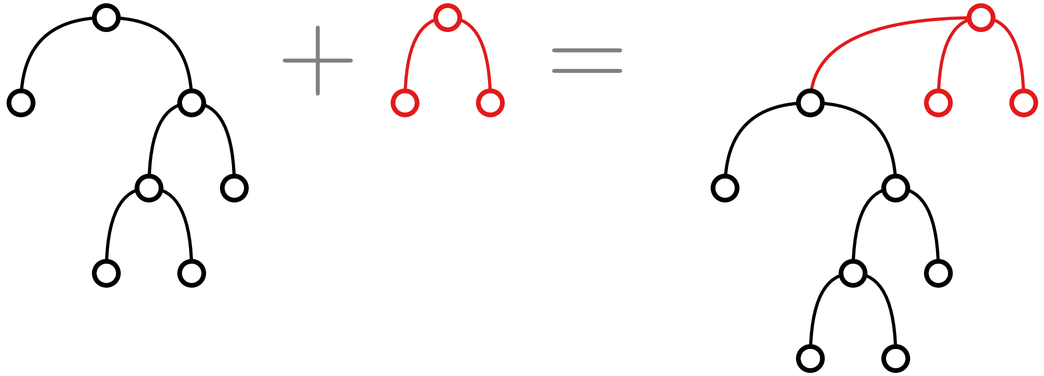
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

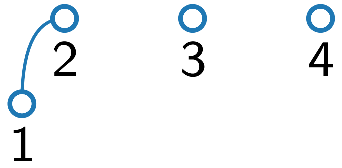


➡ Hänge einen Baum an den anderen:



Beispiel.

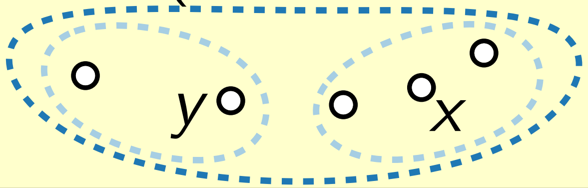
- UNION$(1, 2)$



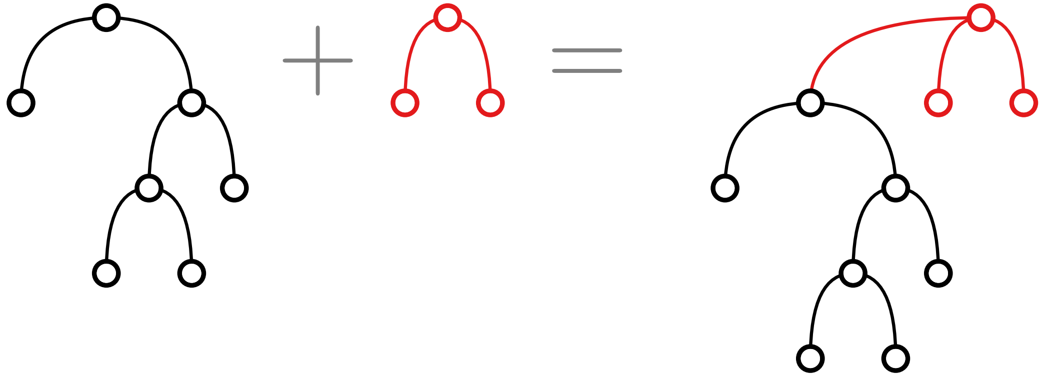
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

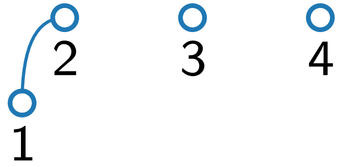


➡ Hänge einen Baum an den anderen:



Beispiel.

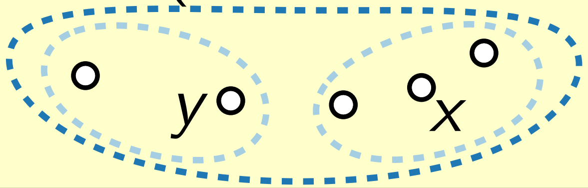
- UNION(1, 2)



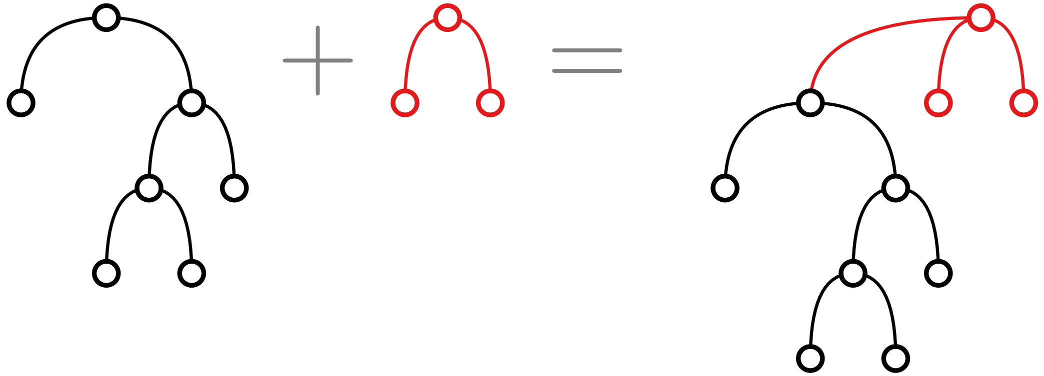
- UNION(2, 3)

Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

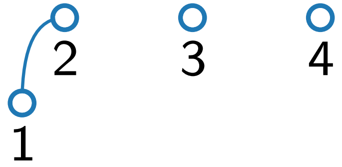
$\textsc{Union}(\text{Elem. } x, \text{Elem. } y)$ 	vereinigt die Mengen, die momentan x und y enthalten.

➡ Hänge einen Baum an den anderen:

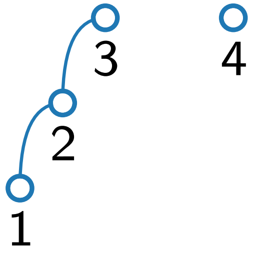


Beispiel.

- $\textsc{Union}(1, 2)$



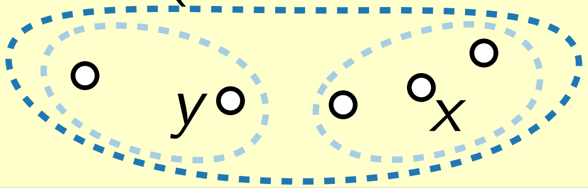
- $\textsc{Union}(2, 3)$



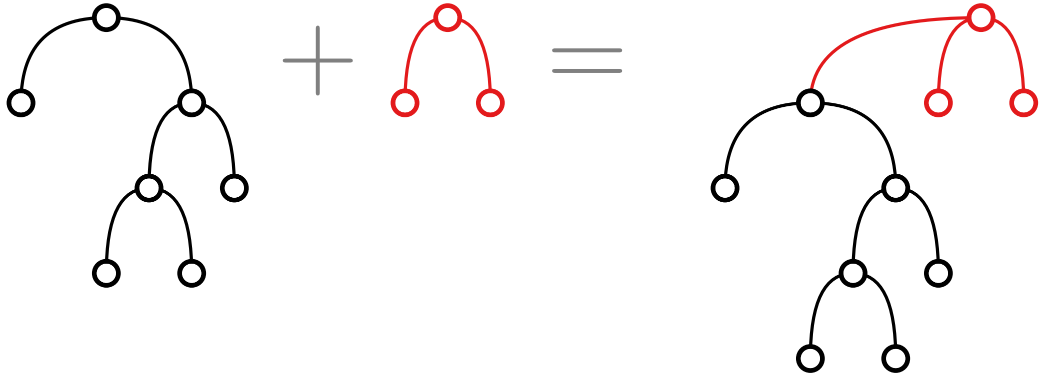
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

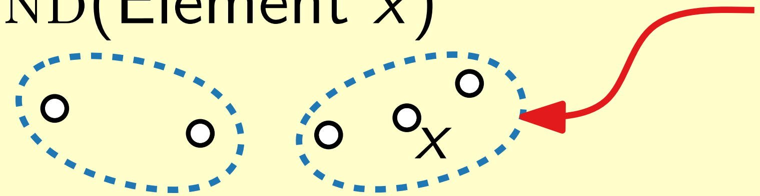
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:

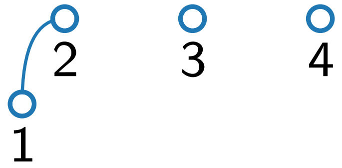


FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.

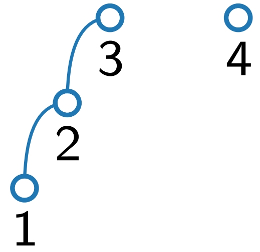


Beispiel.

- UNION(1, 2)



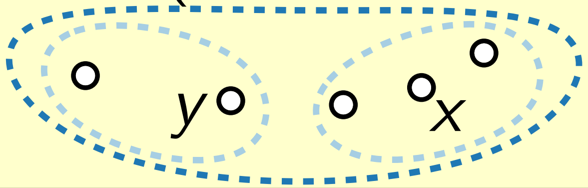
- UNION(2, 3)



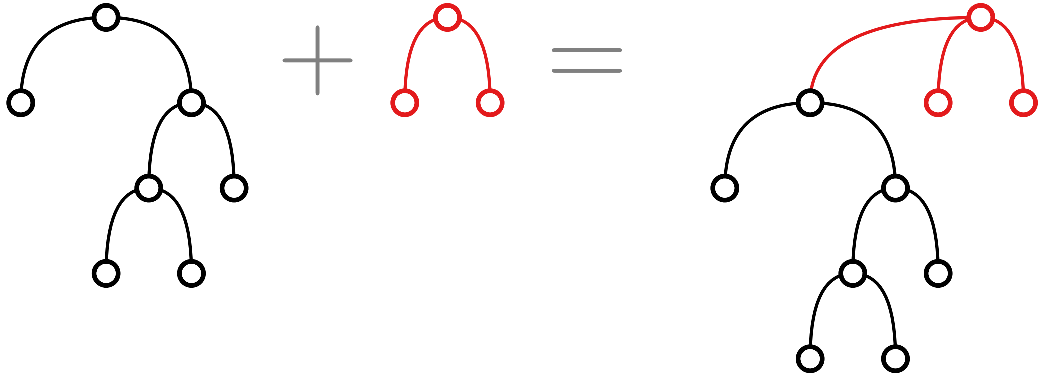
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

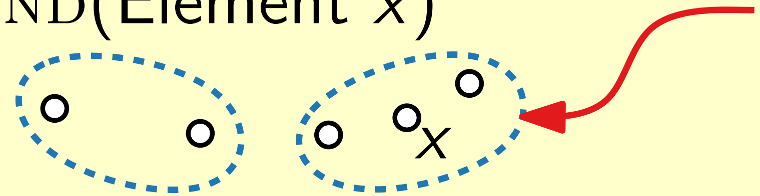
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



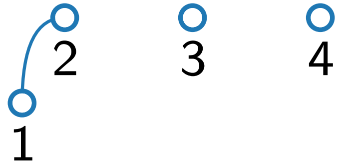
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



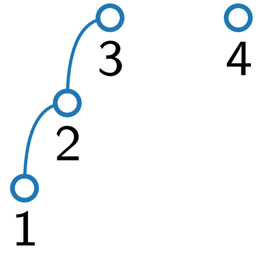
➡ Laufe zur Wurzel, gib Wurzel zurück.

Beispiel.

- UNION(1, 2)



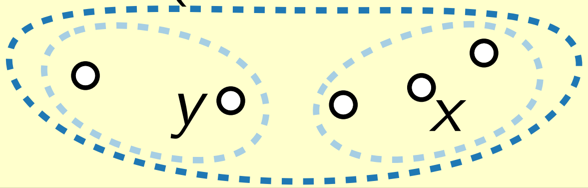
- UNION(2, 3)



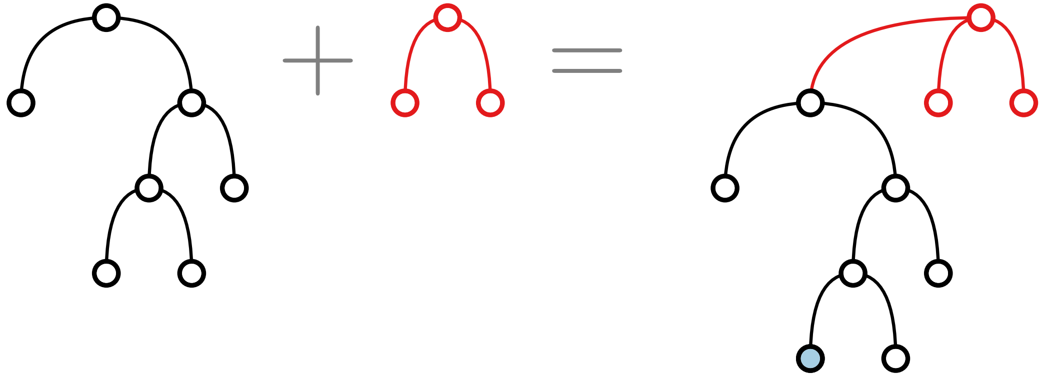
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

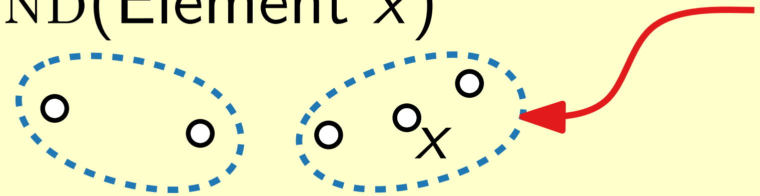
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



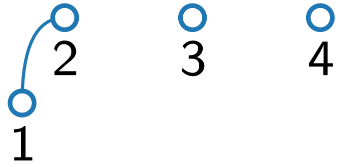
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



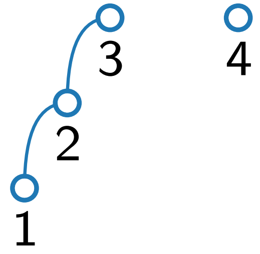
➡ Laufe zur Wurzel, gib Wurzel zurück.

Beispiel.

- UNION(1, 2)



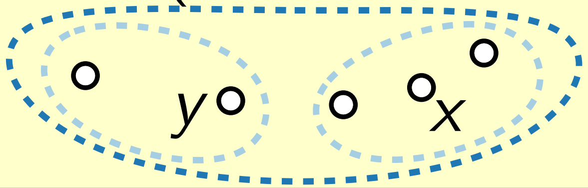
- UNION(2, 3)



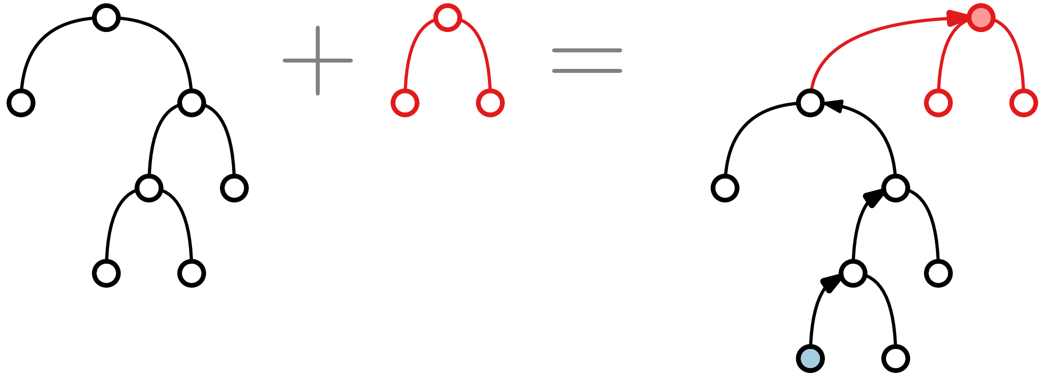
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

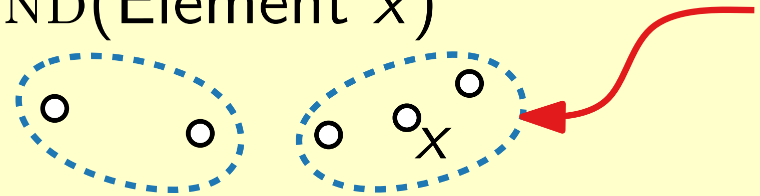
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



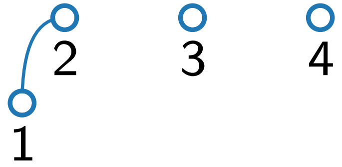
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



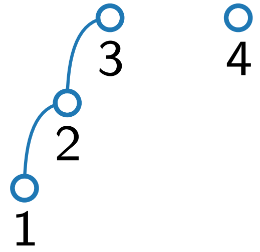
➡ Laufe zur Wurzel, gib Wurzel zurück.

Beispiel.

- UNION(1, 2)



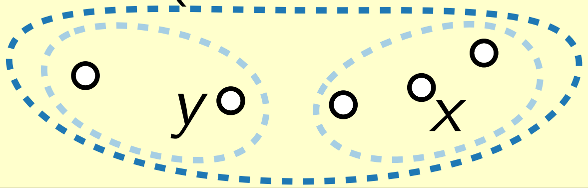
- UNION(2, 3)



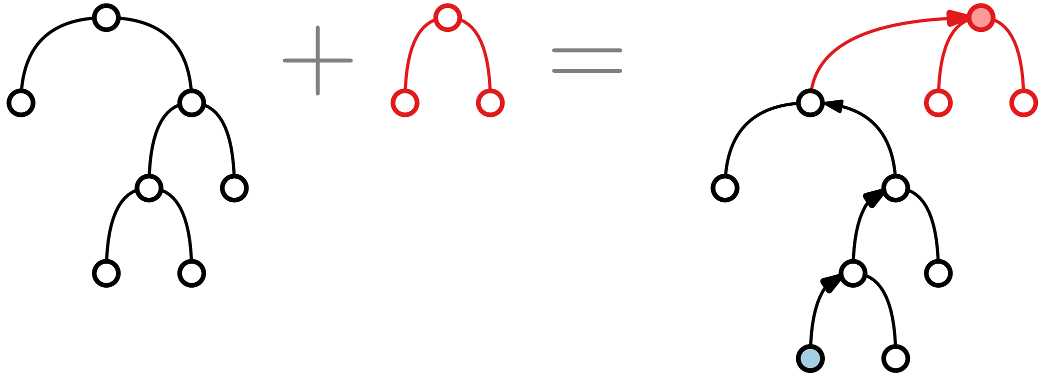
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

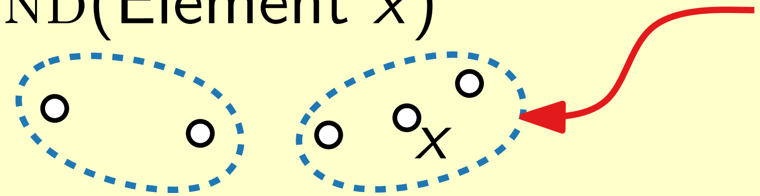
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



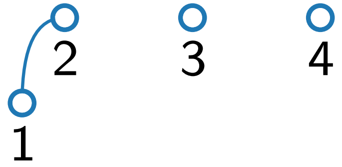
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



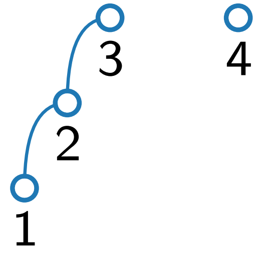
➡ Laufe zur Wurzel, gib Wurzel zurück.

Beispiel.

- UNION(1, 2)



- UNION(2, 3)

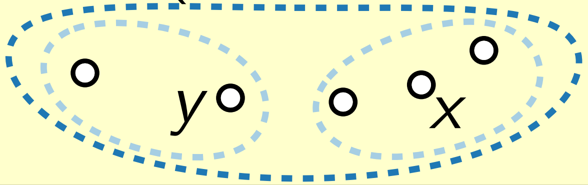


- FIND(1)?

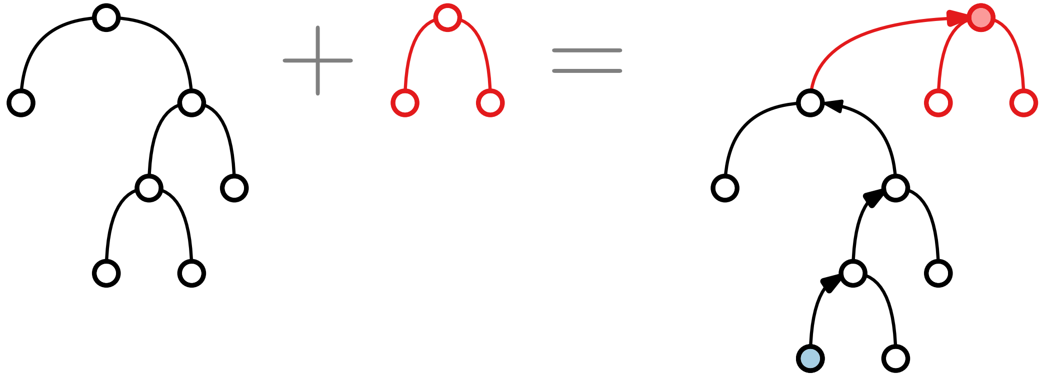
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

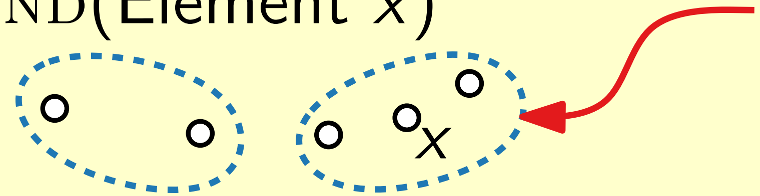
UNION(Elem. x, Elem. y)	vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



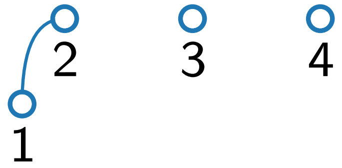
FIND(Element x)	liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



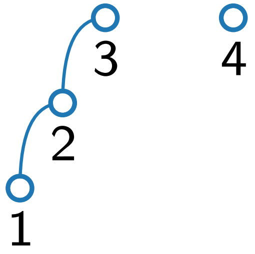
➡ Laufe zur Wurzel, gib Wurzel zurück.

Beispiel.

- UNION(1, 2)



- UNION(2, 3)

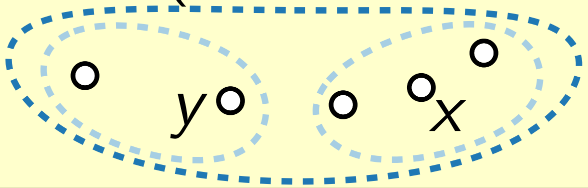


- FIND(1)? ➡ 3

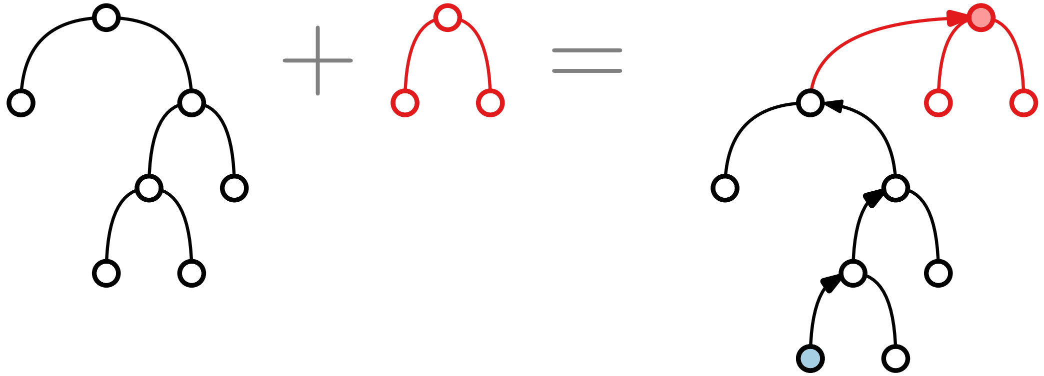
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

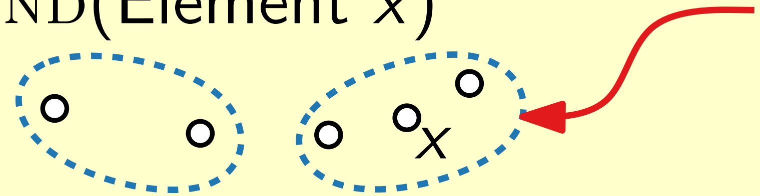
Union(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



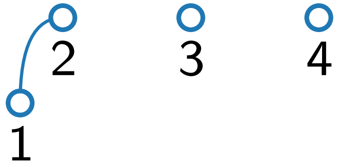
Find(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



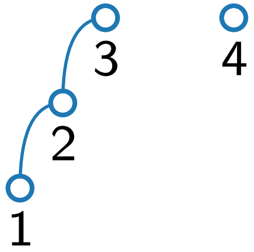
➡ Laufe zur Wurzel, gib Wurzel zurück.

Beispiel.

- Union(1, 2)



- Union(2, 3)

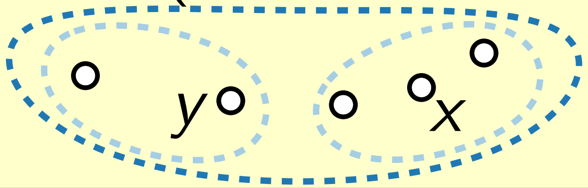


- Find(1)? ➡ 3
- Find(3)?

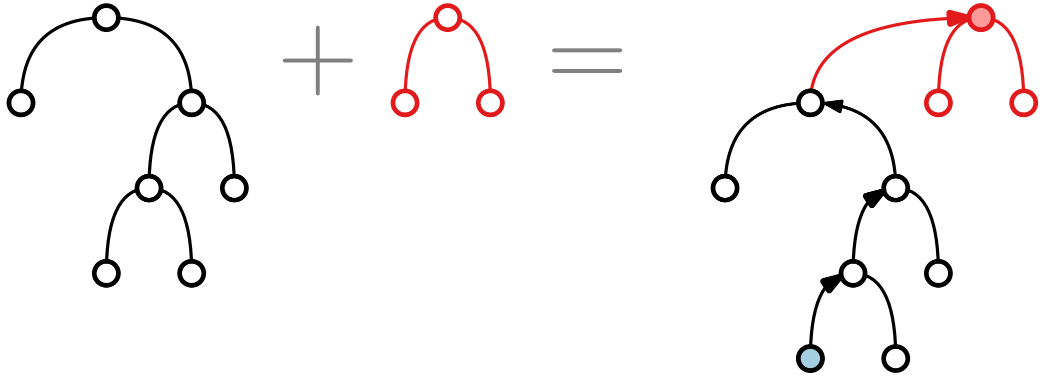
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

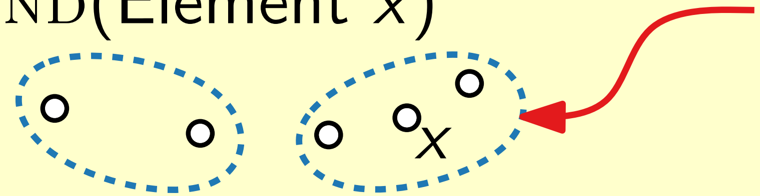
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



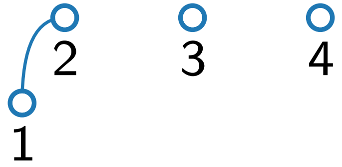
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



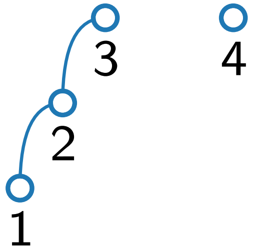
➡ Laufe zur Wurzel, gib Wurzel zurück.

Beispiel.

- UNION(1, 2)



- UNION(2, 3)

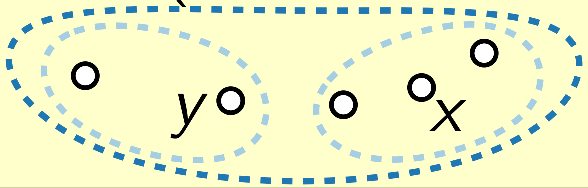


- FIND(1)? ➡ 3
- FIND(3)? ➡ 3

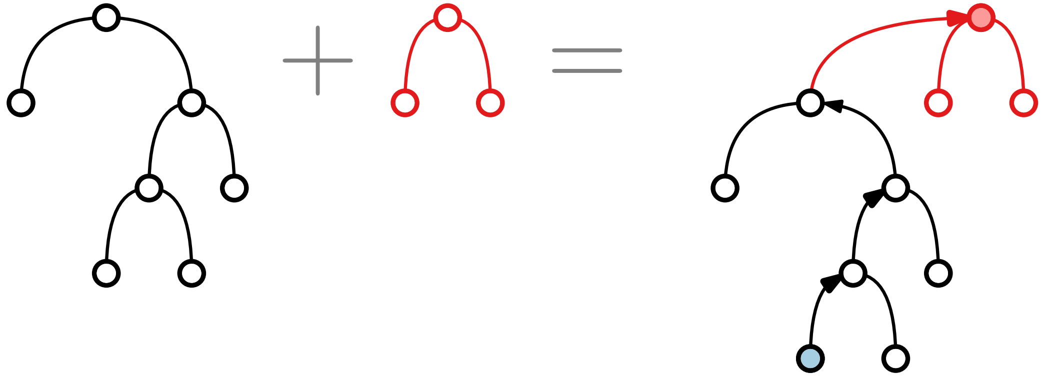
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

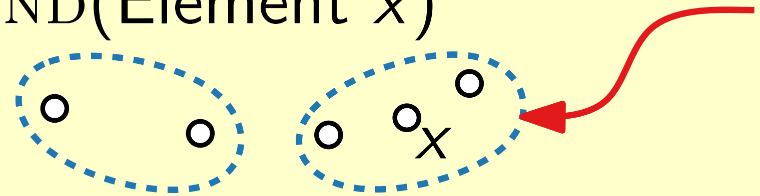
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



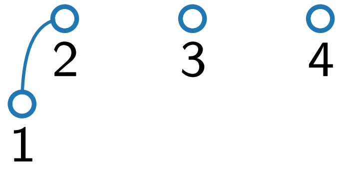
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



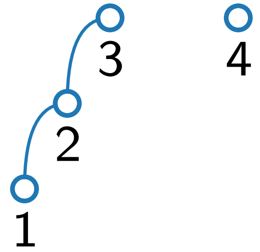
➡ Laufe zur Wurzel, gib Wurzel zurück.

Beispiel.

- UNION(1, 2)



- UNION(2, 3)

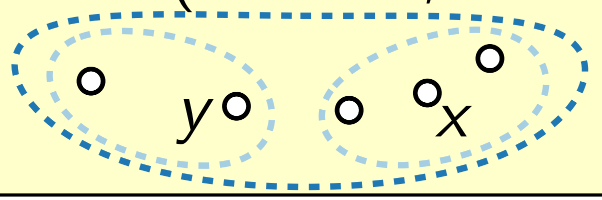


- FIND(1)? ➡ 3
- FIND(3)? ➡ 3
- FIND(1) = FIND(3)?

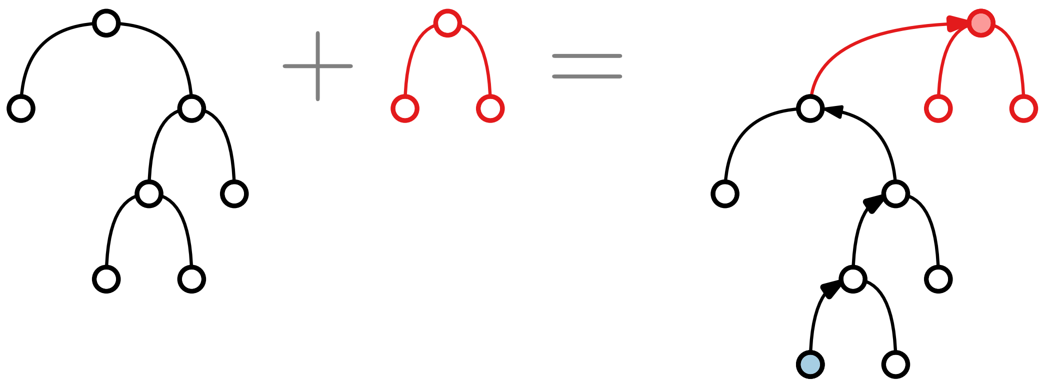
Realisierung Union-Find-Datenstruktur

Baumstruktur für jede Menge

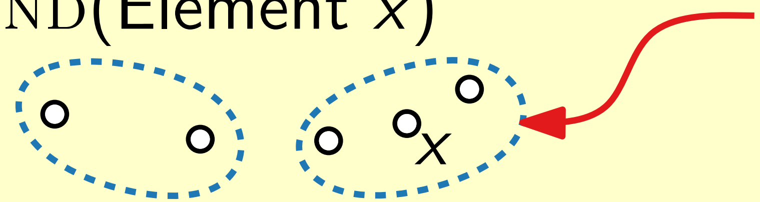
UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



➡ Hänge einen Baum an den anderen:



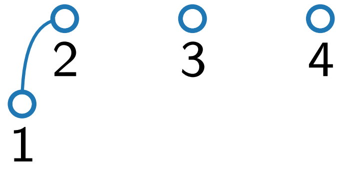
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



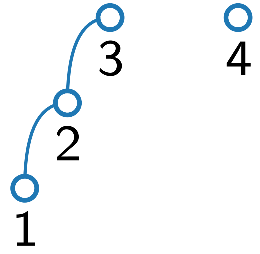
➡ Laufe zur Wurzel, gib Wurzel zurück.

Beispiel.

- UNION(1, 2)



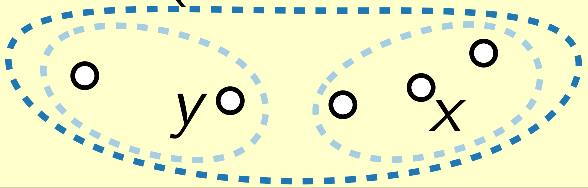
- UNION(2, 3)



- FIND(1)? ➡ 3
- FIND(3)? ➡ 3
- FIND(1) = FIND(3)?
  ➡ `true`

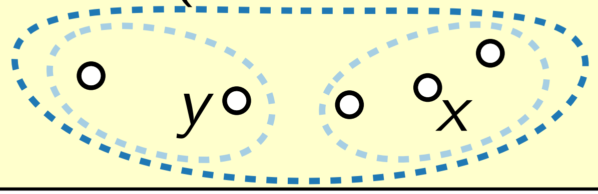
Zwei Verbesserungen

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



Zwei Verbesserungen

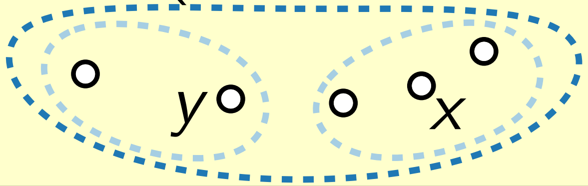
Union(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



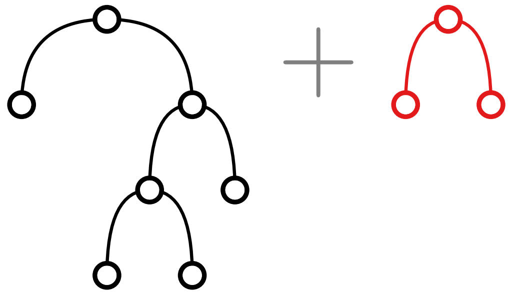
Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

Zwei Verbesserungen

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

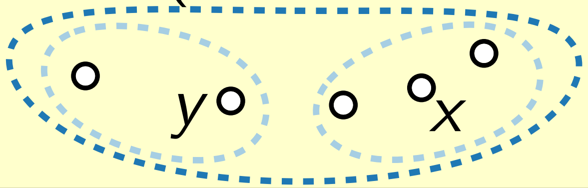


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

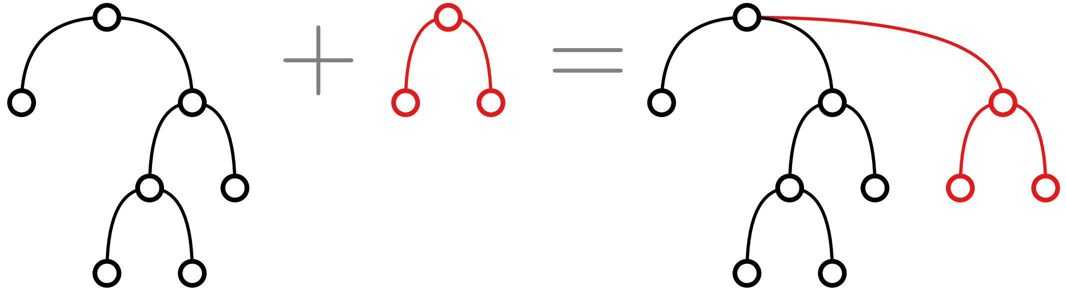


Zwei Verbesserungen

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

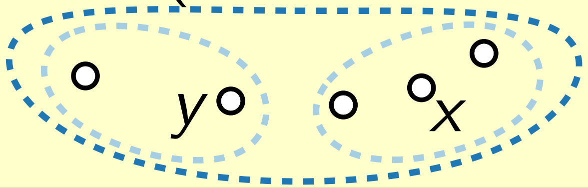


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:



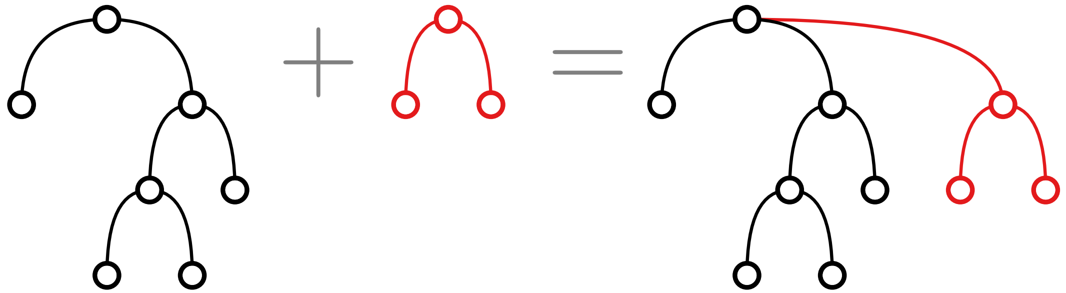
Zwei Verbesserungen

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.



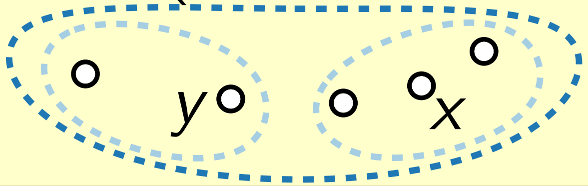
Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

➡ Hänge Baum mit kleinerer Tiefe an den mit größerer.



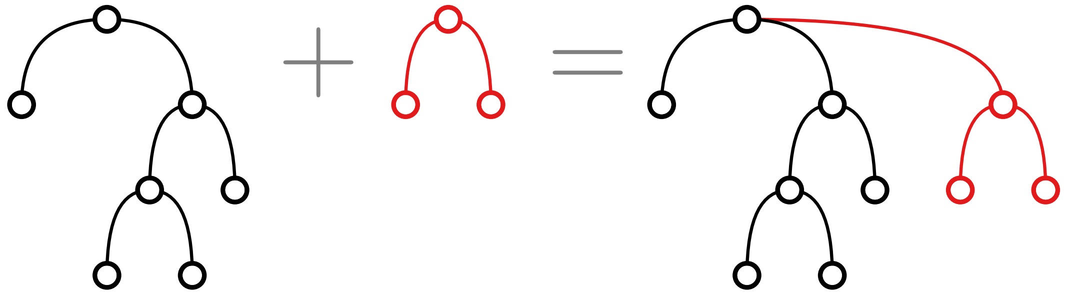
Zwei Verbesserungen

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

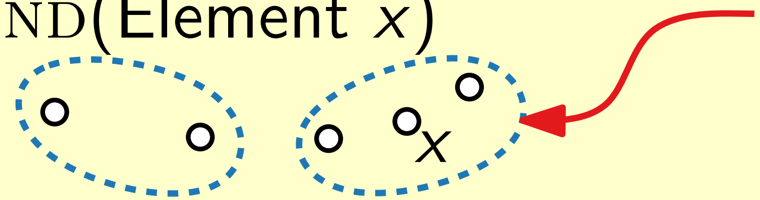


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

➡ Hänge Baum mit kleinerer Tiefe an den mit größerer.

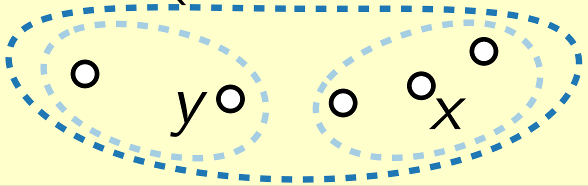


FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



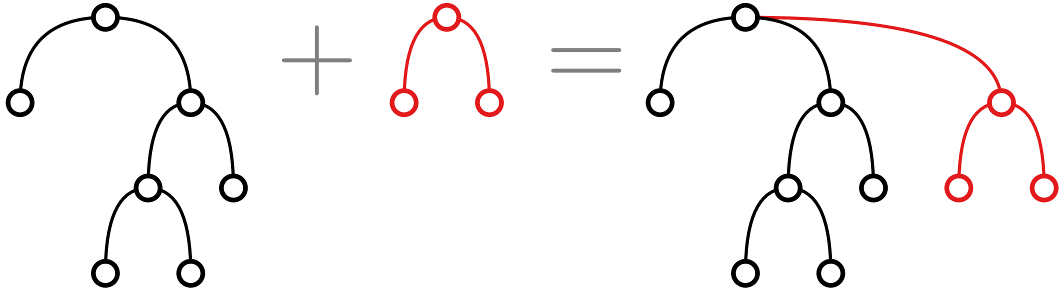
Zwei Verbesserungen

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

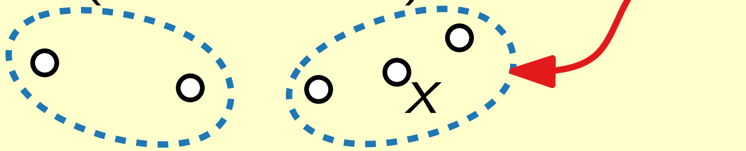


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

➡ Hänge Baum mit kleinerer Tiefe an den mit größerer.



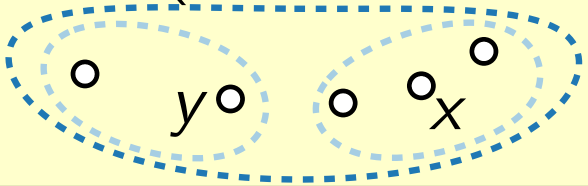
FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



Pfadkompression: Laufe zur Wurzel r, merke alle besuchten Knoten und mache sie zu Kindern von r.

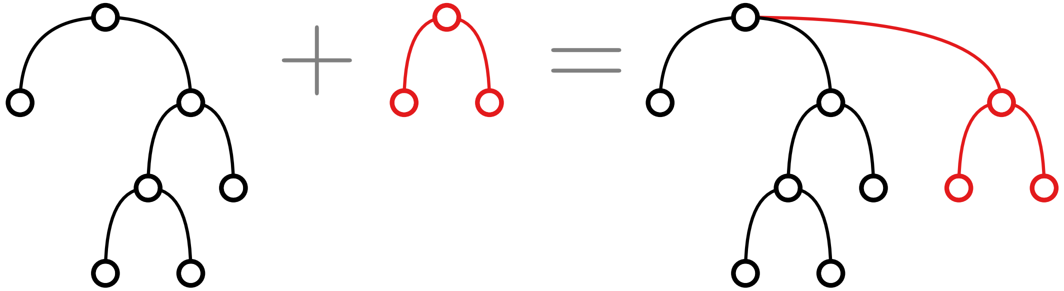
Zwei Verbesserungen

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

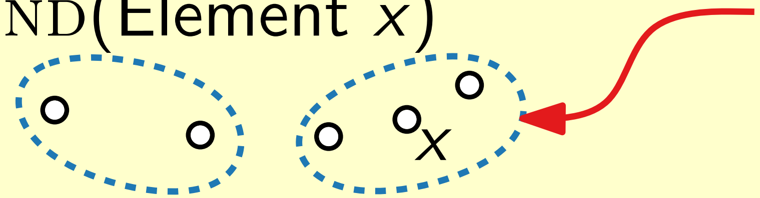


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

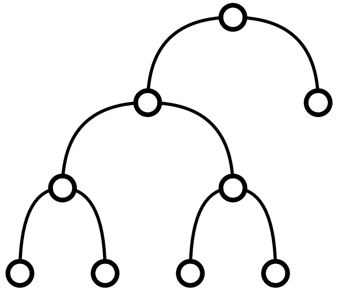
➡ Hänge Baum mit kleinerer Tiefe an den mit größerer.



FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.

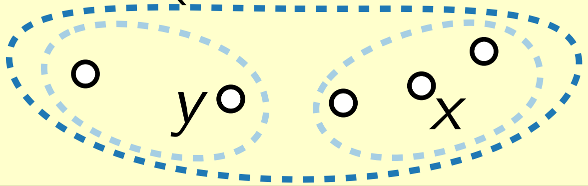


Pfadkompression: Laufe zur Wurzel r, merke alle besuchten Knoten und mache sie zu Kindern von r.



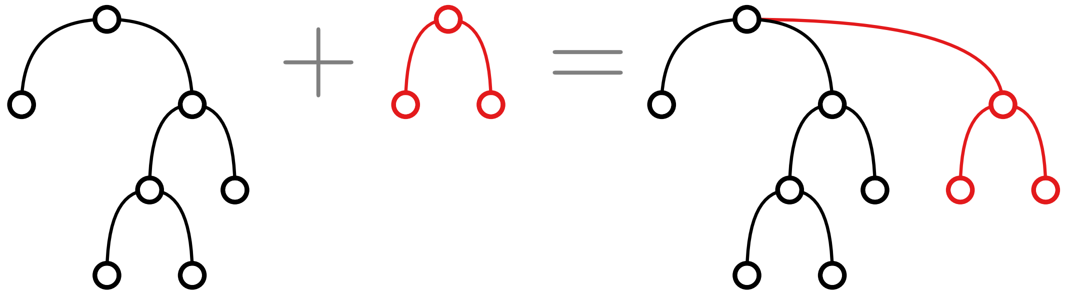
Zwei Verbesserungen

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

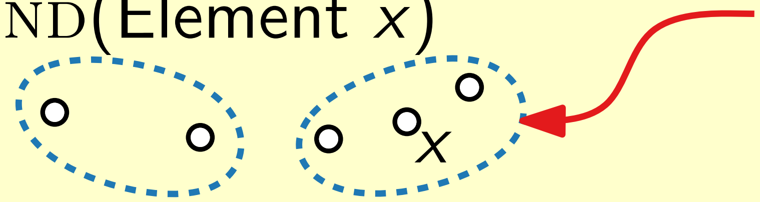


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

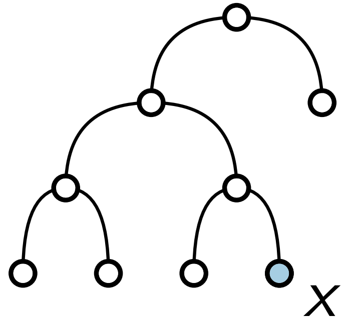
➡ Hänge Baum mit kleinerer Tiefe an den mit größerer.



FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.

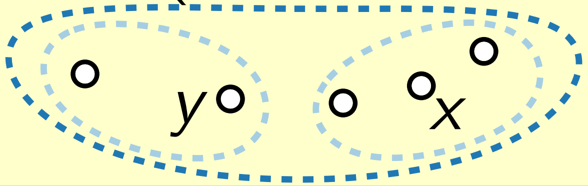


Pfadkompression: Laufe zur Wurzel r, merke alle besuchten Knoten und mache sie zu Kindern von r.



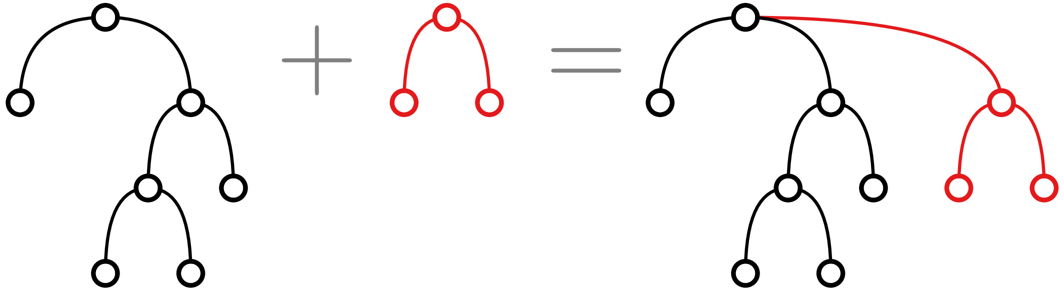
Zwei Verbesserungen

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

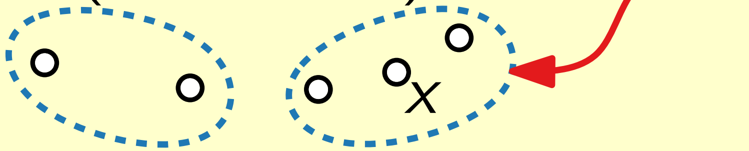


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

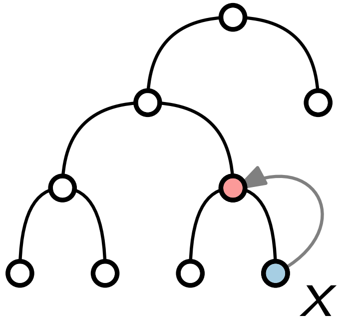
➡ Hänge Baum mit kleinerer Tiefe an den mit größerer.



FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.

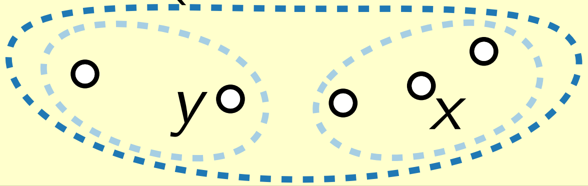


Pfadkompression: Laufe zur Wurzel r, merke alle besuchten Knoten und mache sie zu Kindern von r.

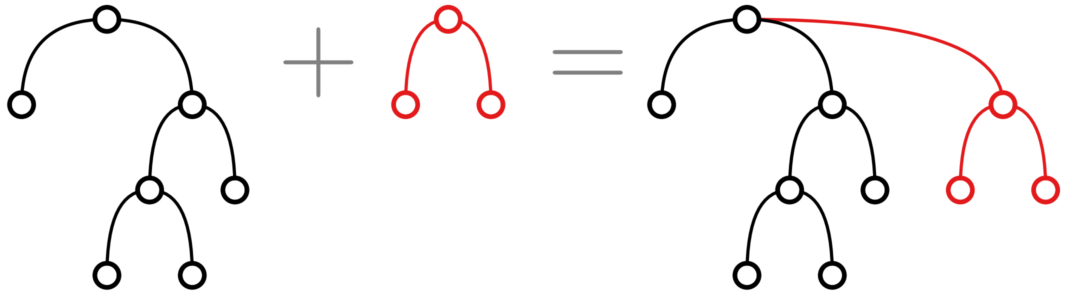


Zwei Verbesserungen

Union(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

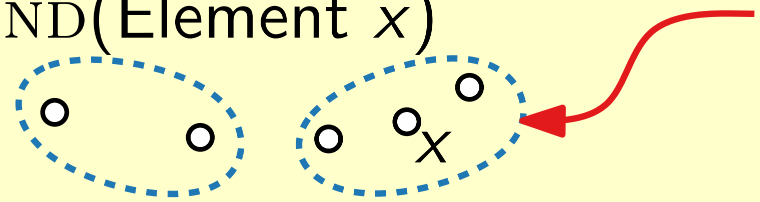


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

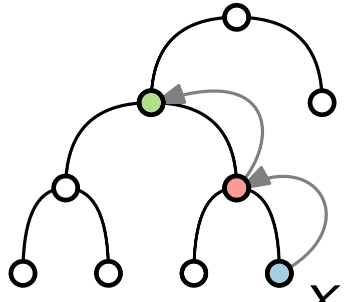


➡ Hänge Baum mit kleinerer Tiefe an den mit größerer.

Find(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.

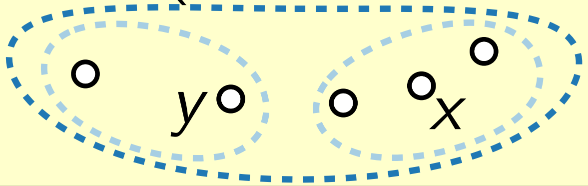


Pfadkompression: Laufe zur Wurzel r, merke alle besuchten Knoten und mache sie zu Kindern von r.



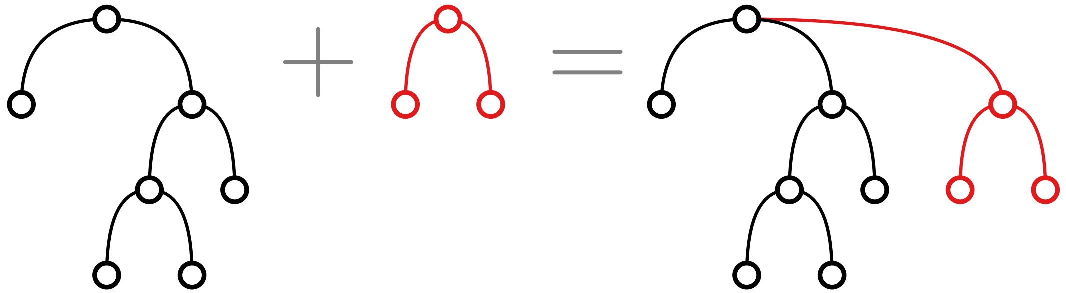
Zwei Verbesserungen

Union(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

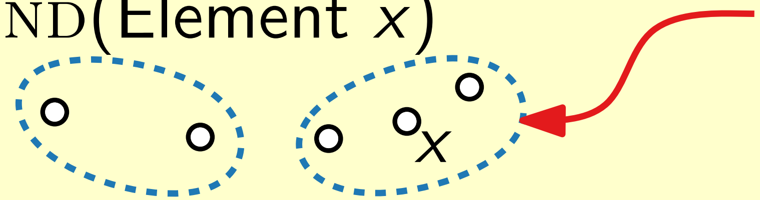


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

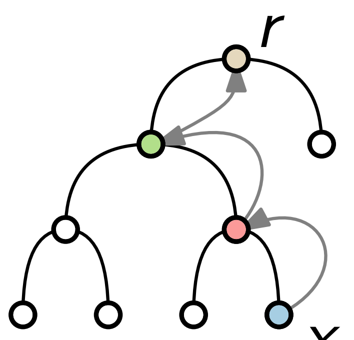
➡ Hänge Baum mit kleinerer Tiefe an den mit größerer.



Find(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.

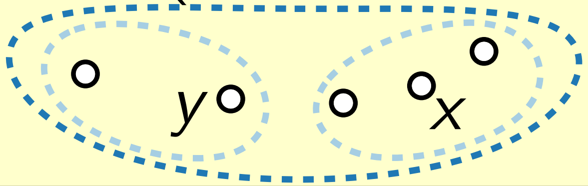


Pfadkompression: Laufe zur Wurzel r, merke alle besuchten Knoten und mache sie zu Kindern von r.



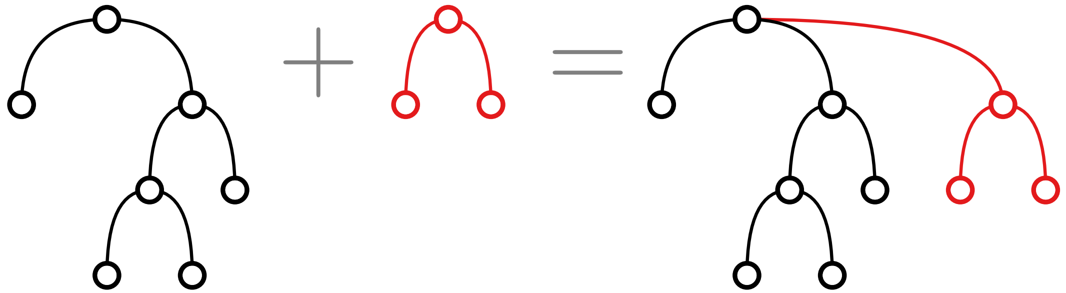
Zwei Verbesserungen

Union(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

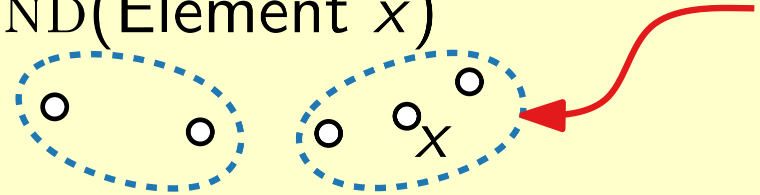


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

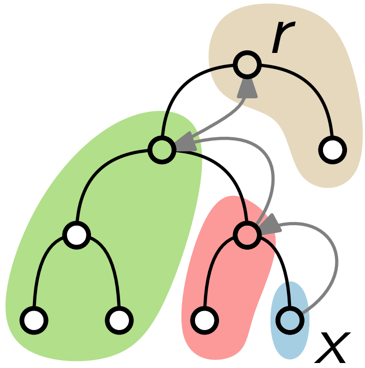
➡ Hänge Baum mit kleinerer Tiefe an den mit größerer.



Find(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.

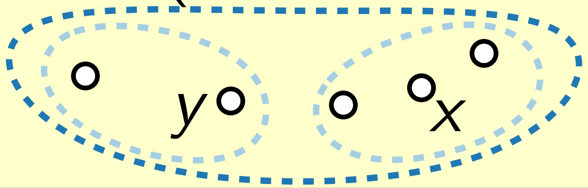


Pfadkompression: Laufe zur Wurzel r, merke alle besuchten Knoten und mache sie zu Kindern von r.



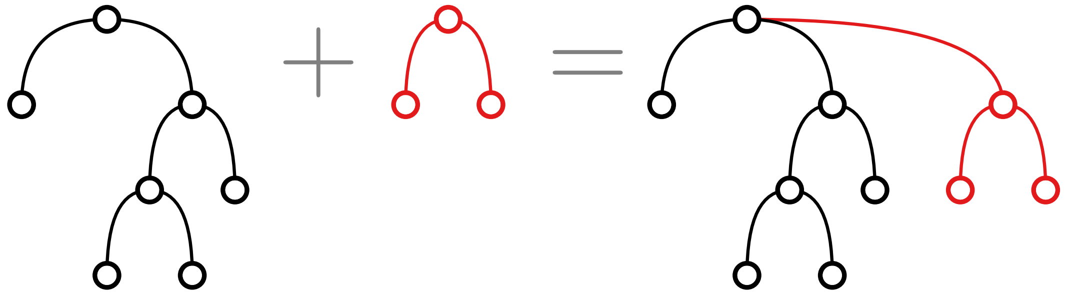
Zwei Verbesserungen

UNION(Elem. x, Elem. y) vereinigt die Mengen, die momentan x und y enthalten.

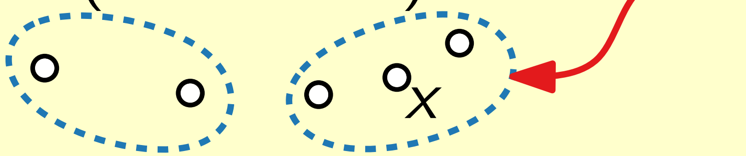


Union-by-Rank: Führe die Op. so aus, dass der neue Baum möglichst geringe Tiefe hat:

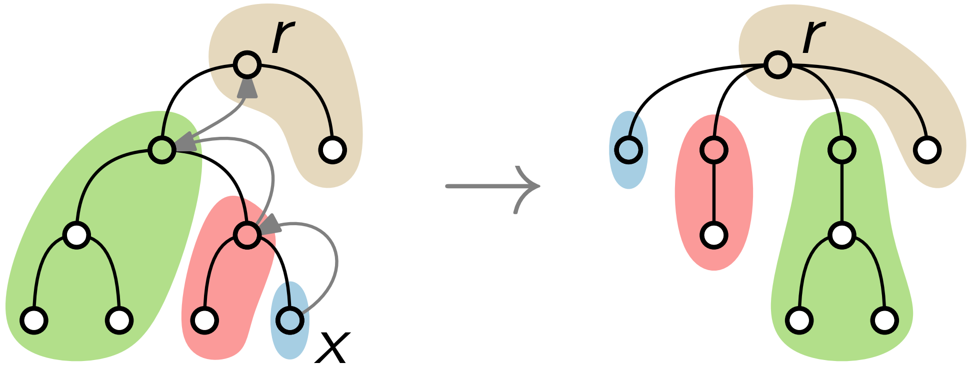
➡ Hänge Baum mit kleinerer Tiefe an den mit größerer.



FIND(Element x) liefert (Zeiger auf) die Menge zurück, die momentan x enthält.



Pfadkompression: Laufe zur Wurzel r, merke alle besuchten Knoten und mache sie zu Kindern von r.



nicht klausurrelevant

Kosten für Union-Find

Satz. Kosten für $m \times \textsc{Find}$ und $n \times \textsc{Union}$:

Kosten für Union-Find

Satz. Kosten für $m \times \textsc{Find}$ und $n \times \textsc{Union}$:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank

nicht klausurrelevant

Kosten für Union-Find

Satz. Kosten für $m \times$ Find und $n \times$ Union:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

Kosten für Union-Find

Satz. Kosten für $m \times$ Find und $n \times$ Union:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

Kosten für Union-Find

Satz. Kosten für $m \times$ FIND und $n \times$ UNION:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion

nicht klausurrelevant

Kosten für Union-Find

Satz. Kosten für $m \times$ Find und $n \times$ Union:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$

nicht klausurrelevant

Kosten für Union-Find

Satz. Kosten für $m \times$ Find und $n \times$ Union:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?“

nicht klausurrelevant

Kosten für Union-Find

Satz. Kosten für $m \times$ Find und $n \times$ Union:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?“
- $\alpha_2(n) = \lceil \log n \rceil$

Kosten für Union-Find

Satz. Kosten für $m \times$ FIND und $n \times$ UNION:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?"
- $\alpha_2(n) = \lceil \log n \rceil$
- $\alpha_3(n) = \log^*(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^*(\log n) & \text{sonst} \end{cases}$

Kosten für Union-Find

Satz. Kosten für $m \times$ FIND und $n \times$ UNION:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?“
- $\alpha_2(n) = \lceil \log n \rceil$
- $\alpha_3(n) = \log^*(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^*(\log n) & \text{sonst} \end{cases}$

z.B. $\log^*(2^{2^{2^2}}) = \log^*(65536) = 4$

Kosten für Union-Find

Satz. Kosten für $m \times$ FIND und $n \times$ UNION:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?"
- $\alpha_2(n) = \lceil \log n \rceil$
- $\alpha_3(n) = \log^*(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^*(\log n) & \text{sonst} \end{cases}$

z.B. $\log^*(2^{2^{2^2}}) = \log^*(65536) = 4$

$\log^*(2^{2^{2^{2^2}}}) = \log^*(2^{65536}) = 5$

Kosten für Union-Find

Satz. Kosten für $m \times$ Find und $n \times$ Union:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?“
- $\alpha_2(n) = \lceil \log n \rceil$
- $\alpha_3(n) = \log^*(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^*(\log n) & \text{sonst} \end{cases}$

z.B. $\log^*(2^{2^{2^2}}) = \log^*(65536) = 4$

$\log^*(2^{2^{2^{2^2}}}) = \log^*(\underbrace{2^{65536}}_{\approx 2 \cdot 10^{19729}}) = 5$

nicht klausurrelevant

Kosten für Union-Find

Satz. Kosten für $m \times$ FIND und $n \times$ UNION:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?“
- $\alpha_2(n) = \lceil \log n \rceil$
- $\alpha_3(n) = \log^*(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^*(\log n) & \text{sonst} \end{cases}$
- $\alpha_4(n) = \log^{**}(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^{**}(\log^* n) & \text{sonst} \end{cases}$

z.B. $\log^*(2^{2^{2^2}}) = \log^*(65536) = 4$

$\log^*(2^{2^{2^{2^2}}}) = \log^*(\underbrace{2^{65536}}_{\approx 2 \cdot 10^{19729}}) = 5$

Kosten für Union-Find

Satz. Kosten für $m \times$ FIND und $n \times$ UNION:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?“
- $\alpha_2(n) = \lceil \log n \rceil$
- $\alpha_3(n) = \log^*(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^*(\log n) & \text{sonst} \end{cases}$
- $\alpha_4(n) = \log^{**}(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^{**}(\log^* n) & \text{sonst} \end{cases}$

. . .

z.B. $\log^*(2^{2^{2^2}}) = \log^*(65536) = 4$

$\log^*(2^{2^{2^{2^2}}}) = \log^*(\underbrace{2^{65536}}_{\approx 2 \cdot 10^{19729}}) = 5$

nicht klausurrelevant

Kosten für Union-Find

Satz. Kosten für $m \times$ FIND und $n \times$ UNION:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?"
- $\alpha_2(n) = \lceil \log n \rceil$
- $\alpha_3(n) = \log^*(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^*(\log n) & \text{sonst} \end{cases}$

  z.B. $\log^*(2^{2^{2^2}}) = \log^*(65536) = 4$

  $\log^*(2^{2^{2^{2^2}}}) = \log^*(\underbrace{2^{65536}}_{\approx 2 \cdot 10^{19729}}) = 5$
- $\alpha_4(n) = \log^{**}(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^{**}(\log^* n) & \text{sonst} \end{cases}$

...

- $\alpha(n)$ ist das kleinste k, so dass $\alpha_k(n) \leq 3$

Kosten für Union-Find

Satz. Kosten für $m \times$ FIND und $n \times$ UNION:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

---

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?“
- $\alpha_2(n) = \lceil \log n \rceil$
- $\alpha_3(n) = \log^*(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^*(\log n) & \text{sonst} \end{cases}$
- $\alpha_4(n) = \log^{**}(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^{**}(\log^* n) & \text{sonst} \end{cases}$

. . .

- $\alpha(n)$ ist das kleinste k, so dass $\alpha_k(n) \leq 3$

z.B. $\log^*(2^{2^{2^2}}) = \log^*(65536) = 4$

$\log^*(2^{2^{2^{2^2}}}) = \log^*(\underbrace{2^{65536}}_{\approx 2 \cdot 10^{19729}}) = 5$

$\alpha(n) \leq 4$ für $n \leq 2^{2^{2^{2^{2^{2^2}}}}} \approx 10^{10^{10^{19729}}}$

Kosten für Union-Find

Satz. Kosten für $m \times$ FIND und $n \times$ UNION:

- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?"
- $\alpha_2(n) = \lceil \log n \rceil$
- $\alpha_3(n) = \log^*(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^*(\log n) & \text{sonst} \end{cases}$
- $\alpha_4(n) = \log^{**}(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^{**}(\log^* n) & \text{sonst} \end{cases}$

...

- $\alpha(n)$ ist das kleinste k, so dass $\alpha_k(n) \leq 3$

z.B. $\log^*(2^{2^{2^2}}) = \log^*(65536) = 4$

$\log^*(2^{2^{2^{2^2}}}) = \underbrace{\log^*(2^{65536})}_{\approx 2 \cdot 10^{19729}} = 5$

$\alpha(n) \leq 4$ für $n \leq 2^{2^{2^{2^{2^{2^2}}}}} \approx 10^{10^{10^{19729}}}$

$\alpha(n) \leq 5$ für $n \leq 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \Big\} 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \Big\} 2^{2^{2^2}}$ mal mal

nicht klausurrelevant

Kosten für Union-Find

Satz. Kosten für $m \times$ FIND und $n \times$ UNION:
- $\mathcal{O}(n + m \log n)$ mit Union-by-Rank
- $\mathcal{O}(n + m \cdot \alpha(n))$ mit Union-by-Rank und Pfadkompression

$\Omega(n + m \cdot \alpha(n))$ ist untere Schranke für Union-Find [Tarjan '79]

- $\alpha(n)$ ist die inverse Ackermannfunktion
- $\alpha_1(n) = \lceil n/2 \rceil$
- $\alpha_k(n) =$ „wie oft muss ich $\alpha_{k-1}(n)$ auf n anwenden, um auf 1 zu kommen?“
- $\alpha_2(n) = \lceil \log n \rceil$
- $\alpha_3(n) = \log^*(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^*(\log n) & \text{sonst} \end{cases}$
- $\alpha_4(n) = \log^{**}(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ 1 + \log^{**}(\log^* n) & \text{sonst} \end{cases}$

...

- $\alpha(n)$ ist das kleinste k, so dass $\alpha_k(n) \leq 3$

z.B. $\log^*(2^{2^{2^2}}) = \log^*(65536) = 4$

$\log^*(2^{2^{2^{2^2}}}) = \log^*(\underbrace{2^{65536}}_{\approx 2 \cdot 10^{19729}}) = 5$

$\alpha(n) \leq 4$ für $n \leq 2^{2^{2^{2^{2^{2^2}}}}} \approx 10^{10^{10^{19729}}}$

$\alpha(n) \leq 5$ für $n \leq 2^{2^{\cdot^{\cdot^{\cdot^2}}}} \Big\} 2^{2^{\cdot^{\cdot^{\cdot^2}}}} \Big\} 2^{2^{2^2}}$ mal mal

Übersicht: Algorithmen für minimale Spannbäume

Jarník-Prim

- geht (wie Dijkstra / BFS) wellenförmig von einem Startknoten aus,

Kruskal

- bearbeitet Kanten nach aufsteigendem (genauer: nicht-absteig.) Gewicht,

Übersicht: Algorithmen für minimale Spannbäume

Jarník-Prim

- geht (wie Dijkstra / BFS) wellenförmig von einem Startknoten aus,
- aktuelle Kantenmenge zusammenhängend,

Kruskal

- bearbeitet Kanten nach aufsteigendem (genauer: nicht-absteig.) Gewicht,
- nach Einfügen der i. Kante gibt es $n-i$ Zusammenhangskomponenten,

Übersicht: Algorithmen für minimale Spannbäume

Jarník-Prim

- geht (wie Dijkstra / BFS) wellenförmig von einem Startknoten aus,
- aktuelle Kantenmenge zusammenhängend,
- Laufzeit $\mathcal{O}(E + V \log V)$.

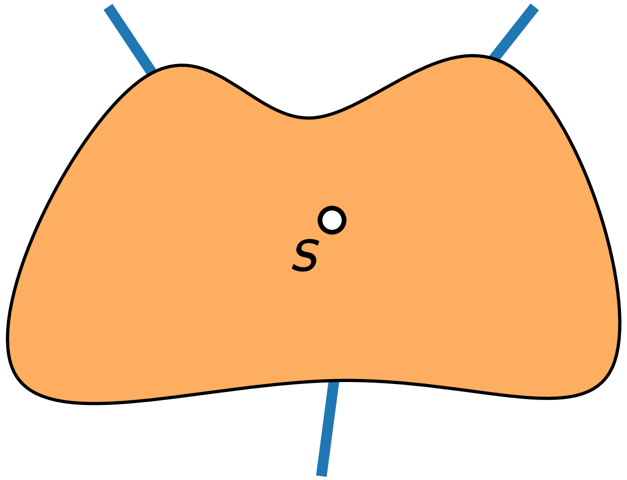
Kruskal

- bearbeitet Kanten nach aufsteigendem (genauer: nicht-absteig.) Gewicht,
- nach Einfügen der i. Kante gibt es $n - i$ Zusammenhangskomponenten,
- Laufzeit $\mathcal{O}(E \log V)$ oder $\mathcal{O}(E \cdot \alpha(V))$ falls vorsortiert.

Übersicht: Algorithmen für minimale Spannbäume

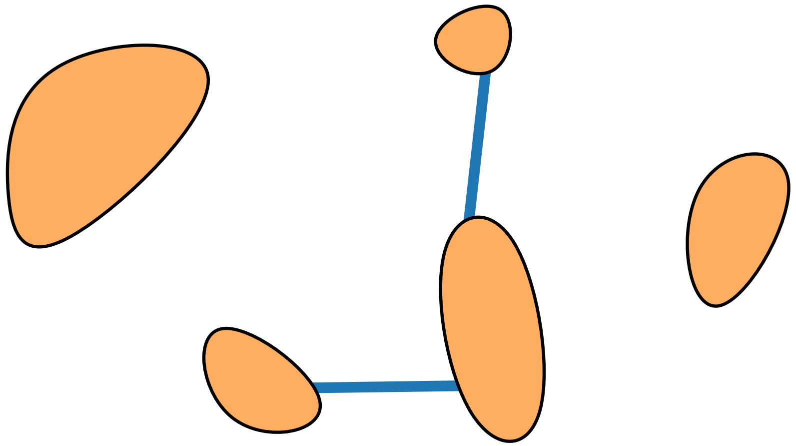
Jarník-Prim

- geht (wie Dijkstra / BFS) wellenförmig von einem Startknoten aus,
- aktuelle Kantenmenge zusammenhängend,
- Laufzeit $\mathcal{O}(E + V \log V)$.



Kruskal

- bearbeitet Kanten nach aufsteigendem (genauer: nicht-absteig.) Gewicht,
- nach Einfügen der i. Kante gibt es $n-i$ Zusammenhangskomponenten,
- Laufzeit $\mathcal{O}(E \log V)$ oder $\mathcal{O}(E \cdot \alpha(V))$ falls vorsortiert.



Übersicht: Algorithmen für minimale Spannbäume

Jarník-Prim

- geht (wie Dijkstra / BFS) wellenförmig von einem Startknoten aus,
- aktuelle Kantenmenge zusammenhängend,
- Laufzeit $\mathcal{O}(E + V \log V)$.

Kruskal

- bearbeitet Kanten nach aufsteigendem (genauer: nicht-absteig.) Gewicht,
- nach Einfügen der i. Kante gibt es $n - i$ Zusammenhangskomponenten,
- Laufzeit $\mathcal{O}(E \log V)$ oder $\mathcal{O}(E \cdot \alpha(V))$ falls vorsortiert.

GreedySpannbaum(G, w)
  Wende blaue Regel oder rote Regel an,
   bis alle Kanten gefärbt sind.
  Gib $E' = \{$blaue Kanten$\}$ zurück.

Übersicht: Algorithmen für minimale Spannbäume

Jarník-Prim

- geht (wie Dijkstra / BFS) wellenförmig von einem Startknoten aus,
- aktuelle Kantenmenge zusammenhängend,
- Laufzeit $\mathcal{O}(E + V \log V)$.

Kruskal

- bearbeitet Kanten nach aufsteigendem (genauer: nicht-absteig.) Gewicht,
- nach Einfügen der i. Kante gibt es $n - i$ Zusammenhangskomponenten,
- Laufzeit $\mathcal{O}(E \log V)$ oder $\mathcal{O}(E \cdot \alpha(V))$ falls vorsortiert.

GreedySpannbaum(G, w)
Wende blaue Regel oder rote Regel an, bis alle Kanten gefärbt sind.
Gib $E' = \{$blaue Kanten$\}$ zurück.

Blaue Regel:
Wähle Schnitt, den keine blaue Kante kreuzt.
Färbe leichte Kante blau.

Rote Regel:
Wähle Kreis ohne rote Kante.
Färbe größte ungefärbte Kante auf Kreis rot.