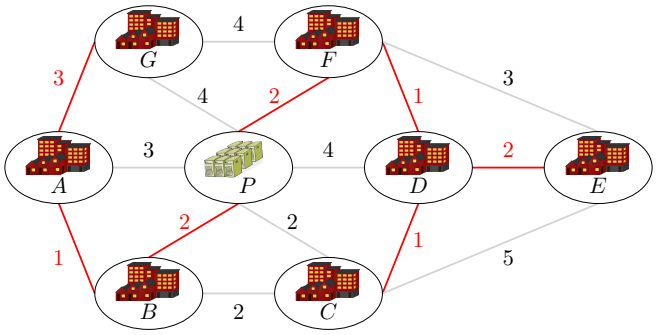
Minimum spanning tree problem MmF

Ein Internet-Provider möchte 7 Städte mit Glasfaserkabeln in sein Netzwerk anbinden.
Die Anbindung kann entweder direkt mit dem Provider oder indirekt über Städte erfolgen.
Alle möglichen Verbindungen und die Kosten für die Kabelverlegung (in Mio. €) sind dargestellt.



Wie viele Verbindungen sind mindestens notwendig, um alle Städte anzubinden?

7 Verbindungen

Versuche ein Netzwerk zu finden, bei dem die Gesamtkosten so klein wie möglich sind.

Minimale Gesamtkosten: 12 Mio. €

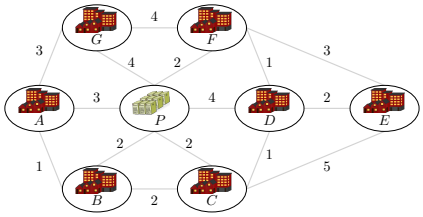
Kantengewichteter Graph MmF

In der Graphentheorie sprechen wir von **kantengewichteten Graphen** und nennen ...

... die Orte A, B, C, D, E, F, G, P auch **Knoten**.

... die Strecken zwischen 2 Knoten auch **Kanten**.

... die Zahlen neben den Kanten auch **Gewichte**.



Die positiven Gewichte geben in diesem Kontext die Kosten zur Verbindung zweier Orte an.

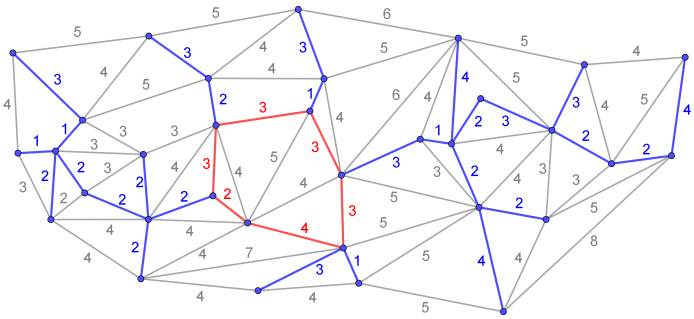
Kreise MmF

Im rechts unten dargestellten Netzwerk sind alle Städte direkt oder indirekt miteinander verbunden.

Die Gesamtkosten sind aber sicher *nicht* so klein wie möglich, weil es einen **Kreis** gibt.

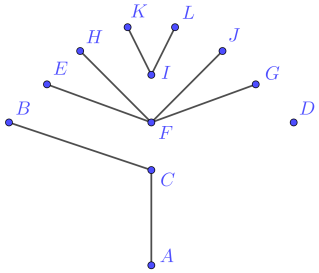
Zeichne ihn rechts ein.

Erkläre, um wieviel du die Gesamtkosten sicher reduzieren kannst.
Entfernt man im Kreis die Verbindung mit Kosten 4, bleiben alle Städte verbunden.

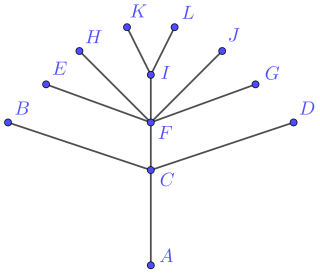


Zusammenhangskomponenten & Bäume MmF

Alle Knoten eines Graphen, die direkt oder indirekt miteinander verbunden sind, bilden eine sogenannte **Zusammenhangskomponente** (kurz: **Komponente**).
Zum Beispiel hat der Graph links die 4 Komponenten $\{A, B, C\}$, $\{D\}$, $\{E, F, G, H, J\}$ und $\{I, K, L\}$.



Der Graph rechts hat nur eine Komponente.
Solche Graphen heißen **zusammenhängend**.



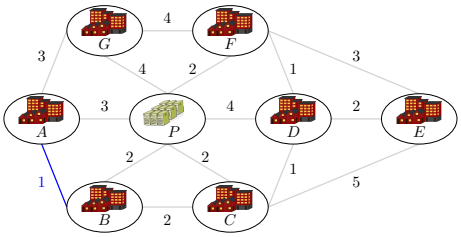
Ein zusammenhängender Graph, der keinen Kreis enthält, heißt **Baum**.

Kruskal-Algorithmus (1956)

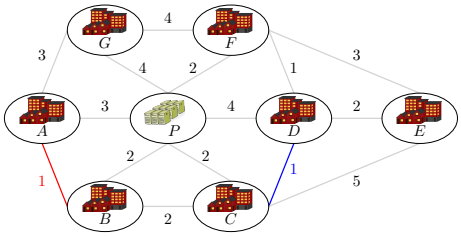
Der **Kruskal-Algorithmus** findet in jedem zusammenhängenden Graphen mit positiven Kantengewichten einen **minimal aufspannenden Baum**.

Dieser Baum verbindet also alle Knoten des Graphen so, dass die Gesamtkosten so klein wie möglich sind.

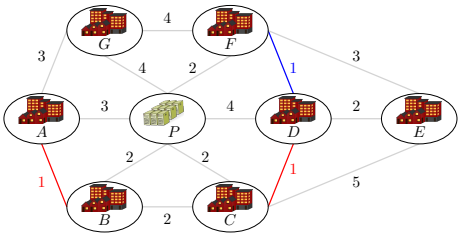
Wenn der Graph n Knoten enthält, dann benötigt der Algorithmus dafür $n-1$ Durchläufe.
In jedem Durchlauf werden zwei Komponenten durch Hinzufügen einer Kante verschmolzen:



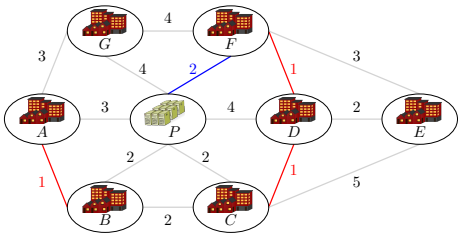
Durchlauf 1:
Komponenten: $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, $\{E\}$, $\{F\}$, $\{G\}$, $\{P\}$
Wähle unter allen Kanten, die zwei Komponenten verbinden, eine günstigste Kante aus.



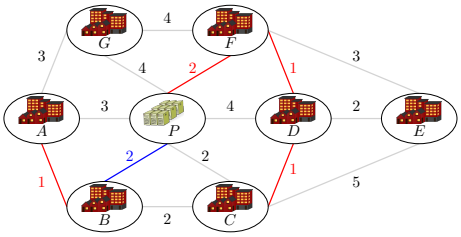
Durchlauf 2:
Komponenten: $\{A,B\}$, $\{C\}$, $\{D\}$, $\{E\}$, $\{F\}$, $\{G\}$, $\{P\}$
Wähle unter allen Kanten, die zwei Komponenten verbinden, eine günstigste Kante aus.



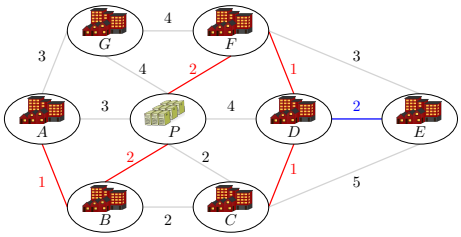
Durchlauf 3:
Komponenten: $\{A,B\}$, $\{C,D\}$, $\{E\}$, $\{F\}$, $\{G\}$, $\{P\}$
Wähle unter allen Kanten, die zwei Komponenten verbinden, eine günstigste Kante aus.



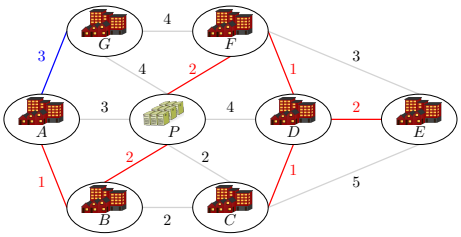
Durchlauf 4:
Komponenten: $\{A,B\}$, $\{C,D,F\}$, $\{E\}$, $\{G\}$, $\{P\}$
Wähle unter allen Kanten, die zwei Komponenten verbinden, eine günstigste Kante aus.



Durchlauf 5:
Komponenten: $\{A,B\}$, $\{C,D,F,P\}$, $\{E\}$, $\{G\}$
Wähle unter allen Kanten, die zwei Komponenten verbinden, eine günstigste Kante aus.



Durchlauf 6:
Komponenten: $\{A,B,C,D,F,P\}$, $\{E\}$, $\{G\}$
Wähle unter allen Kanten, die zwei Komponenten verbinden, eine günstigste Kante aus.



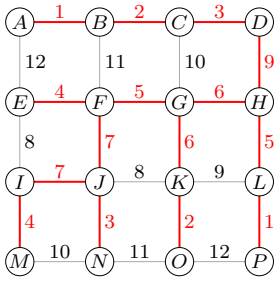
Durchlauf 7:
Komponenten: $\{A,B,C,D,E,F,P\}$, $\{G\}$
Wähle unter allen Kanten, die zwei Komponenten verbinden, eine günstigste Kante aus.

Beachte, dass die günstigeren Kanten keine Komponenten verbinden.

Der Kruskal-Algorithmus ist ein sogenannter **Greedy-Algorithmus**. „greedy“ ist englisch für gierig.
Er trifft in jedem Durchlauf jene Entscheidung, die zum aktuellen Zeitpunkt am besten erscheint.

Kruskal-Algorithmus

Rechts ist ein zusammenhängender Graph mit positiven Kantengewichten dargestellt.
Ermittle einen minimal aufspannenden Baum mit dem Kruskal-Algorithmus.
Welches Gesamtgewicht hat dieser Baum?



$$2 \cdot (1+2+3+4+5+6+7)+9 = 65$$

Euklidischer Abstand

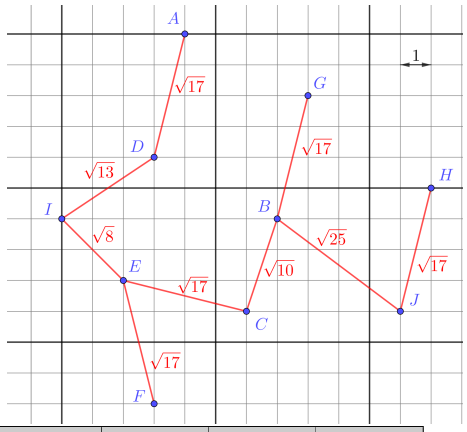
Im quadratischen Raster rechts sind
10 Gitterpunkte eingezeichnet.

1) Trage die Entfernungen zwischen den Punkten in der Tabelle unten ein.

2) Die Entfernungen sind die Kantengewichte. Ermittle einen minimal aufspannden Baum, und zeichne ihn rechts ein.

3) Welches Gesamtgewicht hat dieser Baum?

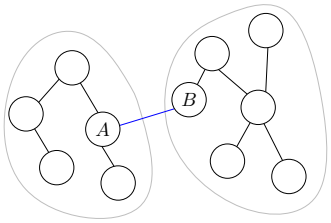
$$\sqrt{8}+\sqrt{10}+\sqrt{13}+5\cdot\sqrt{17}+\sqrt{25} = 35{,}21\ldots$$



	A	B	C	D	E	F	G	H	I	J
A	–	$\sqrt{45}$	$\sqrt{85}$	$\sqrt{17}$	$\sqrt{68}$	$\sqrt{145}$	$\sqrt{20}$	$\sqrt{89}$	$\sqrt{52}$	$\sqrt{130}$
B	–	–	$\sqrt{10}$	$\sqrt{20}$	$\sqrt{29}$	$\sqrt{52}$	$\sqrt{17}$	$\sqrt{26}$	$\sqrt{49}$	$\sqrt{25}$
C	–	–	–	$\sqrt{34}$	$\sqrt{17}$	$\sqrt{18}$	$\sqrt{53}$	$\sqrt{52}$	$\sqrt{45}$	$\sqrt{25}$
D	–	–	–	–	$\sqrt{17}$	$\sqrt{64}$	$\sqrt{29}$	$\sqrt{82}$	$\sqrt{13}$	$\sqrt{89}$
E	–	–	–	–	–	$\sqrt{17}$	$\sqrt{72}$	$\sqrt{109}$	$\sqrt{8}$	$\sqrt{82}$
F	–	–	–	–	–	–	$\sqrt{125}$	$\sqrt{130}$	$\sqrt{45}$	$\sqrt{73}$
G	–	–	–	–	–	–	–	$\sqrt{25}$	$\sqrt{80}$	$\sqrt{58}$
H	–	–	–	–	–	–	–	–	$\sqrt{145}$	$\sqrt{17}$
I	–	–	–	–	–	–	–	–	–	$\sqrt{130}$
J	–	–	–	–	–	–	–	–	–	–

Warum liefert Kruskal einen aufspannenden Baum?

Ein Graph enthält keinen Kreis.
Die Knoten A und B sind in verschiedenen Komponenten.
Eine Kante zwischen A und B wird hinzugefügt.
Erkläre, warum der Graph dann *keinen* Kreis enthalten kann.
Indirekte Begründung: Angenommen, der Graph enthält einen Kreis.
Dann gibt es aber auch ohne die neue Kante einen Weg von A zu B.
Die Knoten A und B sind aber in verschiedenen Komponenten. ϟ

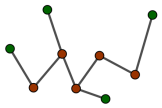


Der Kruskal-Algorithmus liefert also einen Graphen, der alle Knoten enthält, aber keinen Kreis.

Ein solcher Graph heißt **aufspannender Baum**.

Bäume haben Blätter.

Erkläre, warum es in jedem Baum mit mindestens 2 Knoten einen Knoten geben muss, von dem nur eine Kante wegführt.



Ein solcher Knoten heißt auch **Blatt**.

Indirekte Begründung: Wenn von jedem Knoten mindestens 2 Kanten wegführen, dann könnte man ausgehend von einem Knoten immer wieder zu einem Nachbarn weitergehen. Spätestens nach n Schritten landet man bei einem Knoten, den man schon besucht hat. Der Graph hat also einen Kreis und ist damit kein Baum. ↯

Kantenanzahl in Bäumen

Begründe mit vollständiger Induktion, warum jeder Baum mit $n \geq 2$ Knoten genau $n-1$ Kanten hat.

1) Induktionsanfang $n = 2$: ⟷ ✓

2) Induktionsschritt $n \to n+1$: Entferne vom Baum mit $n+1$ Knoten ein Blatt und die zugehörige Kante. Der neue Baum hat n Knoten. Aus der Induktionsvoraussetzung folgt, dass der neue Baum $n-1$ Kanten hat. Der ursprüngliche Baum hat also n Kanten. ✓

Warum liefert Kruskal einen *minimal* aufspannenden Baum?

Gegeben ist ein zusammenhängender Graph mit $n \geq 2$ Knoten und positiven Kantengewichten.
Dann liefert der Kruskal-Algorithmus einen aufspannenden Baum mit minimalen Gesamtkosten.
Wir geben dafür eine indirekte Begründung:
Angenommen, es gibt eine beste Lösung mit *weniger* Gesamtkosten als die Kruskal-Lösung.

1) Warum kann diese beste Lösung keinen Kreis enthalten?
Wenn sie einen Kreis, enthält, könnte man eine beliebige Kante vom Kreis entfernen. Der neue Graph ist auch zusammenhängend und hat weniger Gesamtkosten, weil alle Gewichte positiv sind.

2) Die beste Lösung muss also ein Baum sein, der $n-1$ Kanten hat.

3) Fügt man einem Graphen eine Kante hinzu, wird die Anzahl der Komponenten höchstens um 1 kleiner. Damit ein Graph mit n Knoten und $n-1$ Kanten schließlich zusammenhängend ist, muss also *jede* Kante jeweils 2 Komponenten verschmelzen.

4) Der Kruskal-Algorithmus wählt $n-1$ Kanten mit aufsteigenden Gewichten $k_1 \leq k_2 \leq \cdots \leq k_{n-1}$. Wir sortieren die Kanten von der besten Lösung auch nach den Gewichten $b_1 \leq b_2 \leq \cdots \leq b_{n-1}$. Warum muss es einen Durchlauf d geben mit $k_d > b_d$?
Sonst kann die beste Lösung nicht weniger Gesamtkosten als die Kruskal-Lösung haben.

5) Behauptung: Die Kruskal-Lösung hat *vor* Durchlauf d höchstens so viele Komponenten wie die beste Lösung *nach* Durchlauf d. Das ist schließlich ein Widerspruch zu **3)**. ↯

Es kann also keine günstigere Lösung als die Kruskal-Lösung geben.

Wir begründen die Behauptung. Dazu nehmen wir die Knoten einer beliebigen Komponente der besten Lösung *nach* Durchlauf d. Erkläre, warum die Kruskal-Lösung bereits *vor* Durchlauf d eine Komponente haben muss, die alle diese Knoten enthält.
Für jede Kante dieser beliebigen Komponente gilt:
Die Kosten sind kleiner als k_d, aber Kruskal wählt sie in Durchlauf d nicht aus.
Die beiden Knoten müssen also schon vor Durchlauf d in der gleichen Kruskal-Komponente liegen.
Alle Knoten dieser beliebigen Komponente sind also in einer Kruskal-Komponente enthalten.