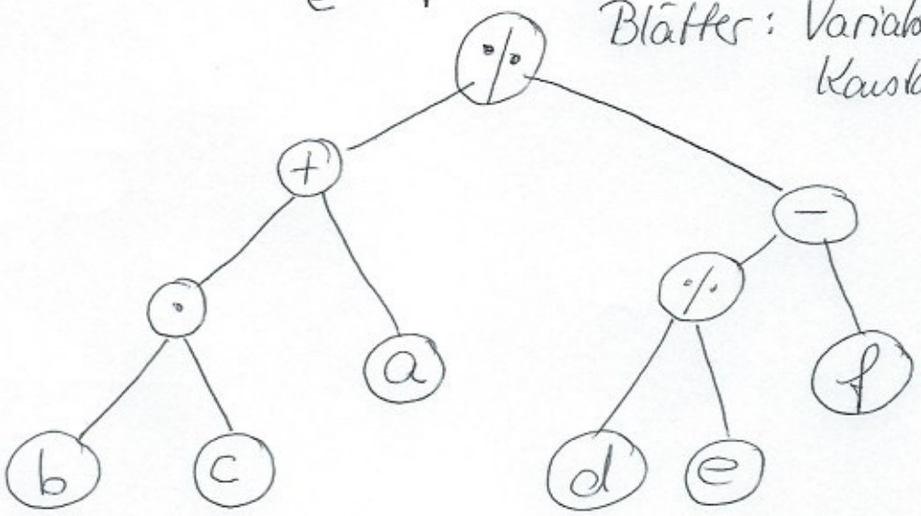
1. Beispiel

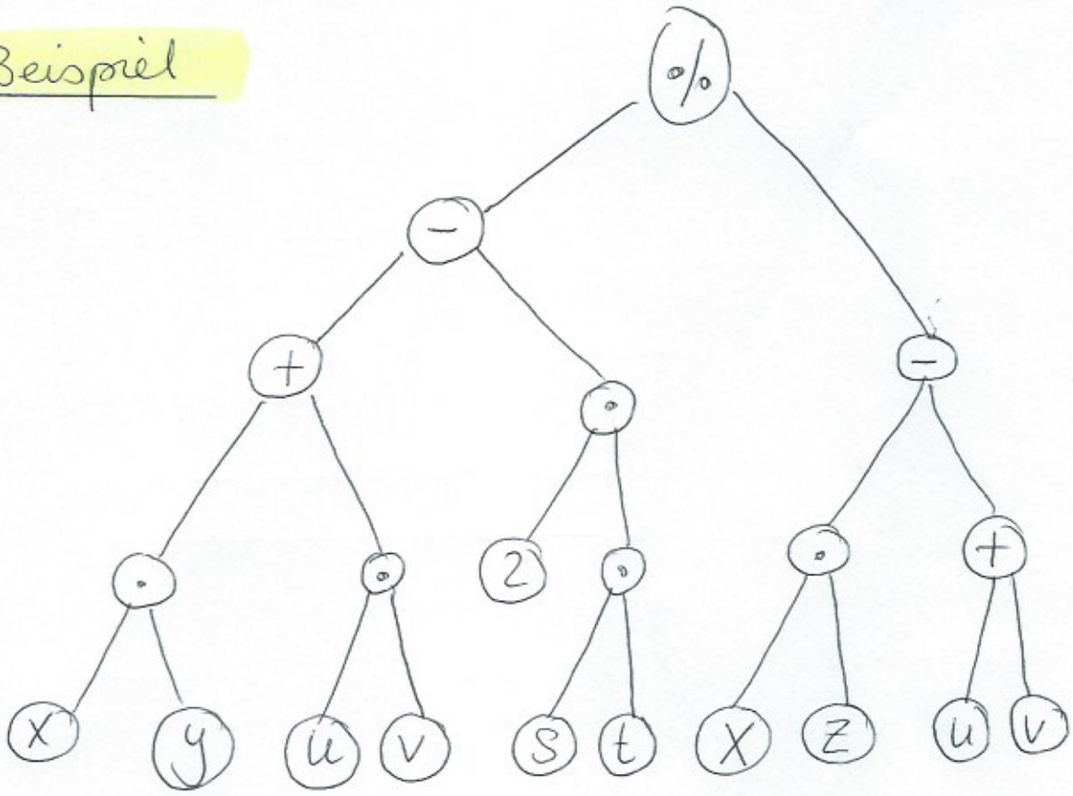
$$\frac{a+b\cdot c}{\frac{d}{e}-f}$$

innere Knoten: Operatoren

Blätter: Variablen, Konstanten



2. Beispiel



$$\frac{x\cdot y + u\cdot v - 2s\cdot t}{x\cdot z - (u+v)}$$

Nachtrag:

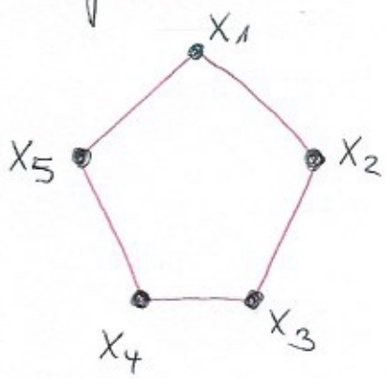
Def: Isomorphie Graphen

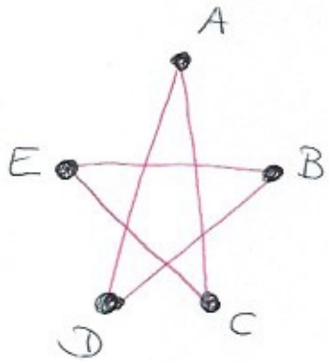
Zwei Graphen heißen isomorph, wenn es eine bijektive Abbildung φ von der Knotenmenge des einen Graphen G in die Knotenmenge des anderen Graphen G' gibt, so dass gilt:

$$\{x_i, x_j\} \text{ Kante von } G \iff \{\varphi(x_i), \varphi(x_j)\} \text{ Kante von } G'$$

Damit werden auch die Kantenmengen bijektiv aufeinander abgebildet.

Ein schönes Beispiel für auf den ersten Blick recht unterschiedlich aussehende Graphen finden Sie im Skript von Prof. Konen:





Die bijektive Abbildung kann nur zwischen zwei gleichmächtigen Knoten- und Kantenmengen definiert werden. Das ist also das erste, was zu überprüfen ist, wenn man eine solche bijektive Abbildung sucht.

Überprüfen Sie, ob folgende Abbildung die Isomorphie zwischen den beiden Graphen ausdrückt im obigen Beispiel.

$$\varphi(x_1) = A \quad \varphi(x_2) = C \quad \varphi(x_3) = E$$
$$\varphi(x_4) = B \quad \varphi(x_5) = D$$

Die Abbildung der Kanten kann entsprechend definiert werden.

Nachtrag zu den Adjazenzmatrizen:

Adjazenzmatrix A eines gerichteten Graphen

Mit Hilfe der Potenzen $A^r = A \cdot \ldots \cdot A$ lassen sich Aussagen über die Existenz und über die Anzahl von Wegen (Pfeilfolgen) in Digraphen machen.

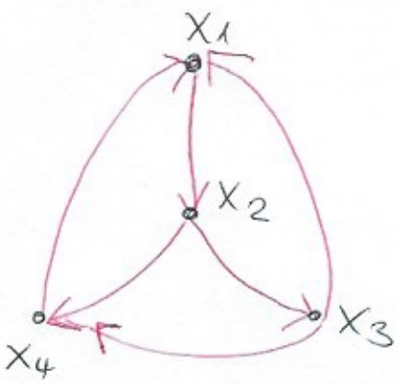
Zur Erinnerung: A Adjazenzmatrix mit $a_{ij} = \begin{cases} 1 & \text{wenn Pfeil von } x_i \text{ nach } x_j \\ 0 & \text{sonst} \end{cases}$

Sei $A^r = A \cdot \ldots \cdot A$ r-te Potenz von A

Das Element $a_{ij}^{(r)}$ der Matrix A^r (steht in der i-ten Zeile und der j-ten Spalte)

gibt die Anzahl verschiedener Pfeilfolgen der Länge r von x_i nach x_j in Graphen an. Der Beweis erfolgt mit dem Beweisverfahren der vollständigen Induktion und soll an dieser Stelle nicht gemacht werden.

Bp:



$$\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$A^2 = A \cdot A = \begin{pmatrix} 0&1&0&0 \\ 0&0&1&1 \\ 1&0&0&1 \\ 1&0&0&0 \end{pmatrix} \begin{pmatrix} 0&1&0&0 \\ 0&0&1&1 \\ 1&0&0&1 \\ 1&0&0&0 \end{pmatrix} = \begin{pmatrix} 0&0&1&1 \\ \boxed{2}&0&0&1 \\ 1&1&0&0 \\ 0&\boxed{1}&0&0 \end{pmatrix}$$

1 Weg der Länge 2 von x_4 nach x_2: $x_4 \to x_1 \to x_2$

2 Wege der Länge 2 von x_2 nach x_1: $x_2 \to x_3 \to x_1$, $x_2 \to x_4 \to x_1$

Wenn ein Digraph einen Zyklus enthält, dann werden die Potenzen A^r der Adjazenzmatrix niemals die Nullmatrix werden, da man ja beliebig oft den Zyklus durchlaufen kann.

Ein Digraph mit n Knoten ist genau dann azyklisch (= zyklenfrei), wenn es eine Zahl r mit $1 \leq r \leq n$ gibt mit $A^r \neq 0$ und $A^s = 0$ für alle $s > r$. Dabei ist 0 die Nullmatrix.

Durchlaufen von Graphen

Nachdem wir nun die Graphen mit all den verschiedenen Eigenschaften kennengelernt haben und auch schon an der ein oder anderen Stelle von Durchlaufen von Graphen gesprochen haben, es sei ganz am Anfang des Kapitels Graphentheorie erinnert, an L. Euler und seinen gesuchten Weg durch Königsberg. Das Königsberger Brückenproblem ist auch ein "Durchlaufen" von Graphen. Bei zahlreichen anderen Anwendungen muss man von einem bestimmten Knoten zu einem bestimmten anderen Knoten gelangen, entlang einer Kantenfolge (auch bereits definiert). Über die Adjazenzmatrix und ihre Potenzen kann man Aussagen über die Anzahl und Existenz von Wegen vorgegebener "Länge" machen.
Für viele dieser Anwendungen vor allem in großen, unübersichtlichen Graphen, wird eine Suche mit System gefordert. Aufgabenstellungen wie z.B. bei der Suche nach einem Eulerweg ("Jede Kante nur einmal") müssen mit System angegangen werden.
Unnötige Kanten weglassen und dennoch von einem zum anderen oder besser zu allen anderen Knoten kommen, das wird zunächst einmal die Forderung sein.
Dazu benötigen wir noch einmal ein paar Begriffe aus der Graphentheorie.

Def.: (aufspannender Baum, Gerüst)

Ein zusammenhängender Teilgraph eines zusammenhängenden Graphen mit minimaler Kantenzahl heißt aufspannender Baum, Gerüst oder Spannbaum.

Bem: die Knotenzahl ist bei einem aufspannenden Baum unverändert, es werden nur Kanten weggelassen (s. Def. Teilgraph Blatt 82)

Bem: Ein aufspannender Baum enthält keinen Kreis bzw. Zyklus und ist auf alle Fälle ein Baum. Er ist auf alle Fälle zusammenhängend.
Im englischen Sprachraum wird dafür der Begriff spanning tree verwendet.

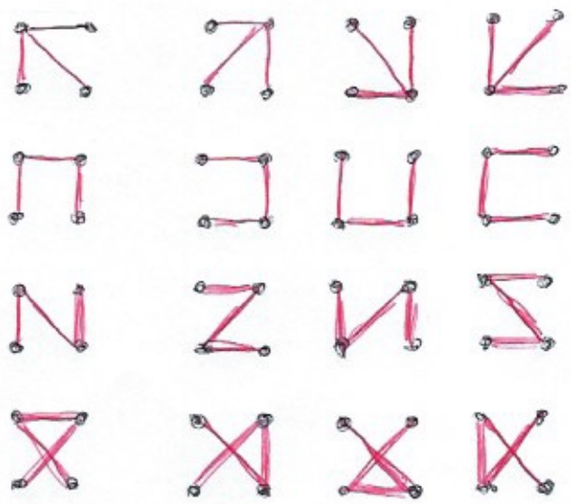
Aufgabe: Wie viele Spannbäume gibt es in einem vollständigen Graphen mit 4 Knoten.

Cayley-Formel:
Ein vollständiger Graph mit n Knoten hat

n^{n-2}

aufspannende Bäume

hier $n = 4$ also $4^2 = 16$



Des Beweis der Cayley-Formel soll hier nicht erbracht werden.

Wie im Eingangsbeispiel an der Karte mit den Städten rund um Köln zu sehen war, waren die Kanten, die Straßenverbindungen repräsentierten, bewertet. Einen bewerteten Graphen haben wir auf Blatt (84) definiert.
Wenn man aufspannende Bäume bewerteter Graphen betrachtet, dann werden besonders die von Interesse sein, die z.B. in der Summe das kürzeste Streckennetz, die geringsten Kosten usw. widerspiegeln.
Das führt zu folgender Definition:

Def.: Minimal aufspannender Baum (MST)

minimum spanning tree

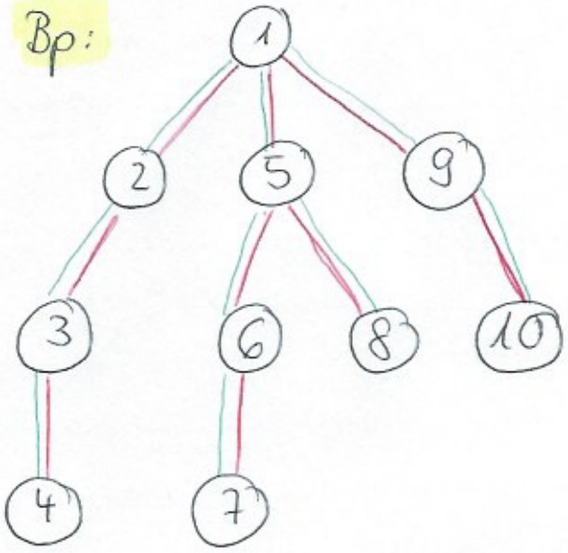
T sei ein aufspannender Baum eines bewerteten Graphen G. T heißt minimal aufspannender Baum, wenn die Summe seiner Kantenbewertungen minimal ist.

Zunächst einmal soll unabhängig von bewerteten Kanten untersucht werden, mit welchen Suchalgorithmen man mit System in großen, unübersichtlichen Graphen, Wege zu finden, jeden Knoten mindestens einmal zu besuchen.
Man hat zwei Möglichkeiten, nach Auswahl eines Startknotens vorzugehen:

1) Tiefensuche

Startknoten festlegen, als besucht markieren
Suche einen Nachfolgerknoten, markieren, wieder Nachfolger suchen. Wenn kein Nachfolger mehr vorhanden, Rückkehr zum zuletzt besuchten Knoten, der noch nicht alle Nachfolger markiert hat.

Bp:



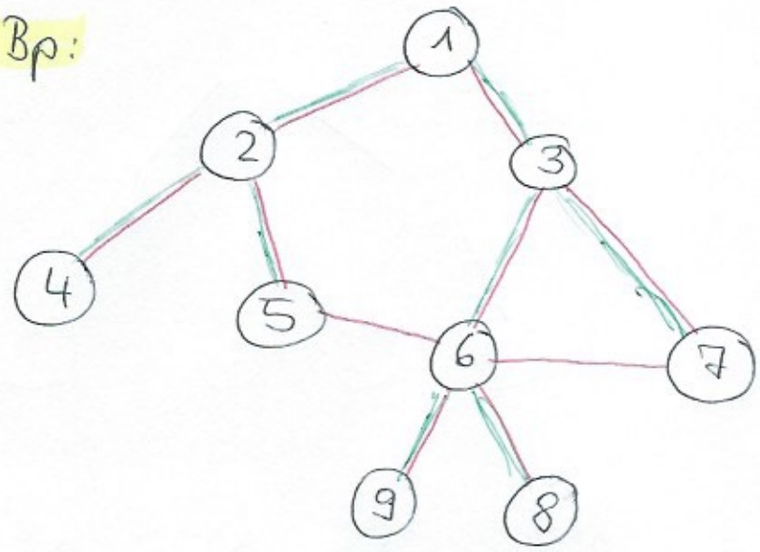
Tiefensuche: Startknoten (1)

Nachfolger (2)
Nachfolger (3)
Nachfolger (4) kein Nachfolger
zurück nach (1)
Nachfolger (5)
Nachfolger (6)
Nachfolger (7) kein Nachfolger
zurück nach (5)
Nachfolger (8) kein Nachfolger
zurück nach (1)
Nachfolger (9)
Nachfolger (10) kein Nachfolger
alle Knoten besucht

2) Breitensuche

Startknoten festlegen, als besucht markieren
Alle Nachfolgerknoten markieren ("Breite")
von diesen wieder alle Nachfolgerknoten, bis alle Knoten markiert sind.

Bp:



Breitensuche: Starthnoten (1)
Markiere (2) und (3)
Markiere bei (2) (4) und (5)
bei (3) (6) und (7)
Markiere bei (6) (9) und (8)
alle Knoten besucht

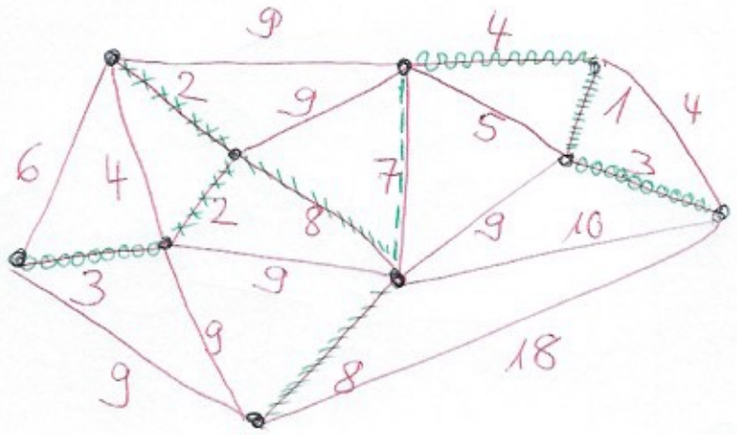
Frage: Wie findet man einen minimal aufspannenden Baum?

Mit dem Algorithmus von Kruskal hat man eine klar formulierte, relativ einfache Lösung.
Joseph Kruskal (1928–2010) veröffentliche diesen Algorithmus 1956.
Der Algorithmus ist anzuwenden auf einen ungerichteten bewerteten Graphen.

1) Suche in G (bewertet) die Kante mit dem kleinsten Gewicht, falls die Kante mit den schon markierten einen Kreis bildet, verwerfen, ansonsten hinzufügen
2) Dieser Schritt wird so oft wiederholt, bis nichts mehr hinzugefügt werden kann.

Bp.:

(1)



Algorithmus von Kruskal:

1) Kante mit dem kleinsten Gewicht : 1
2) Kante mit dem kleinsten Gewicht : 2, 2
3) Kante mit dem kleinsten Gewicht : 3, 3
4) Kante mit dem kleinsten Gewicht : 4
   die anderen beiden gehen nicht (Zyklus)
5) Kante mit dem kleinsten Gewicht : 5 (Zyklus)
6) Kante mit dem kleinsten Gewicht : 6 (Zyklus)
7) Kante mit dem kleinsten Gewicht : 7
8) Kante mit dem kleinsten Gewicht : 8

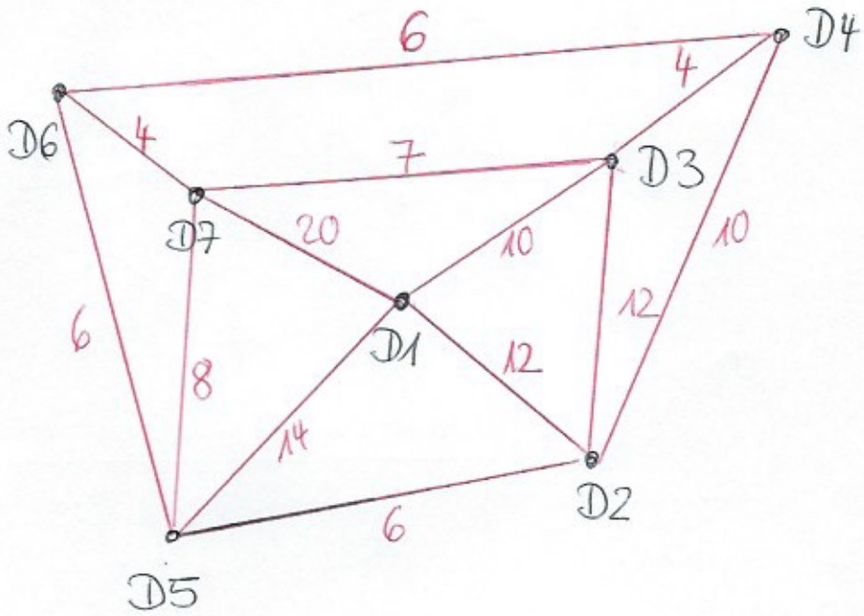
Alle weiteren in Frage kommenden Kanten bilden Kreise
Durch Addition der markierten Kantenwerte erhält man das Minimum: 38

(2)

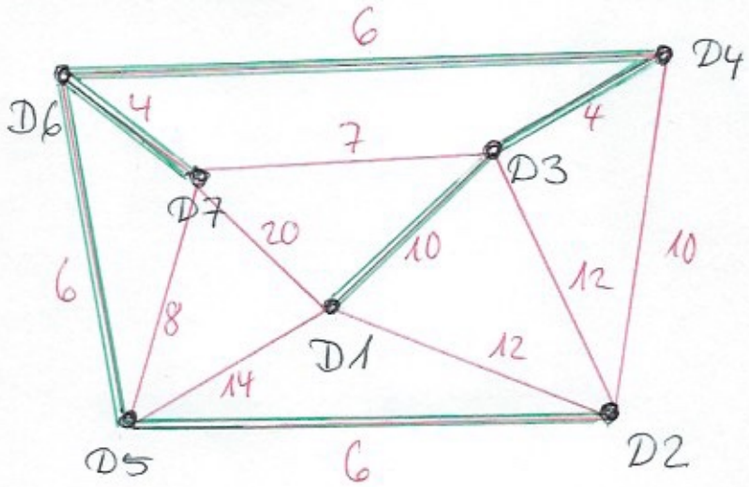
Sieben Dörfer sollen mittels Straßen verbunden werden. Für diesen Bau soll möglichst wenig Geld ausgegeben werden. Zeichnen Sie den Graphen und ermitteln Sie die minimalen Kosten.

	D1	D2	D3	D4	D5	D6	D7
D1	0	12	10	0	14	0	20
D2	12	0	12	10	6	0	0
D3	10	12	0	4	0	0	7
D4	0	10	4	0	0	6	0
D5	14	6	0	0	0	6	8
D6	0	0	0	6	6	0	4
D7	20	0	7	0	8	4	0

Der (kreuzungsfrei) gezeichnete Graph:



1) Man wählt zuerst die Kanten D7, D6 und D3, D4 mit den Kantenbewertungen 4 aus
2) Man wählt nun D6, D5 und D4, D6 und D5, D2 Kantenbewertungen 6 aus
3) Die Kante mit der Bewertung 7 kann nicht gewählt werden, da es einen Kreis geben würde
4) Die Kante mit der Bewertung 8 kann auch nicht gewählt werden, da es einen Kreis geben würde
5) Man wählt D3, D1 mit Bewertung 10



Die Summe der Kantenbewertungen beträgt: 36