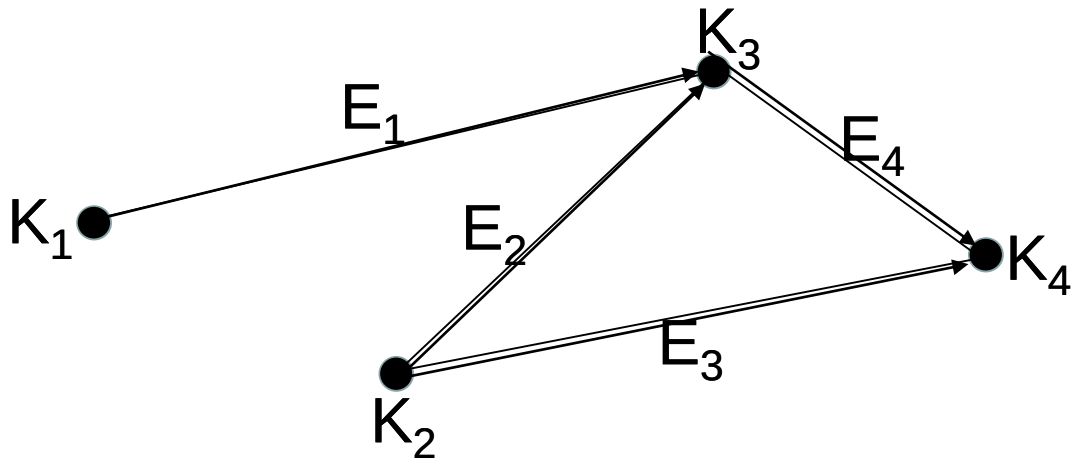
Vorkurs

-- 6. Ordnungsrelationen –

8.10.2018

Graphen

Graph besteht aus Knoten (Ecken) und Kanten (Verbindungen zwischen Knoten)



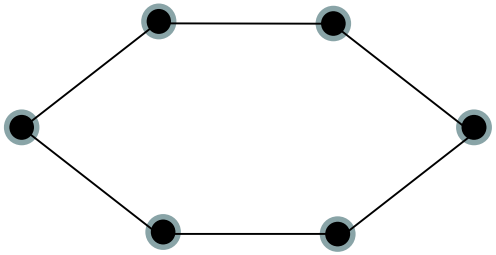
$E_1=(K_1,K_3)$, $E_2=(K_2,K_3)$, $E_3=(K_2,K_4)$, $E_4=(K_3,K_4)$,

$K=\{K_1,K_2,K_3,K_4\}$, $E=\{E_1,E_2,E_3,E_4\}$, Graph $G=\{K,E\}$.

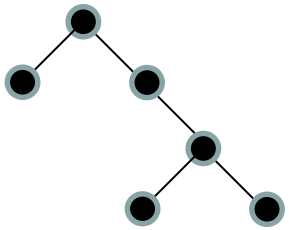
Graphen

Liste:

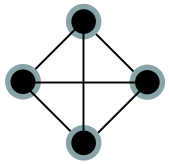
Schleife:



Baum:



Clique:



Graphen in der Informatik:

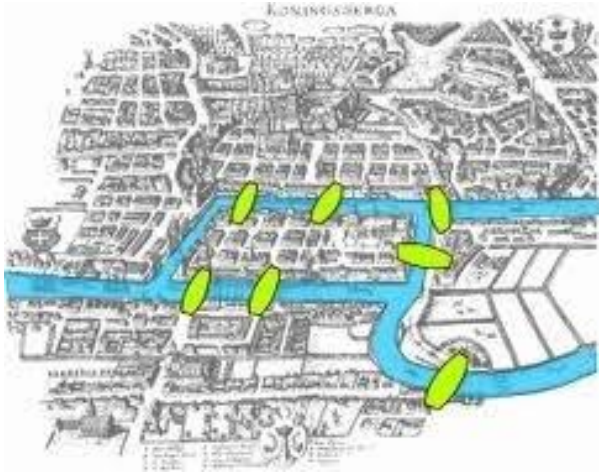
-Datenstrukturen

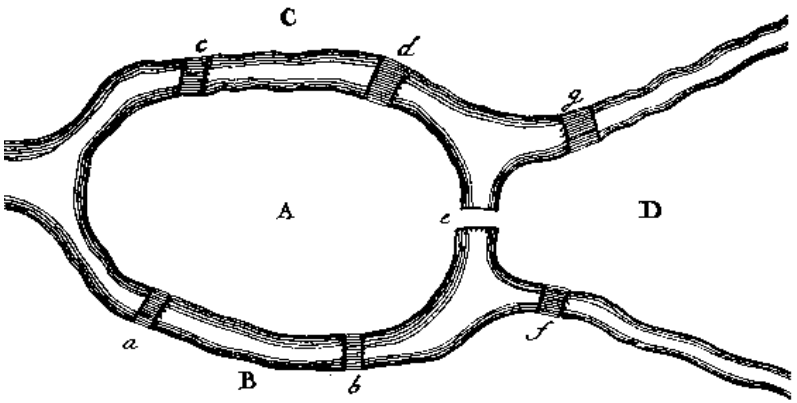
-Rechnerverbindung

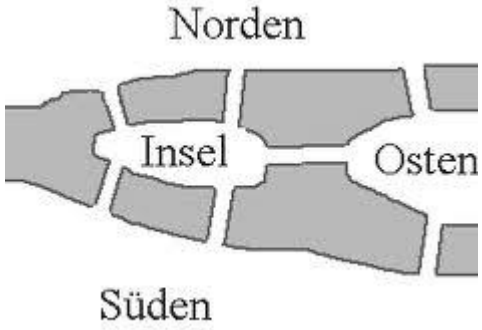
-Programmabhängigkeit

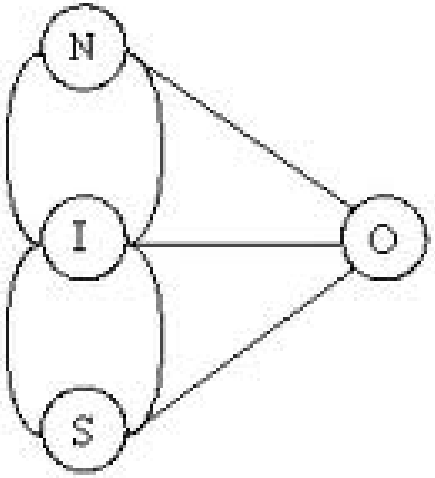
-Programmaufrufe

Königsberger Brückenproblem:









Ordnungsrelationen

Wir wollen Ordnung (Struktur) in eine Menge bringen, in dem wir jeweils zwei Elemente durch eine Relation vergleichen, z.B. durch ≤ auf den reellen Zahlen.
kunstaufraeumen.ch/sites/default/files/shop/shop.htm

Welche besonderen Eigenschaften aus unserem Katalog sollte eine solche Relation dann haben:

Transitiv: Wenn wir x vor y einsortieren und y vor z, dann sollte auch x vor z landen.

Antisymmetrisch: Gilt für zwei Elemente xRy und yRx, so sind sie gleich, x=y.

Reflexiv: Für ≤ soll auch xRx gelten
(Für < gälte dies nicht)

Partielle Ordnung

Eine transitive, antisymmetrische und reflexive Ordnung heißt partielle Ordnung.

Transitiv: xRy und yRz → xRz,

Antisymmetrisch: xRy und yRx → x=y,

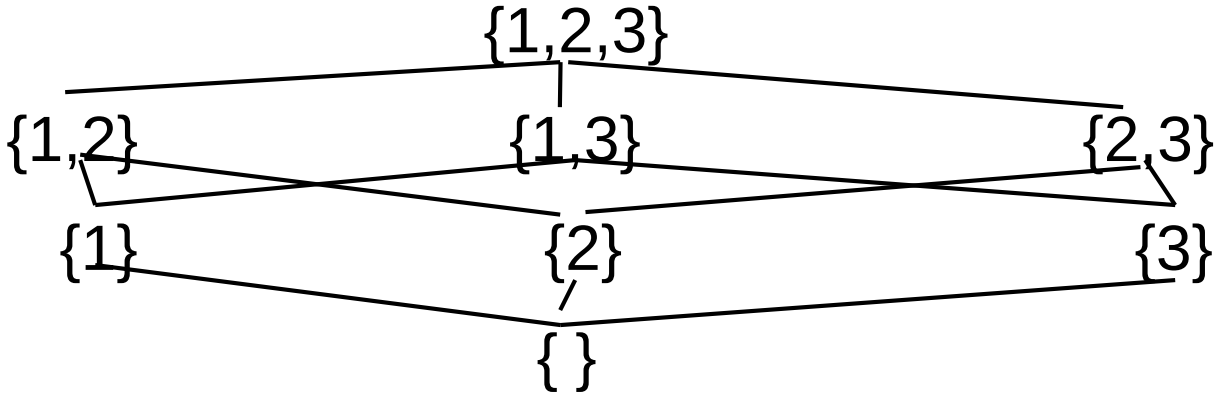
Reflexiv: xRx

Ordnung auf Gerade. Beispiel: “≤”

Partielle Ordnung

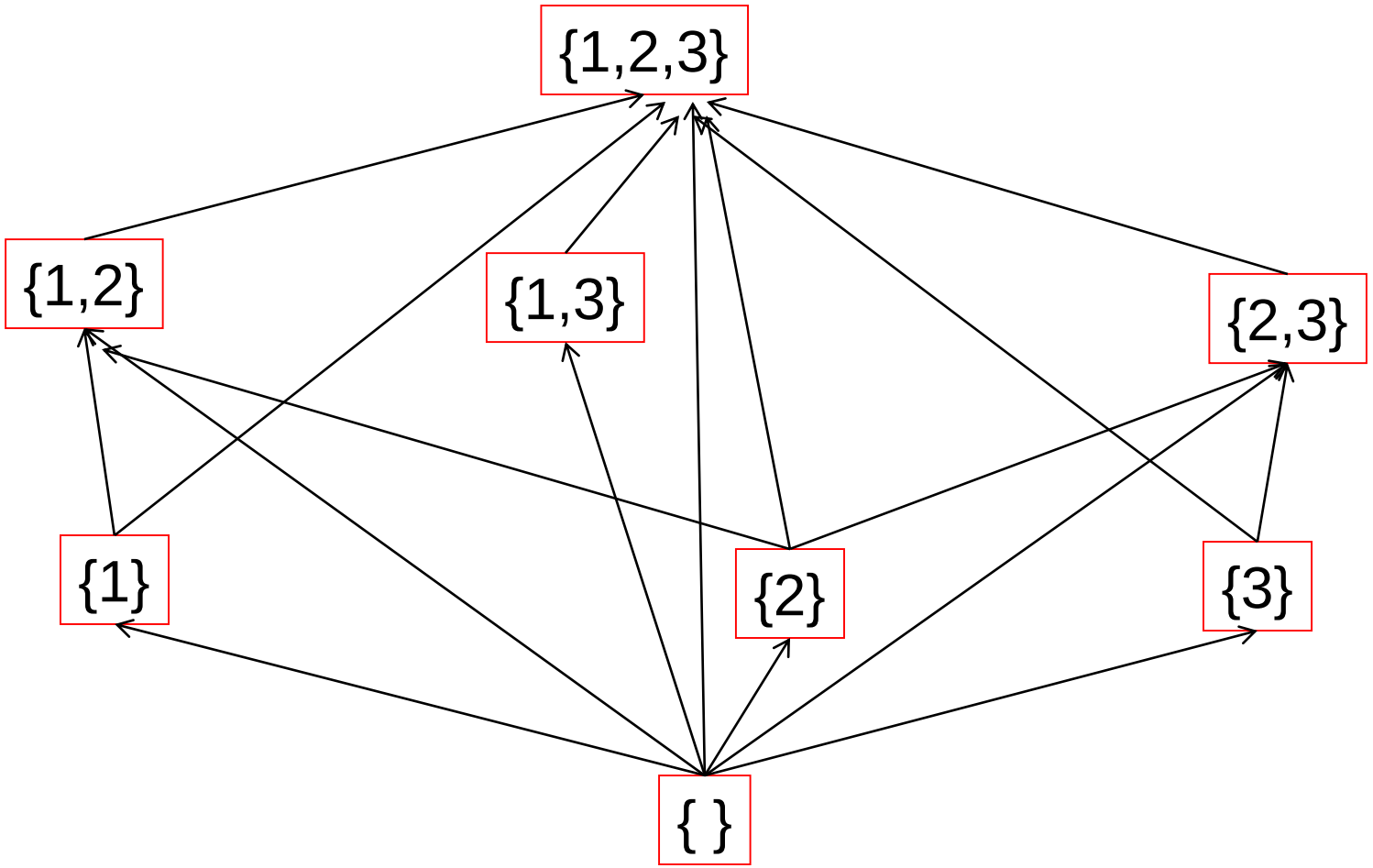
Eine transitive, antisymmetrische und reflexive Ordnung heißt partielle Ordnung.

Warum „partiell“?
Beispiel „Teilmenge“, z.B. A=P({1,2,3}) Potenzmenge von {1,2,3}.
Die Relation $\subseteq$ auf A ist eine partielle Ordnung auf A.



In unserem Beispiel gilt aber weder $\{1\} \subseteq \{2\}$, noch $\{2\} \subseteq \{1\}$
Dadurch ist diese Menge also noch nicht richtig geordnet!

Relationsgraph:



Graph ohne Zykel, bzw. Kreise!

Totale Ordnungen

Die Relation sollte also auch noch total sein.

Es sollte immer xRy oder yRx gelten.

Beispiel: Ordnung auf einer Geraden.

Eine totale partielle Ordnung nennt man (totale) Ordnung.

Totale ORDNUNG

Satz: In einer endlichen Menge $A=\{x_1,x_2,\ldots,x_n\}$, $n = |A|$, mit einer totalen Ordnung kann man die Elemente so nummerieren, dass gilt:

$$x_1 R x_2, x_2 R x_3, x_3 R x_4, \cdots, x_{n-1} R x_n$$

oder anders ausgedrückt

$$\forall i, j \in \{1,...,n\}: (i \leq j \Leftrightarrow x_i R x_j)$$

Für alle i,j von 1,..,n gilt: $i \leq j$ genau dann wenn $x_i R x_j$: Sortierung von A

Der folgende Beweis ist konstruktiv. Wir geben einfach ein Verfahren (einen Algorithmus) an, das diese Odnung herstellt (Quicksort)

Induktionsbeweis nach n.
Für n=0 und n=1 ist nichts zu beweisen (kein oder ein Element alleine ist immer sortiert)

Sei also 2≤n, A eine Menge mit |A|=n, und die Behauptung sei bereits bewiesen für alle Mengen B mit $0 \leq |B| < n$.

Wir wählen ein beliebiges zϵA (das Pivotelement) und zerlegen A in drei Mengen:

$$A_1 := \{x \in A : xRz \wedge x \neq z\}$$

$$A_2 := \{z\}$$

$$A_3 := \{x \in A : zRx \wedge x \neq z\}$$

Nun ist $A = A_1 \cup A_2 \cup A_3$, denn R ist total. Daher folgt

$$\forall x \in A : (xRz \wedge x \neq z) \vee (x = z) \vee (zRx \wedge x \neq z)$$

Außderdem sind die Mengen A_i paarweise disjunkt!

Paarweise disjunkt:

Widerspruchsbeweis:
Annahme, dass A_1, A_2 und A_3 nicht disjunkt sind.
Dann gibt es

- x aus A_1 und A_2, also $z \epsilon A_1$.
  Aber A_1 enthält nur Elemente verschieden von z!

- x aus A_1 und A_3, also xRz und zRx → Widerspruch Antisym.

- x aus A_2 und A_3, also $z \epsilon A_3$.
  Aber A_3 enthält nur Elemente verschieden von z!

Daher ist $|A| = |A_1| + 1 + |A_3|$, also auch $0 \leq |A_1| =: m < |A|$ und $0 \leq |A_3| =: n-1-m < |A|$.

Daher können wir nach Induktionsvoraussetzung die kleineren Mengen $A_1 = \{x_1,\ldots,x_m\}$ und $A_3 = \{x_{m+2},\ldots,x_n\}$ sortieren.

Dann ist aber auch A insgesamt in der Form

$$A = \{x_1,...,x_m, x_{m+1} = z, x_{m+2},...,x_n\}$$

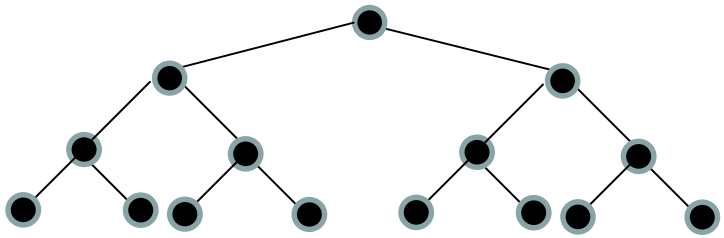
sortiert.

Rekursion

Dieser Induktionsbeweis lässt sich in ein rekursives Programm umsetzen, in dem man zur Sortierung der Teilmengen A_1 und A_3 das Programm selbst wieder aufruft.

Für den Beweis ist die Effizienz der Konstruktion egal. Ein echtes Sortierverfahren in der Praxis sollte aber effizient (schnell) sein.

Binärer Baum:



Andere Sortierverfahren

1) Wiederholt Maximum suchen

2) Der Reihe nach einordnen

3) Merge Sort

4) Bucket Sort

Unterschiedliche Kosten, induktiv oder rekursiv