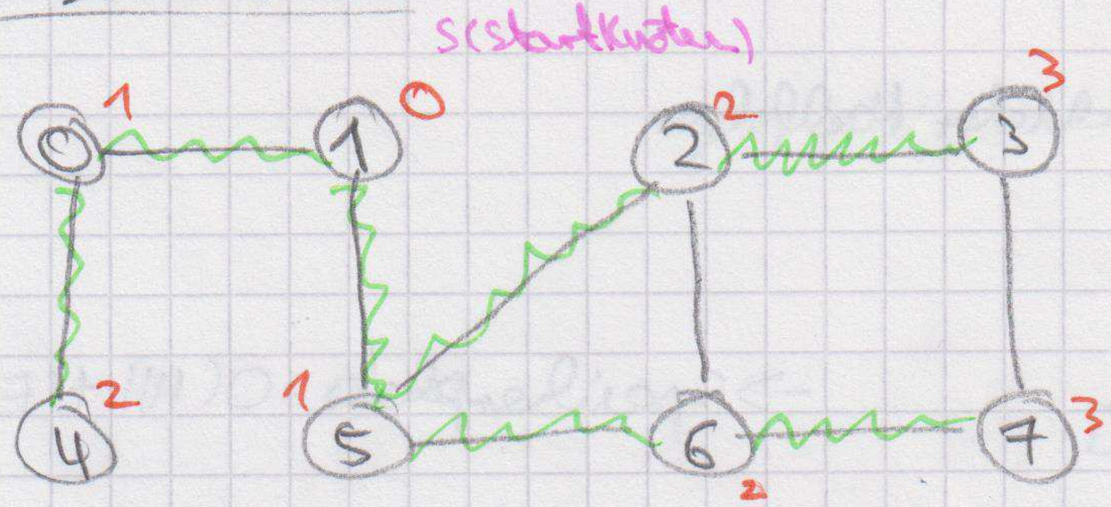
Bsp: Breitensuche



Attribut d

~ Kante des Breiten-Such-Baumes

→ Inhalt der Schlange: (graue Knoten)

d = 0: ~~1~~

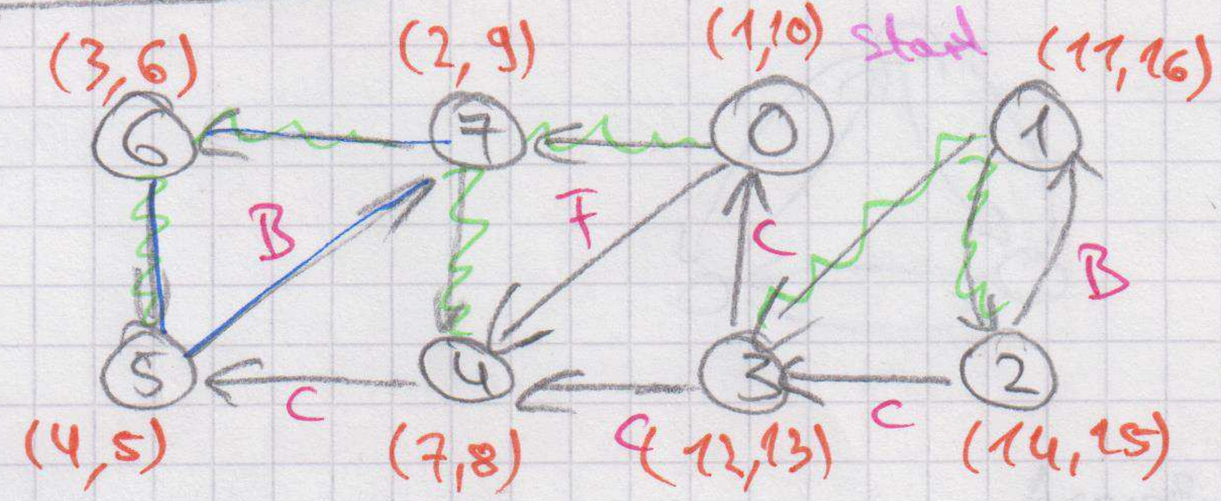
d = 1: ~~0~~ ~~5~~

d = 2: ~~4~~ ~~2~~ ~~6~~

d = 3: ~~3~~ ~~7~~

d = 4:

Bsp: Tiefensuche

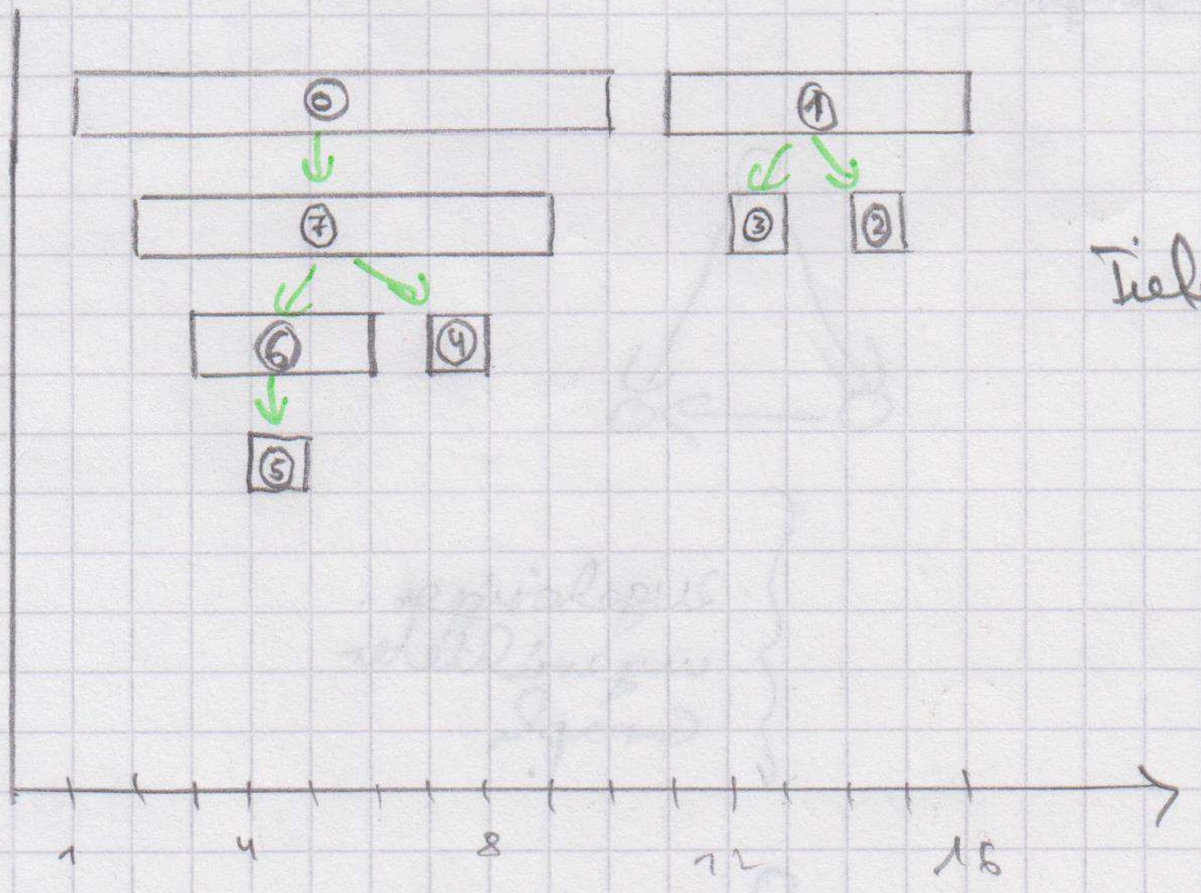


~ Kante des Tiefen Such.-Baumes

(d,f)

d: Entdeckungszeit

f: Endzeit



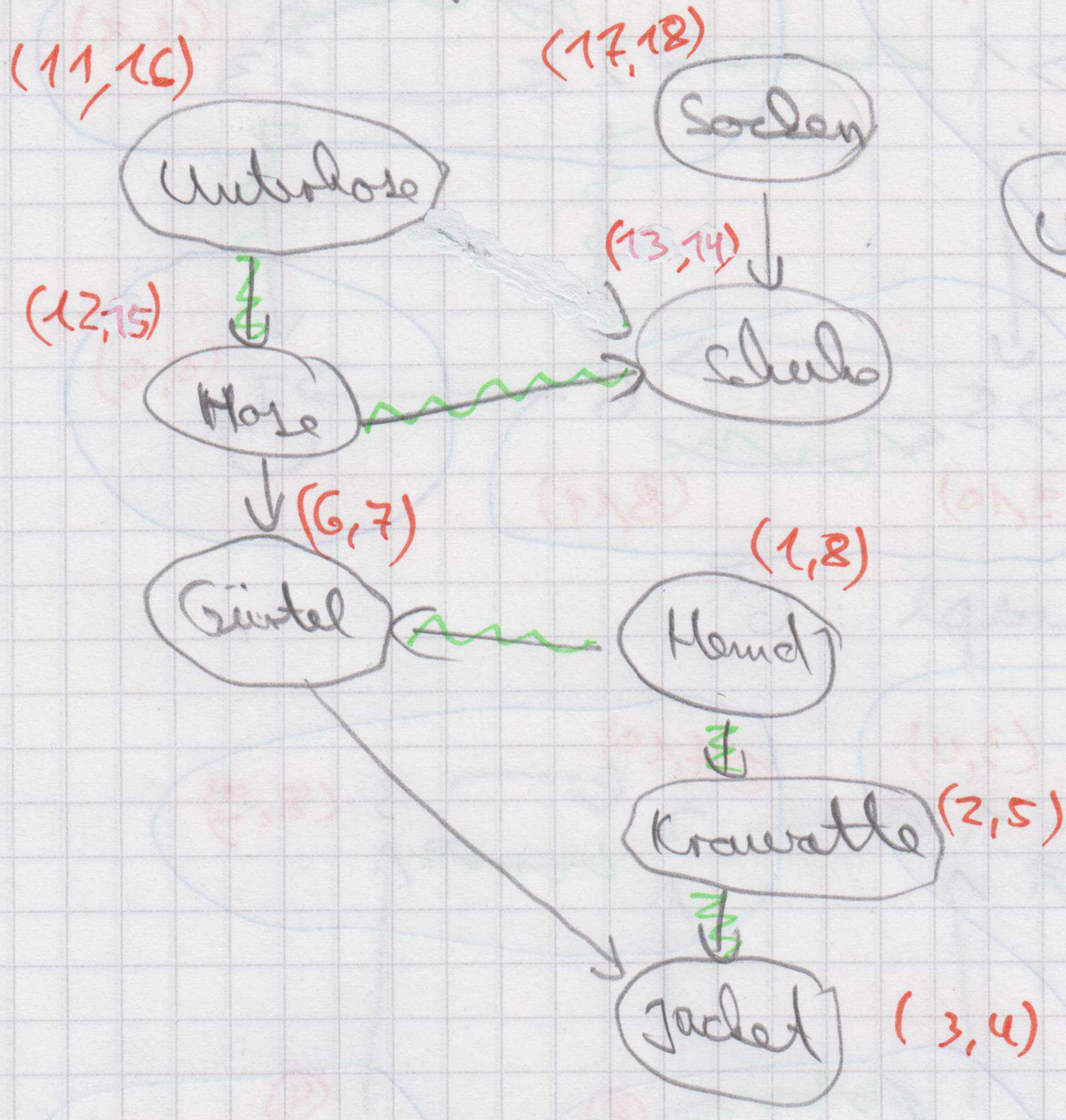
Tiefensuch-Wald

Klassifikation der Kanten:

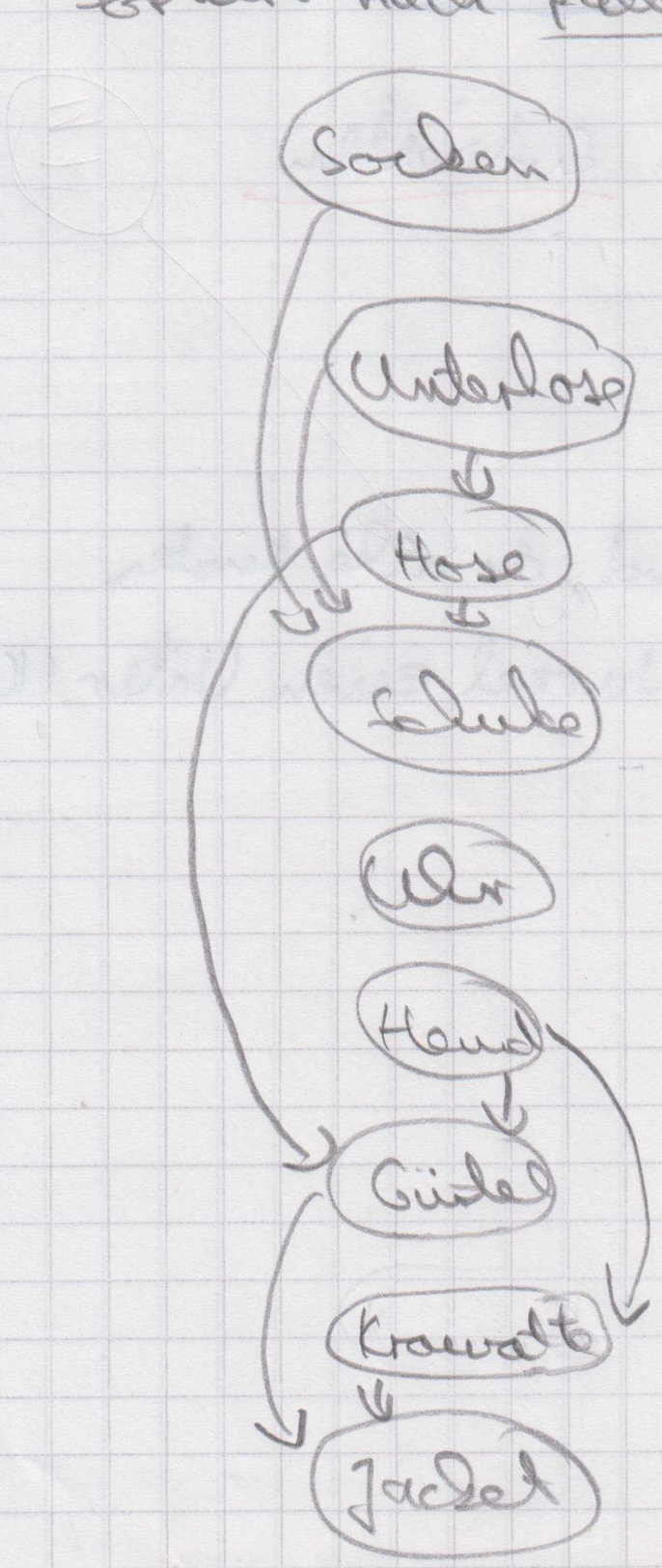
- Rückwärts-Kante (back) B → Graph nicht zyklenfrei
- Vorwärts-Kante (foward) F
- alle anderen Kanten: Crosskanten C

Bsp (Topologische Sortierung)

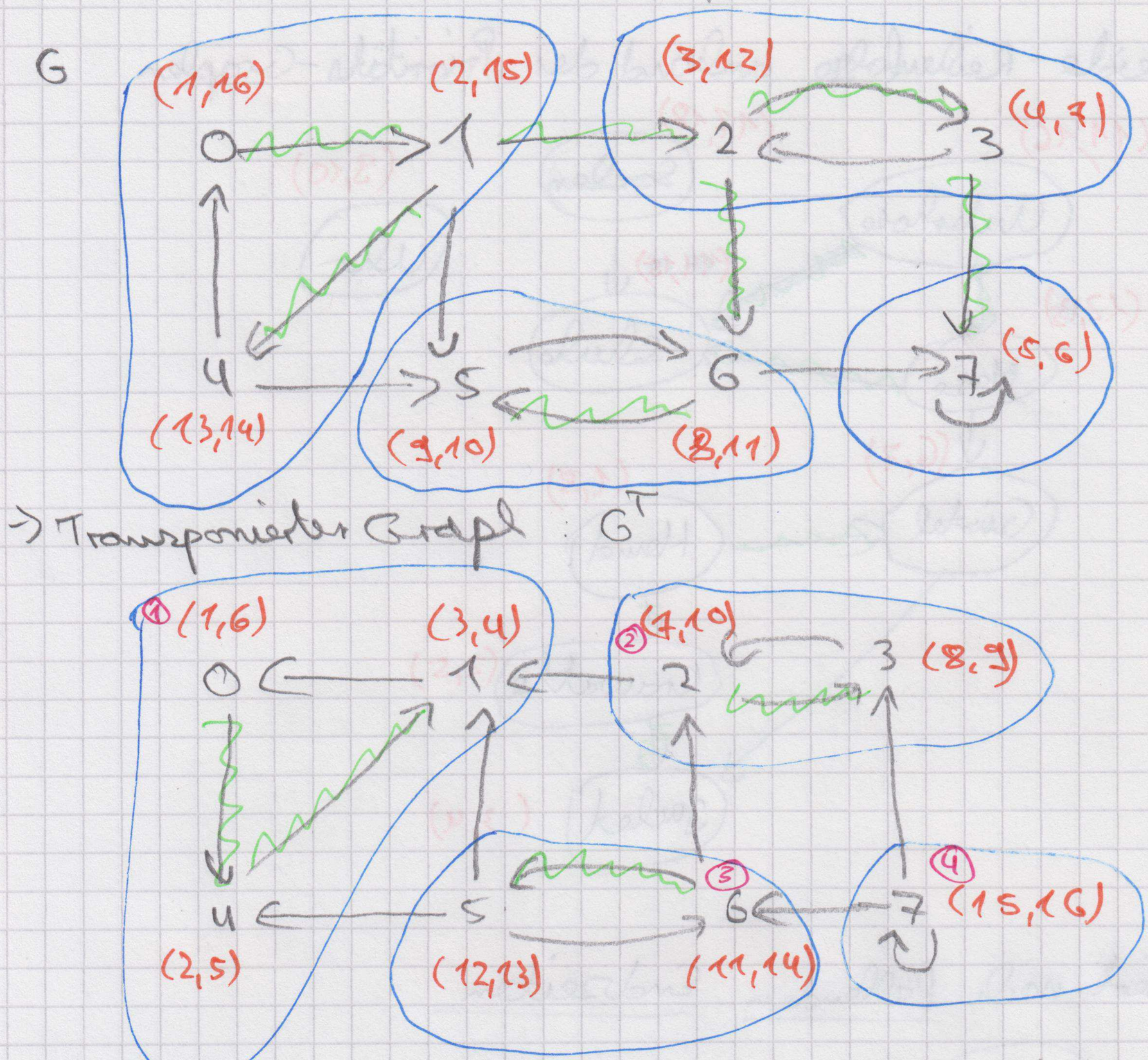
Anfangs-Reihenfolge anhand der Prioritäts-Graphen



sortiert nach fallenden Endzeiten



Bsp: starke Zusammenhangskomponenten



→ Transponierter Graph: G^T

Abarbeitung nach fallenden Endzeiten

Baum:

spezieller Graph: i) zyklenfrei

$|V| = n$

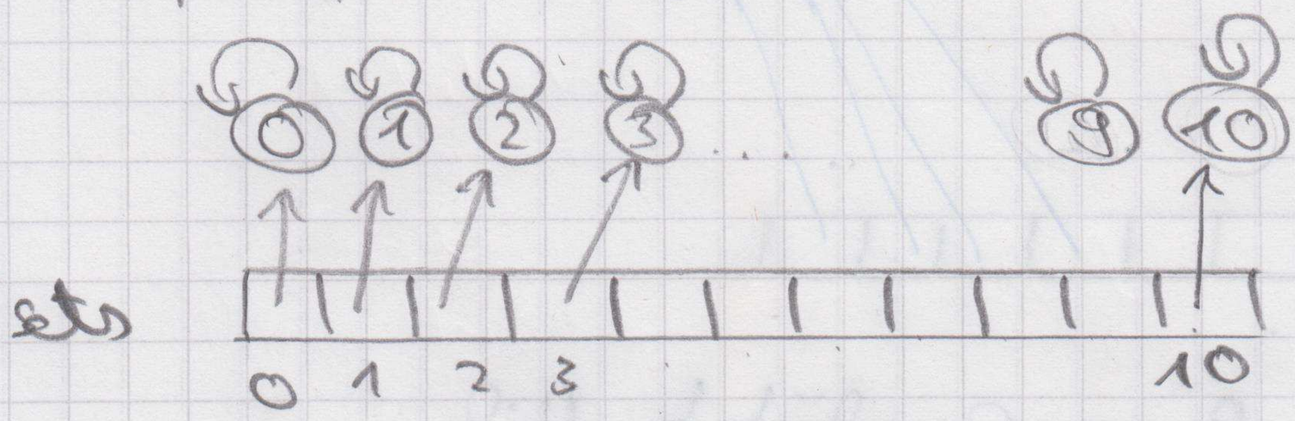
$|E| = n-1$

ii) zusammenhängend, da alle Knoten außer der Wurzel einen Vater (Kante) haben

Datenstrukturen für disjunkte Mengen (union and find)

Ausgangs-Situation: (11 disjunkte Mengen)

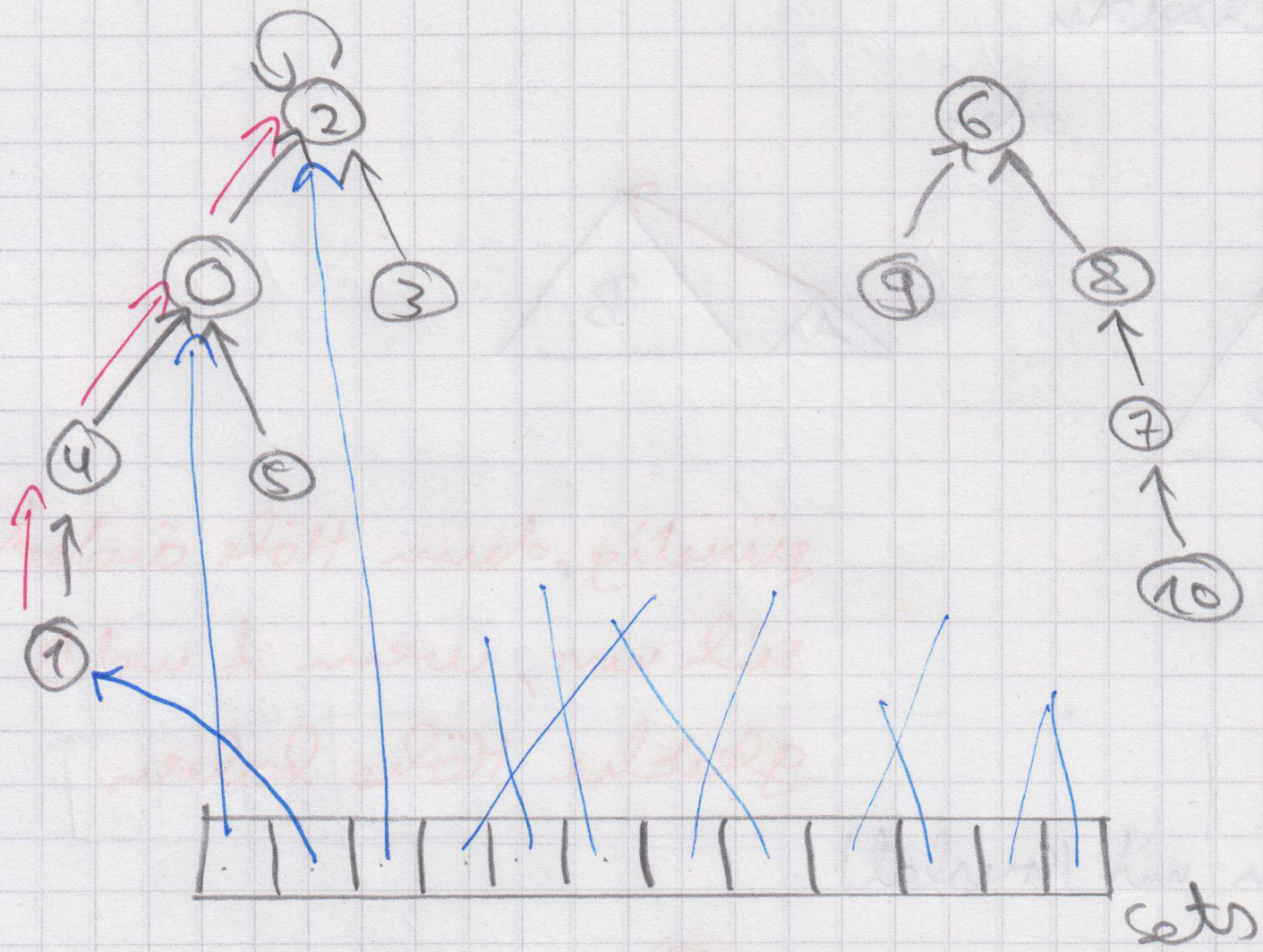
$\{0\}, \{1\}, \ldots \{9\}, \{10\}$



Nach Folge von Vereinigungs-Operation ist folgende Situation eingetreten:

$\{0,1,2,3,4,5\}$ Repräsentant 2

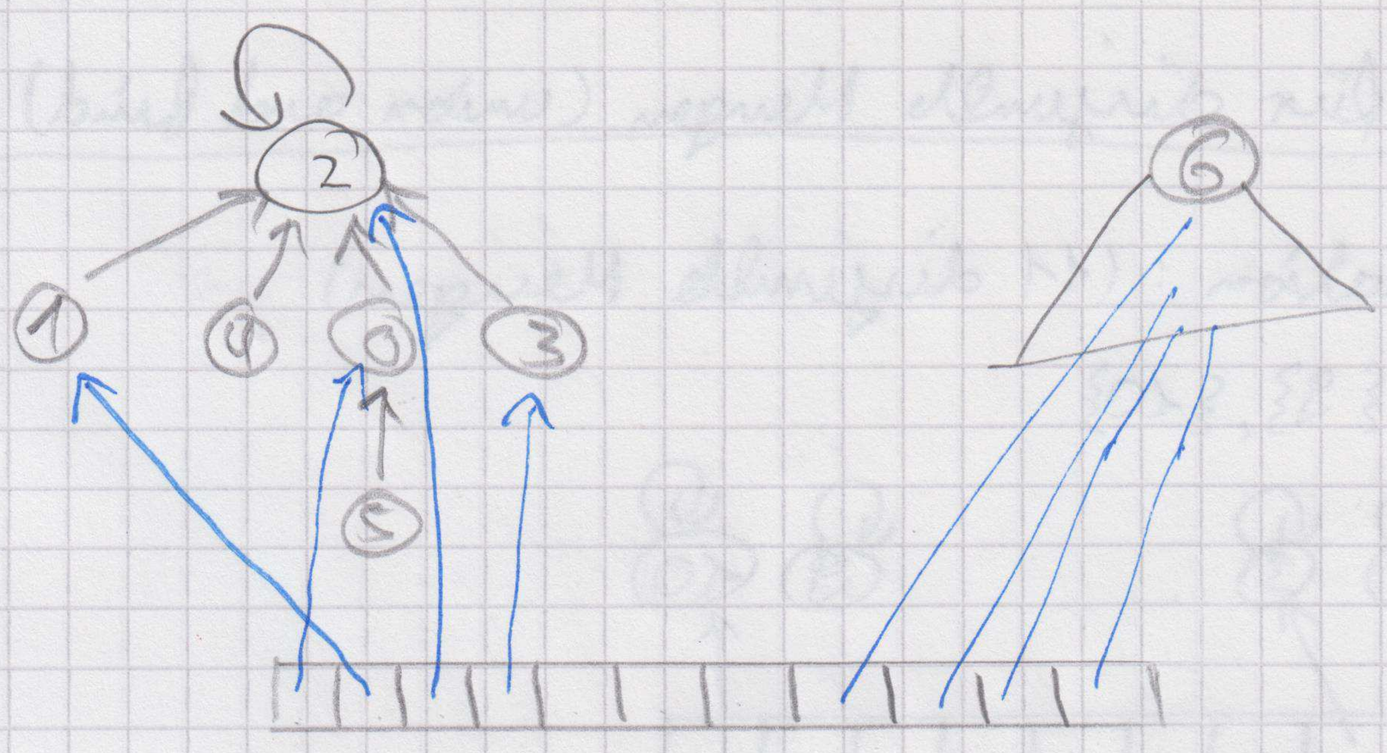
$\{6,7,8,9,10\}$ Repräsentant 6



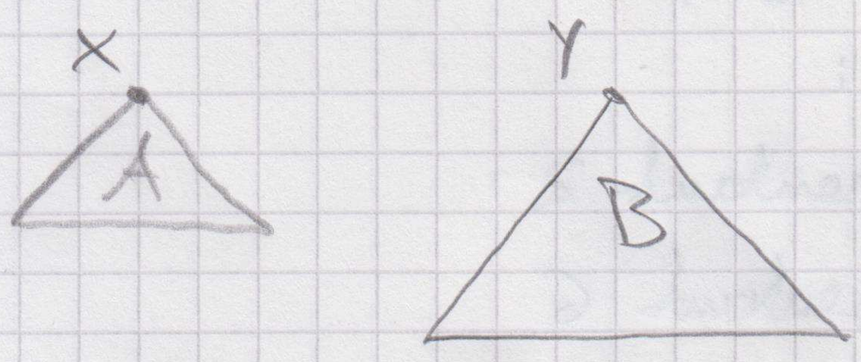
Find set (1)

Optimierung durch Pfadverkürzung: alle Knoten auf Bestimmungspfad erhalten Repräsentanten als Vater:

Es entsteht folgende Situation.

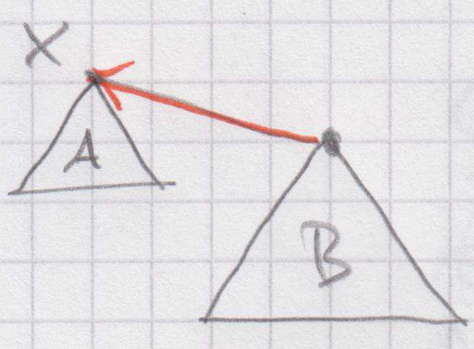


Vereinigung nach Rang: Gespeicherte Höhe



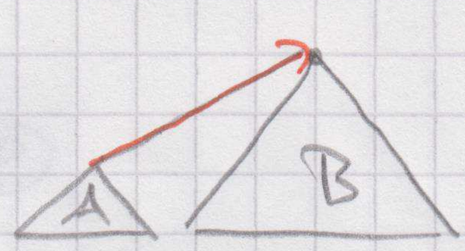
2-Möglichkeiten

entweder



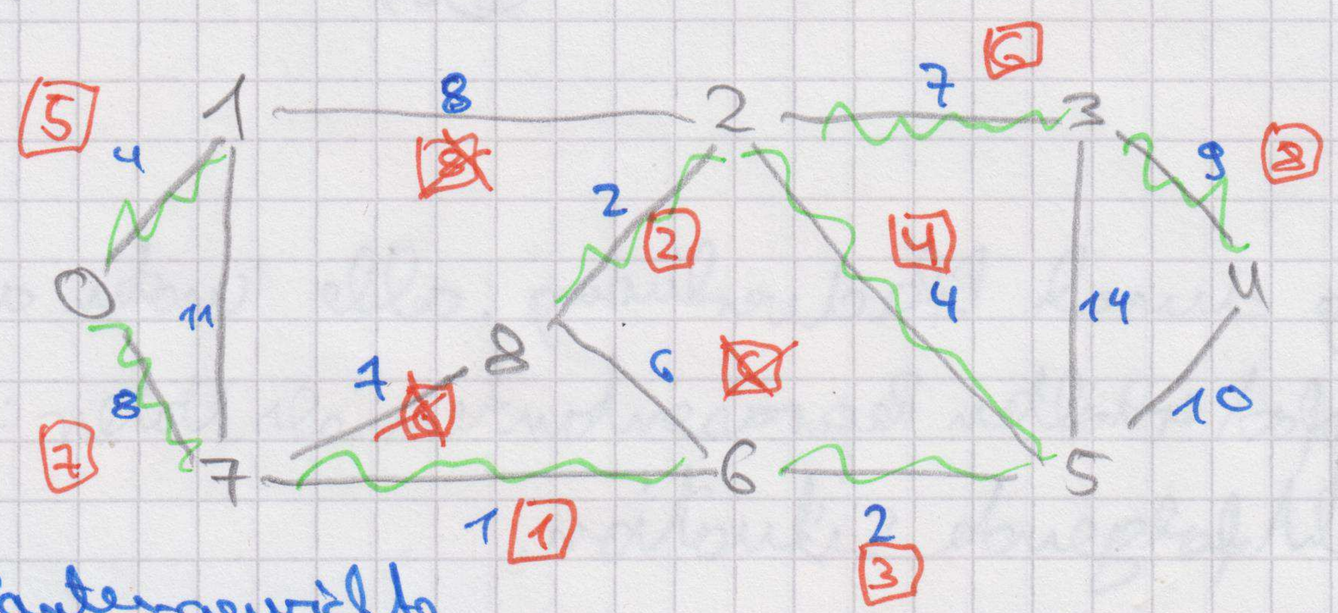
ungünstig

oder



günstig, denn Höhe ändert sich nur, wenn A und B gleiche Höhe haben

Bsp(Spannbaum mit Kruskal)



Kantengewicht

[K] Reihenfolge in der Kante gewählt wird

~ Kante des Spannbaumes

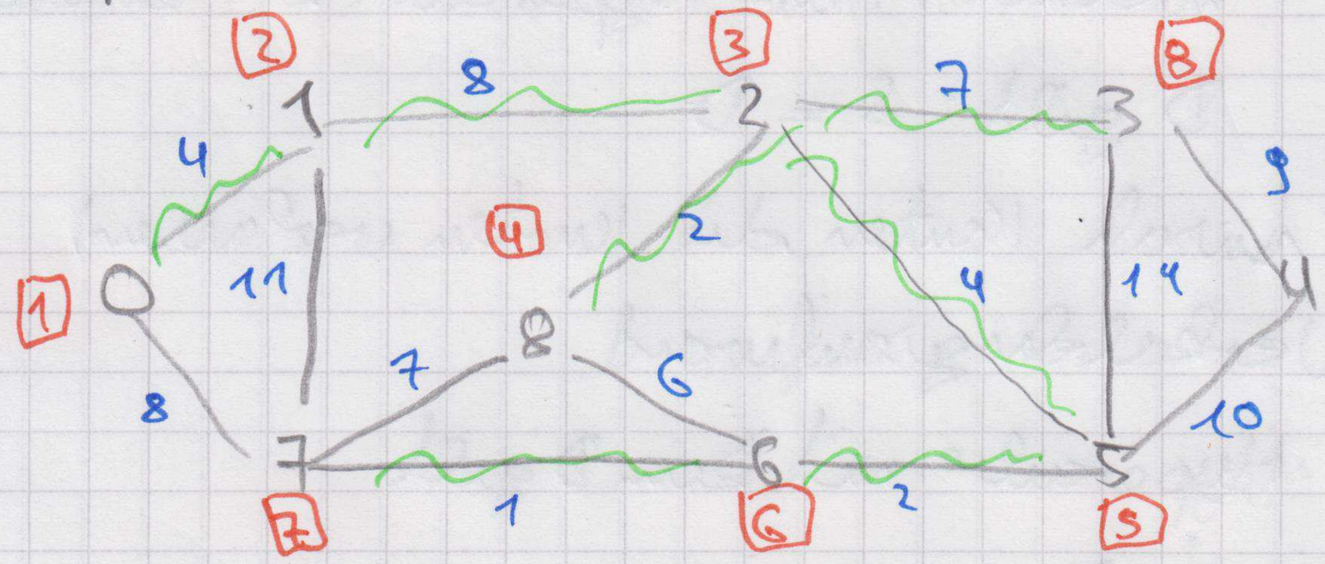
1 + 2 + 2 + 4 + 4 + 7 + 8 + 9 = 37

DS Disjunkte Menge:

Schnitt:

0 : {0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}
1 : " ... " .... {6,7}, "
2 : " ... {2,8} ..... " —
3 : " " {5,6,7} .. —
4 : " {2,5,6,7,8} " — —
5 : {0,1} " " " "
6 : " {2,3,5,6,7,8}
7 : " {0,1,2,3,5,6,7,8} {4}
8 : {0,1, ..., 8}

Bsp (Spannbaum mit Prim)



→ Priority-Queue Q

Key-Werte ≙ Minimaler Abstand zum Teil-Spannbaum

Schritte

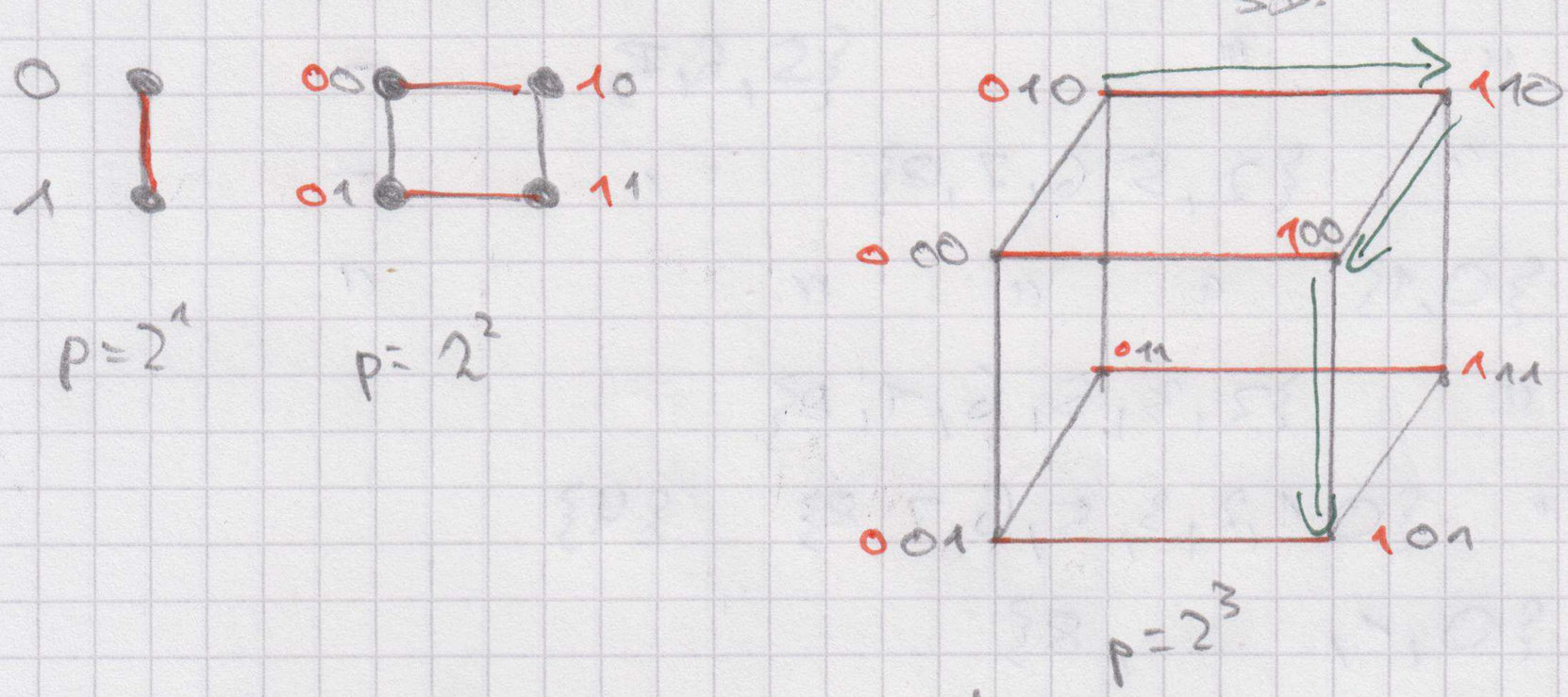
	∞	1	2	3	4	5	6	7	8	9
0	0	X								
1	8	4	X							
2	8		8	X						
3	8			7		()			X	
4	8					10			9	X
5	8			4		X				
6	8						6	2	X	
7	8	8	()			7		1	X	
8	8			2	X					

Anwendung Spannbaum:

Broadcast in Hypercube Architekturen

Knoten ≙ Computer mit eigenem Speicher

Kanten ≙ Netzwerk-Verbindung



$p = 2^1$ $p = 2^2$ $p = 2^3$

- Durchmesser δ ≙ max Abstand 2-er Knoten (Dauer einer Nachricht). Für Hypercube der Dimension D gilt: $\delta = D$
- Grad g (≙ Anzahl Kanten die Knoten verlassen) → Verkabelungsaufwand

  Hypercube mit Dim D gilt $g = D$

E-Cube-Routing

Start S = 010

Ende E = 101

Abstand (S, E) = Hamming-Distanz der Adressen 3

→ S = 010 → 110 → 100 → 101 E

root = (000) sendet Nachricht an alle anderen Knoten mit Spannbaum

- Wurzel = root
- Anzahl Kinder

100 ← 2 Kinder < 110, 101