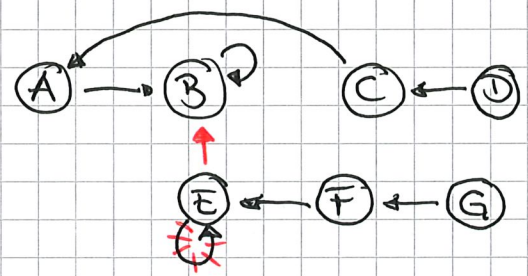
Kruskal und die Union-Find Datenstruktur

Test auf Kreisfreiheit

→ mit BFS/DFS in $O(n)$

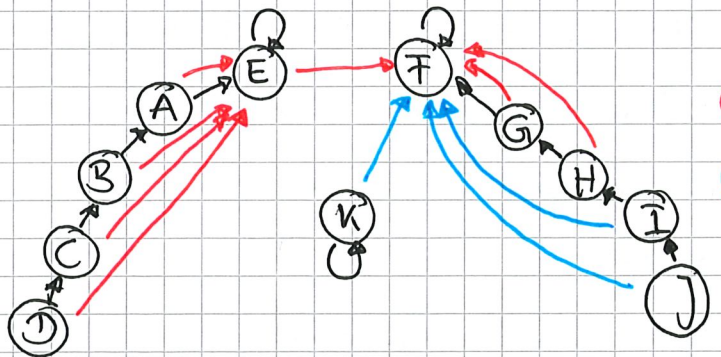
→ mit Union-Find in $O(\log n)$

Union-Find (Disjoint-Set) verwaltet eine Menge an Elementen (Zusammenhangskomponenten) und besitzt die Operationen find(x) und union(x,y).



Wir können die Laufzeit mit einem einfachen Trick stark verringern!

→ Pfadkompression. Haben wir schon mal Elemente gesucht, verändern wir auf dem Weg zur Wurzel alle Pointer auf diesem Pfad, so dass sie danach direkt auf die Wurzel zeigen. Dies verringert die Laufzeit für weitere find(x) Aufrufe!



Der Test auf Kreisfreiheit verringert sich dadurch (amortisiert) auf $O(\alpha(m,n))$, wobei $\alpha(m,n)$ die inverse Ackermannfunktion ist und $\alpha(m,n)$ für "praktische" Eingaben ($n \leq 10^{80}$) kleiner als 5 ist.

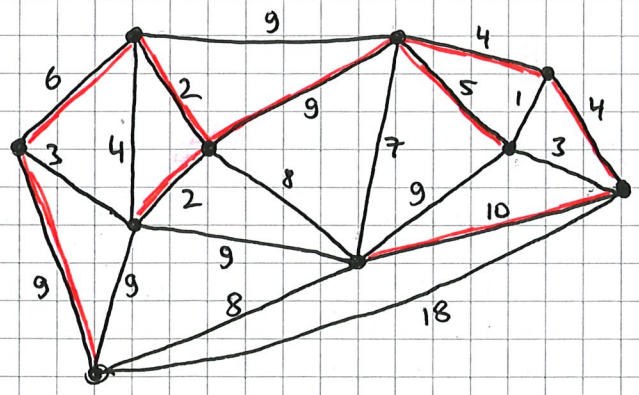
Die Laufzeit von Kruskal verbessert sich dadurch nicht direkt, weil immernoch die Kanten nach Gewicht sortiert werden müssen!

Es gibt allerdings einen Algorithmus von Bernard Chazelle für das MST-Problem mit Laufzeit $O(m\,\alpha(m,n))$!

Bottleneck Spanning Tree (BST)

Gegeben: Graph $G=(V,E)$ mit Kantengewichten $c: E \to \mathbb{R}^+$, $b \in \mathbb{R}^+$

Gesucht: Spannbaum, so dass jede Kante höchstens "b" lang ist.



BST mit $b=10$.

Geht es besser?

Entscheidungsproblem: "Gibt es einen BST mit Bottleneck höchstens "b"?"

→ entferne alle Kanten mit Gewicht $> b$

↳ Graph zusammenhängend? → JA!

→ sonst → NEIN!

Optimierungsproblem: "Finde einen BST mit kleinstmöglichen "b"!

→ Camerinis Algorithmus in $O(E)$ Zeit.

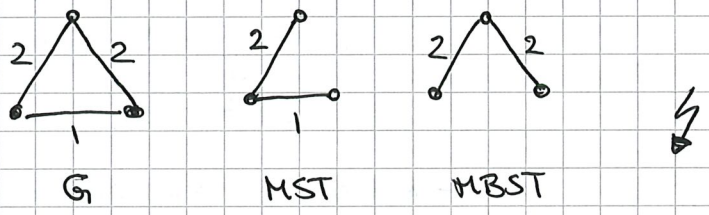
Satz. Ein MST ist ein MBST.

Beweis: Wir zeigen nicht direkt $\underbrace{T \text{ ist MST}}_{A} \Rightarrow \underbrace{T \text{ ist MBST}}_{B}$

sondern $\neg B \Rightarrow \neg A$.

Sei e eine Kante mit höchsten Gewicht in T und sei T kein MBST. Da T ein Baum ist, gibt es zwei Teilbäume T_1 und T_2 die durch e verbunden sind. Da T kein MBST ist, gibt es eine Kante f mit $c(f) < c(e)$ die T_1 und T_2 verbindet. Damit kann T kein MST sein. □

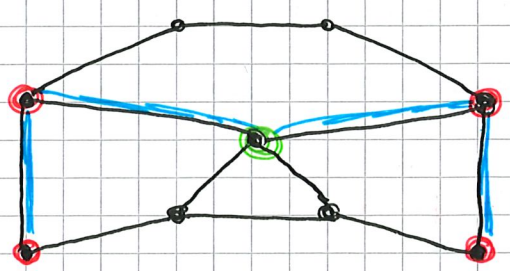
Und umgekehrt?



Steinerbaum Problem

Gegeben: Graph $G=(V,E)$ mit Kantenkosten $c: E \to \mathbb{R}^+$ und Knotenteilmenge $V' \subseteq V$ (Terminals)

Gesucht: Kantenteilmenge $F \subseteq E$ minimalen Gesamtgewichts, so dass alle Knoten V' verbunden/zshg. sind



Terminals V'

Steinerknoten

Steinerbaum F

Ist $|V'|=2$, ist dieses Problem ein kürzeste Wege Problem.

Ist $V'=V$, ist dies das MST Problem.

Das allgemeine Problem ist NP-schwer!

→ Einschub "Komplexität"

Unterteilung von Entscheidungsproblemen in verschiedene "Schwierigkeitsklassen"

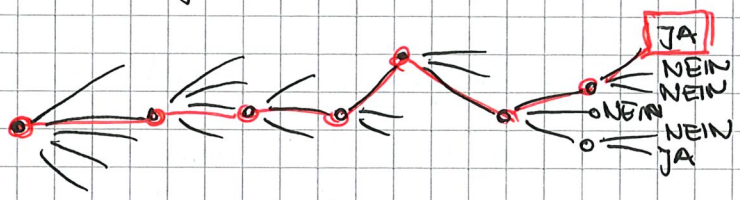
Klasse P: Klasse von Entscheidungsproblemen die in polynomieller Zeit entschieden werden können

Klasse NP: Klasse von Entscheidungsproblemen die in polynomieller Zeit von einem Algorithmus entschieden werden können, wenn der Algorithmus so richtig viel Glück hat!

Was heißt das?

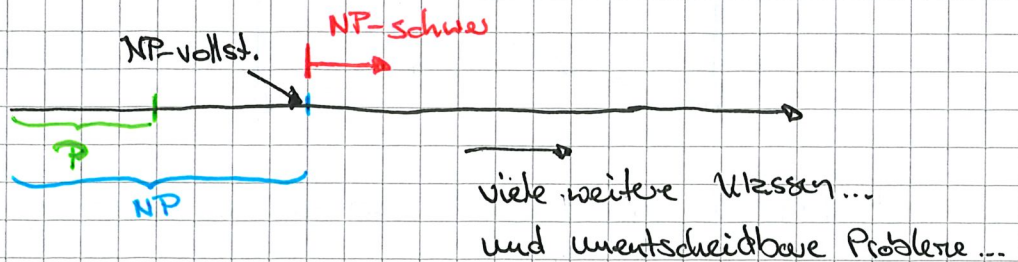
→ Non-deterministic model

- Algorithmus rät und antwortet mit JA/NEIN!
- das Raten garantiert JA! wenn JA! existiert



- dabei werden evtl. als bei dyn. Prog. nicht alle Lösungen probiert, sondern nur eine!

NP-schwer: Klasse von Entscheidungsproblemen die mindestens so schwer sind wie alle in NP

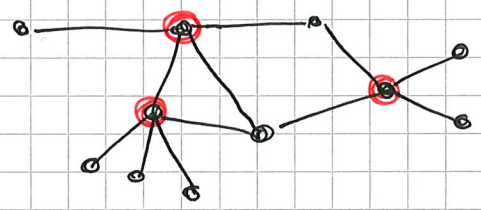


NP-vollständig: Problem $Q \in NP$ und Q ist NP-schwer

Probleme lassen sich aufeinander reduzieren!
Es genügt ein NP-schweres Problem zu lösen um alle zu lösen... und P=NP zu zeigen → $1.000.000

Problem: Vertex Cover

Gegeben: Graph $G = (V, E)$
Gesucht: Teilmenge $V' \subseteq V$ so dass jede Kante aus E einen Knoten in V' besitzt

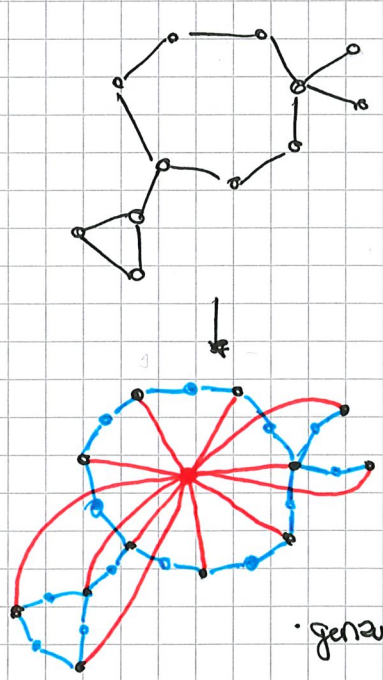


Vertex Cover der Größe 3.
Geht es besser?

Wir nutzen dieses Problem nun zu zeigen, dass das Steinerbaum Problem mindestens genauso schwer ist!

Nun die Idee:

Nehme Instanz von VC und baue den Graphen wie folgt um: Unterteile jede Kante einmal und füge einen zusätzlichen Knoten ein, der mit allen ursprünglichen Knoten verbunden ist.



Der neue Graph ist nun eine Instanz vom Steinerbaum Problem wobei die neuen Knoten die Terminals sind.

Nun kann man leicht argumentieren, dass genau dann einen Steinerbaum mit $|E| + k$ Kanten gibt, wenn es VC mit k Knoten gibt, und umgekehrt!