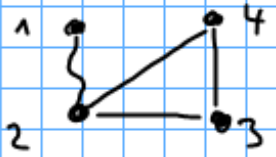
„Algorithmische Diskrete Mathematik"

VL 1, 20.10.11

1. Grundlagen der algorithmischen Graphentheorie

$\mathbb{N} = \{1,2,3,\dots\}$, $\mathbb{N}_0 = \{0,1,2,3,\dots\}$, $[n] = \{1,\dots,n\}$

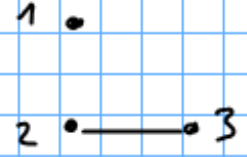
Graph $G = (V,E)$, z.B. $V = [4]$, $E = \{\{1,2\},\{2,3\},\{3,4\},\{2,4\}\}$



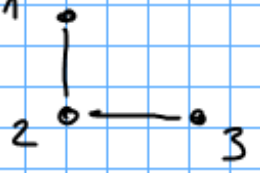
Nachbarschaft $N(4) = \{2,3\}$, $N(2) = \{1,3,4\}$

Grad $\deg(4) = 2$, $\deg(2) = 3$

Subgraph $H_1 \subset G$:

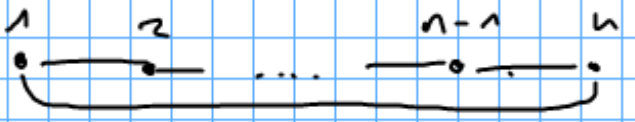


induzierter Subgraph $H_2 = G[\{1,2,3\}] =$



Weg der Länge n: $P_n =$

Kreis der Länge n: $C_n =$

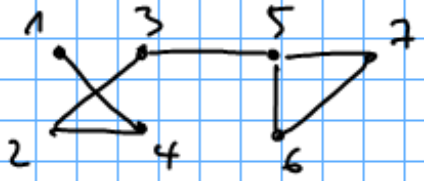


$\cong P_3$

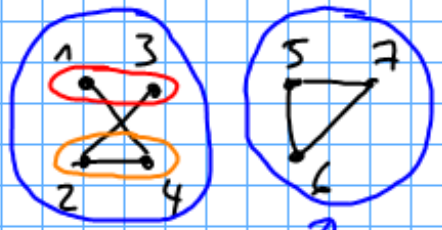
$\cong C_5$

Zusammenhängender Graph:

$\forall x,y \in V \quad \exists\, xy\text{-Weg}$



zshgd

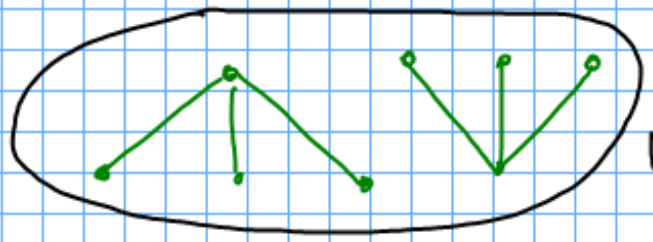


Zshgskomp.

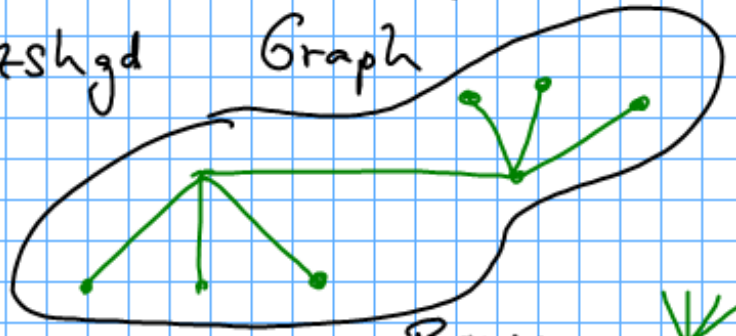
nicht zshgd

Wald := Kreisfreier Graph (d.h. Kein Subgraph ist Kreis)

Baum := Kreisfreier und zshgd Graph



Wald



Baum

Satz 1.1 Sei $G = (V, E)$ Graph, $n = |V|$. Dann äquivalent:

a) G ist Baum

b) G ist zshgd und $|E| = n - 1$.

c) G ist kreisfrei und $|E| = n - 1$.

d) G ist kantenmaximal - kreisfrei

jede weitere Kante würde einen Kreis schließen

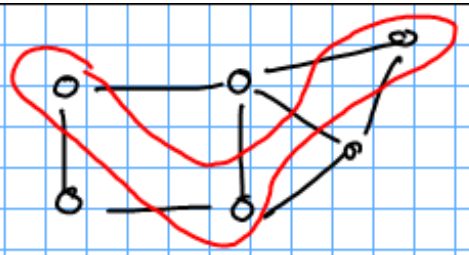
e) G ist kantenminimal - zusammenhängend.

das Löschen einer jeden Kante würde den Zshg zerstören.

f) Je zwei Knoten aus V sind in G durch genau einen Weg verbunden.

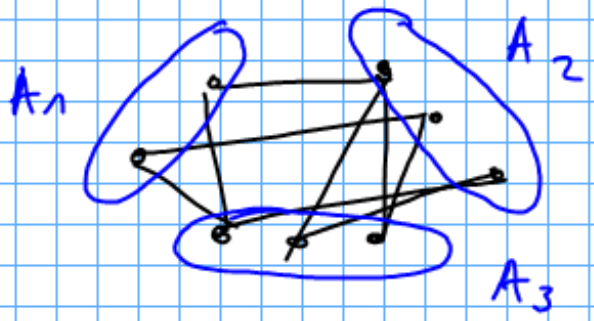
stabile Menge $S \subseteq V$:

$:\Leftrightarrow$ $G[S]$ hat keine Kanten



G ist **k-färbbar**

$:\Leftrightarrow$ $V = \underbrace{A_1 \dot\cup \dots \dot\cup A_k}_{k \text{ stabile Mengen}}$



2-färbbar $\Leftrightarrow$ bipartit

Satz 1.2

G bipartit $\Leftrightarrow$ G enthält keine ungeraden Kreise als Subgraphen

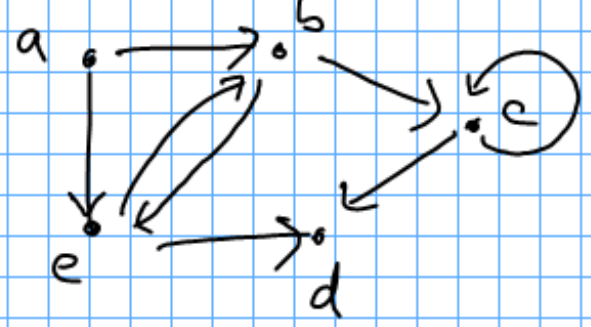
$G = (V, E, \ell)$ (edges) ist gewichteter Graph, wobei

$\ell : E \longrightarrow \mathbb{R}$ Gewichtsfunktion und $\forall H = (W, F) \subset G$

$$\ell(H) := \sum_{e \in F} \ell(e)$$

$G = (V, A)$ (arcs) ist gerichteter Graph, wobei

$A \subset V \times V$ (und nicht $E \subset \binom{V}{2}$).



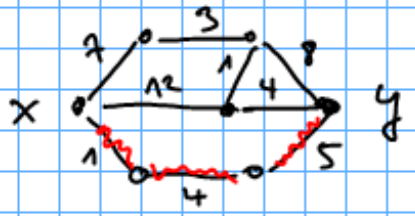
(c,c)

$$A = \{(a,b), (a,e), (b,c), \ldots\}$$

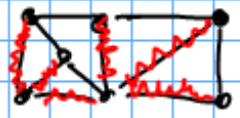
Überblick

Optimierungsprobleme

Kürzeste Wege

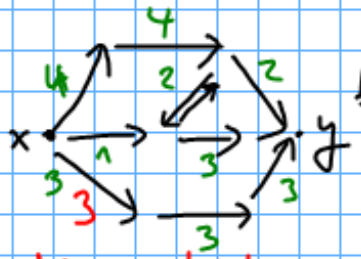


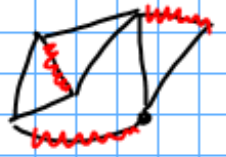
Minimal aufspannende Bäume



Flüsse

Matching





Kapazitätsgrenzen

Algorithmische Prinzipien

dyn. Progr.

Greedy-Algo

↓

Matroide

Augmentierung

$\in P$

$\subset NP$

Reduktion / Approximation

Exkurs: Organisatorisches

1. http://www-m9.ma.tum.de/WS2011/AlgDM

2. Vorlesung

3. Tutorübung:
   a) Übungsaufgaben: online ab Freitag
   b) in die Tü: diskutieren
   c) abgeben bis DIENSTAG, 12:15 im Keller
   d) korrigierte Lsg in Tü zurück
      ↳ Notenbonus → Klausur

50% Punkte

→ Tü-Termine: alle 2 Wochen, siehe Webseite
   Anmeldung: TUM online, ab: heute, 20:00

– Extra-Tü: „andere" Übung.

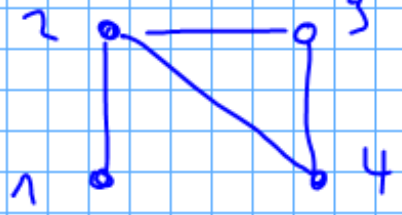
Def 1.4 $G = (V, E)$ mit $V = [n]$, und $|E| = m$

Adjazenzmatrix $A \in \{0,1\}^{n \times n}$ ist gegeben durch

$$A = (a_{i,j}) \quad \text{mit} \quad a_{i,j} := \begin{cases} 1 & : \{i,j\} \in E \\ 0 & : \{i,j\} \notin E \end{cases}$$

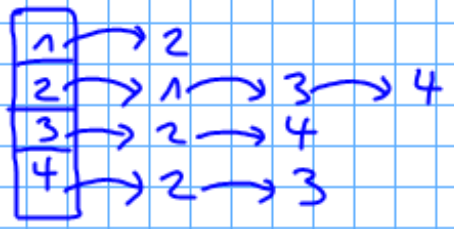
Adjazenzliste L ist gegeben durch einen Vektor, dessen Komponenten (Knoten) Listen der Nachbarn der jeweiligen Knoten enthalten:

Bsp 1.5



$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 1 \\ 3 & 0 & 1 & 0 & 1 \\ 4 & 0 & 1 & 1 & 0 \end{array}$$

$L =$



Vor- / Nachteile:

Adj. matrix:

⊕ Test, ob $\{v,w\} \in E$ in $O(1)$ Schritten. (groß O)

⊖ $\Theta(n^2)$ Speicher (↑ groß Theta)

Adj. liste:

⊖ Test, ob $\{v,w\} \in E$ kostet evtl $\Omega(n)$ Schritte (↑ großes Omega)

⊕ $O(n+m)$ Speicher

Def 1.6 Eine **Warteschlange Q** (Queue) ist eine Folge von Elementen, für die die beiden Operationen

Enqueue (Q, u) ≙ füge Element u am Ende der Folge ein

v := Dequeue (Q) ≙ speichere das erste Element von Q als v ab und entferne es aus Q.

BSP: (1,5,7), (5,7) (dequeue), (5,7,6), (5,7,6,8), (7,6,8) (enqueue)

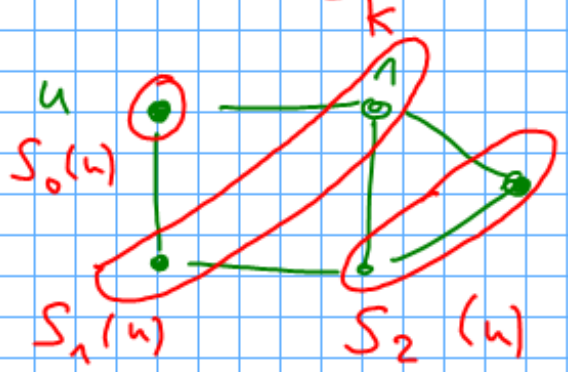
Def 1.7 Sei $G = (V, E)$ ein Graph. Der Abstand zwischen zwei Knoten $u, w \in V$ ist definiert durch

$$\text{dist}(u,w) := \min \{ k \in \mathbb{N}_0 : \exists\ u,w\text{-Weg der Länge } k \}$$

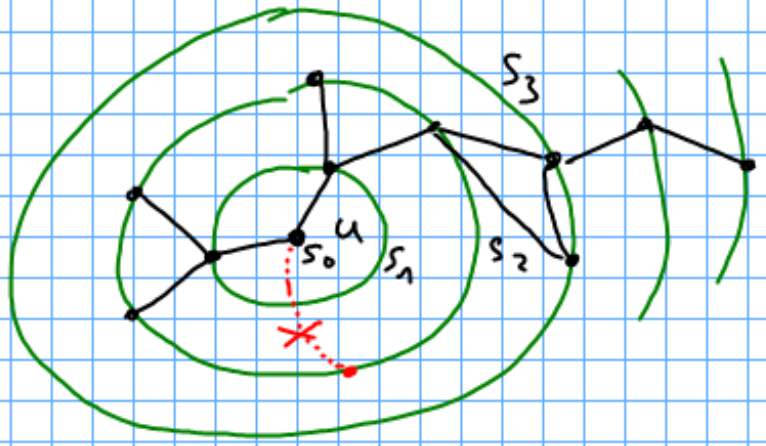
bzw $\text{dist}(u,w) = \infty$, falls kein u,w-Weg in G existiert.

Für $k \in \mathbb{N}_0$ ist die k-te Sphäre von $u \in V$ definiert durch $S_k(u) := \{ w \in V : \text{dist}(u,w) = k \}$.

Bsp:



$\text{dist}(u,1) = 1$



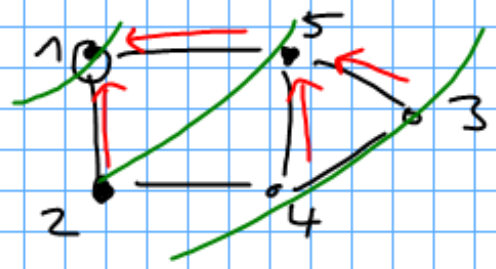
Algo 1.8 Breitensuche (BFS)

Input: Graph $G=(V,E)$, Knoten $u \in V$

Output: Funktionen bekannt: $V \to \{0,1\}$, abst: $V \to \mathbb{N}_0$, vor: $V \setminus \{u\} \to V$

```
BFS (G,u)
1)  Enqueue (Q,u)
2)  bekannt (u) := 1 ,   abst (u) := 0
3)  foreach  w ∈ V \ {u}   bekannt (w) := 0 ,   abst (w) := ∞
4)  while Q ≠ ∅
5)          v := Dequeue (Q)
6)          foreach w ∈ N(v) mit bekannt (w) = 0
7)                  Enqueue (Q,w)
8)                  bekannt (w) := 1
9)                  abst (w) := abst (v) + 1
10)                 vor (w) := v .
```

Bsp 1.9

$G =$ 

BFS(G, 1)

Q =	Welche Knoten w haben bekannt(w) = 1 ?	abst(w) vor(w)
(1)	1	abst(1) = 0
()		
(5,2)	5,2	abst(2) = 1 abst(5) = 1 vor(2) = 1 vor(5) = 1
(2)		
(2,3,4)	3,4	abst(3) = 2 abst(4) = 2 vor(3) = 5 vor(4) = 5
(3,4)		
(4)		
∅		