22.02.2024
Nicolas Wehrli

Algorithmen und Wahrscheinlichkeiten

Heute:
I. Organisation
II. Theory Recap
- Zusammenhang
- Finden von Artikulationsknoten/Brücken: DFS-low
- Eulertour und Hamiltonkreis
- (Satz von Dirac)

III. Aufwärmübung

I. Organisation

- Theorieaufgaben – jede 2. Woche (einzeln)
- Programmieraufgaben – 1 pro Woche auf CE
- Minitests und Peergrading – jede Woche alternierend
  - Minitests behandeln immer nur den Stoff bis und mit Dienstag vorher (ohne Donnerstag Stoff)

    Daten: 29.02., 14.03., 28.03., 02.05. und 23.05.
  - Peergrading besteht aus:
    - Lösen des Peergradingsheets
    - Gegenseitige Beurteilung

    (Relativ einfache Bonuspunkte)

    Daten: 07.03., 21.03., 11.04., 25.04. und 16.05.
- Whatsapp Gruppe? QR-Code nach Übungsstunde

II. Theory Recap

Graphtheorie

Ein ungerichteter Graph ist ein Tupel $G=(V,E)$ wobei
$V=\{v_1, \dots, v_n\}$, $|V|=n$ die Knotenmenge ist, und
$E=\{e_1,\dots,e_m\}\subseteq\{\{u,v\} \mid u,v\in V, u\neq v\}$, $|E|=m$ die Menge der Kanten dazwischen.

keine Multikanten kein self-loop

u und v adjazent $\Leftrightarrow$ $\{u,v\}\in E$

$e\in E$ ist inzident zu v $\Leftrightarrow$ $v\in e$

Nachbarschaft von $v\in V$ in $G=(V,E)$:

$$N_G(v) := \{u\in V \mid \{u,v\}\in E\}$$

$H=(V_H,E_H)$ ist ein Teilgraph von $G=(V_G,E_G)$ gdw.

$$V_H\subseteq V_G \quad \text{und} \quad E_H\subseteq E_G$$

H ist ein Induzierter Teilgraph von G, geschrieben $G[V_H]$ wenn

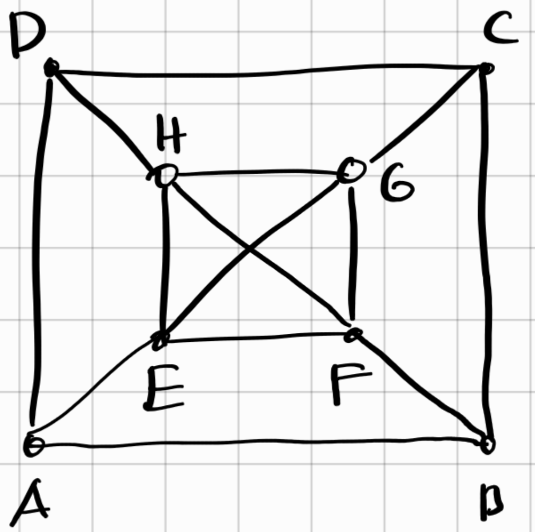
$$V_H\subseteq V_G \quad \text{und} \quad E_H = E_G\cap \binom{V_H}{2}$$

Intuitiv: $\{u,v\} \in E_H \Leftrightarrow \{u,v\} \in E_G \wedge u,v \in V_H$

Wenn zwei Knoten $u,v \in V_H$ schon in G verbunden waren, dann sind sie auch in $G[V_H]$ verbunden.

+ keine neuen Kanten oder Knoten!

Beispielaufgabe:



G

$G[\{E\}]$?

$G[V \setminus \{E\}]$?

$X = \{E, F, G, H\}$

$G[X]$, $G[V \setminus X]$?

Ein gerichteter Graph ist ein Tupel $D = (V, A)$ wobei
$V = \{v_1, \dots, v_n\}$, $|V| = n$ die Knotenmenge ist, und
$E = \{e_1, \dots, e_m\} \subseteq V \times V$, $|E| = m$ die Kantenmenge.

Hier sind Kanten selbst keine Menge sondern Tupel! $e_i = (u,v)$ anstatt $\cancel{\{u,v\}}$

Ungerichtet:

$\deg(v) = \#$ inzidente Kanten

Gerichtet:

$\deg^+(v) = \#$ ausgehende Kanten

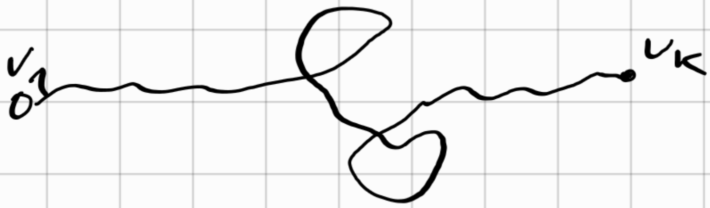
$\deg^-(v) = \#$ eingehende Kanten

Wichtige (ungerichtete) Graphen: (für Gegenbeispiele)

Wir betrachten $\langle v_1, \dots, v_k \rangle$

Weg (engl. Walk):

Eine Folge mit Kanten zwischen v_i und v_{i+1} für alle $i \in \{1, \dots, k-1\}$. Die Länge ist $k-1$.



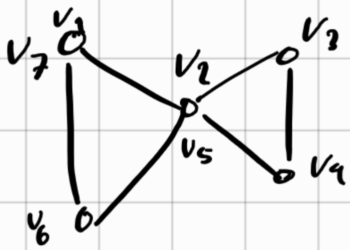
Pfad (engl. Path) P_n

Ein Weg ohne wiederholte Knoten

P_5

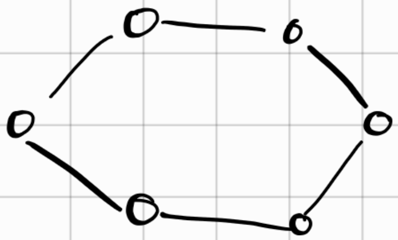
Zyklus (engl. Closed Walk)

Ein Weg mit $v_1 = v_k$

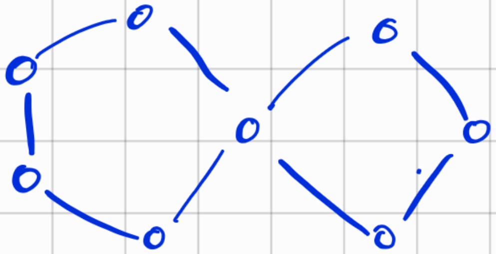


Ein **Kreis (engl. cycle)** C_n ist ein Zyklus, wo alle Knoten ausser Start- und Endknoten verschieden sind.

C_6



$\boxed{\forall v \in V: \deg(v) = 2 \text{ und } G \text{ zsh.}}$



kein Kreis !

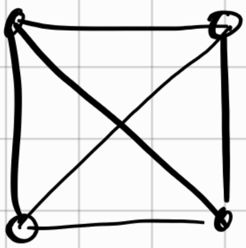
Ein Kreis ist auch ein Zyklus aber ein Zyklus ist nicht unbedingt ein Kreis.

Vollständiger Graph

$K_n = \left(V_n, \binom{V_n}{2}\right)$, $|V_n| = n$

$\binom{V_n}{2} = \{\{u,v\} \mid u,v \in V_n, u \neq v\}$

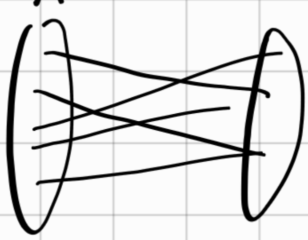
K_4



Bipartiter Graph $G = (A \uplus B, E)$

$E \subseteq \{\{u,v\} \mid u \in A, v \in B\}$ ungerichtet

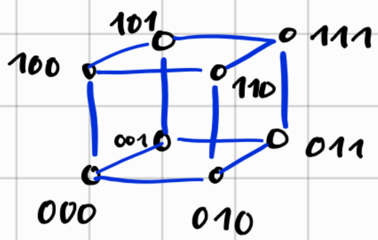
$E \subseteq (A \times B) \cup (B \times A)$ gerichtet



Hyperwürfel Q_d mit Dimension d:

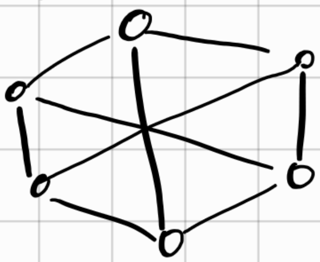
$V = \{0,1\}^d$, Kanten zwischen 2 Knoten, dessen Bitstring sich an einen Position unterscheidet.

Q_3

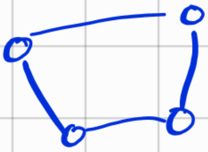


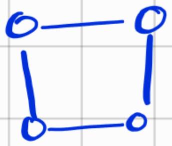
k-regulärer Graph

$\forall v \in V: \deg(v) = k$



3-regulär





2-regulär

2-regulär $\not\Rightarrow$ Kreis

Kreis $\Rightarrow$ 2-regulär

Baum kennt ihr aus AnD

Ein Blatt (leaf) ist ein Knoten mit Grad 1.

1. zsh.
2. kreislos
3. $|E| = |V| - 1$

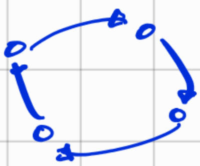
aus beliebigen 2 folgt das 3.

beliebige 2 davon sind äquivalent zu "G ist ein Baum"

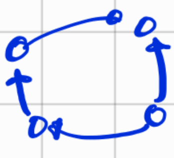
Zusätzlich:

G Baum $\Leftrightarrow$ $\forall x, y \in V$: G enthält genau einen x-y-Pfad.

Sidenote:



gerichteter Kreis

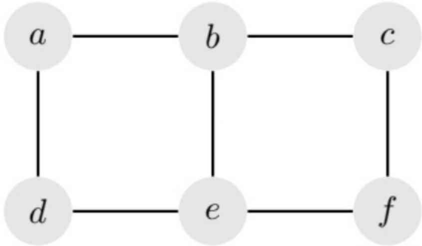


kein gerichteter Kreis

Directed Acyclic Graph
DAG ist ein gerichteter Graph ohne gerichtete Kreise.

Aufgabe 1 – *Pfade, Wege, Kreise*

Betrachten Sie folgenden Graphen $G = (V, E)$.



1. Welche Pfade der Länge 4 (d.h. mit 4 Kanten) gibt es von a nach e?

2. Welche Wege der Länge 4 (d.h. mit 4 Kanten) gibt es von a nach e?

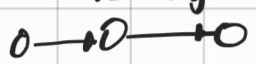
3. Welche Kreise gibt es in G?

4. Wie viele Zykeln gibt es in G?

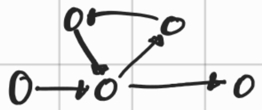
Zusammenhang

> Wir nennen ein Graph $G=(V,E)$ **zusammenhängend**, wenn
> $$\forall s,t \in V: \quad \exists \text{ s-t Pfad}$$

Zur Erinnerung:



Pfad (keine wiederholten Knoten)



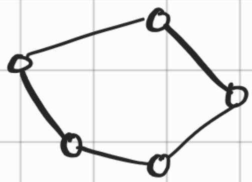
Weg (wiederholte Knoten und Kanten erlaubt)

> Wir nennen einen Teilgraph $C \subseteq G$ eine **Zusammenhangskomponente**, wenn er bezüglich dieser Eigenschaft maximal ist.
> I.e. $\nexists H \subseteq G$: $\underset{C \neq H}{C \subset H}$ und H zusammenhängend.

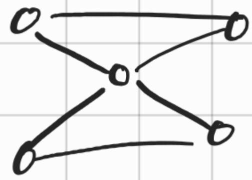
Definition 1.23. Ein Graph $G = (V, E)$ heisst **k-*zusammenhängend***, falls $|V| \geq k+1$ und für alle Teilmengen $X \subseteq V$ mit $|X| < k$ gilt: Der Graph $G[V \setminus X]$ ist zusammenhängend.

Intuitiv: Man muss mindestens k Knoten (und die inzidenten Kanten) löschen, damit der Graph nicht mehr zusammenhängend ist.

Bsp.



ist 2-zsh.

 ist 1-zsh.

Bmk: 1. Für $k \in \mathbb{N}$:

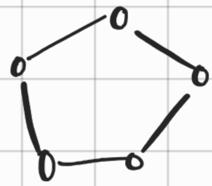
$k+1$ zsh. $\implies$ k zsh.

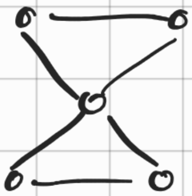
2. $\exists v \in V$: $\deg(v) < k$
$\implies G=(V,E)$ ist nicht k-zsh.

(Man könnte alle Nachbaren von v entfernen)

Definition 1.24. Ein Graph $G = (V, E)$ heisst *k-kanten-zusammenhängend*, falls für alle Teilmengen $X \subseteq E$ mit $|X| < k$ gilt: Der Graph $(V, E \setminus X)$ ist zusammenhängend.

Bsp:

 ist 2-kanten-zsh.

 ist 2-kanten-zsh.

Bmk: 1. Für $k \in \mathbb{N}$:
$k+1$-kanten-zsh. $\implies$ k-kanten zsh.

2. $\exists v \in V$: $\deg(v) < k$
$\implies G=(V,E)$ ist nicht k-kanten-zsh.

(Man könnte alle zu v inzidenten Kanten löschen)

Satz von Menger

Satz von Menger:

Sei $G = (V, E)$ ein Graph. Dann gilt:

G ist **k-zusammenhängend** genau dann wenn (gdw.)

$\forall u, v \in V, u \neq v$ gibt es **k intern-knotendisjunkte** u-v-Pfade

Satz von Menger (Kanten-Version):

Sei $G = (V, E)$ ein Graph. Dann gilt:

G ist **k-kanten-zusammenhängend** genau dann wenn (gdw.)

$\forall u, v \in V, u \neq v$ gibt es **k kantendisjunkte** u-v-Pfade

(Im Skript steht es formaler mit u-v-(Knoten/Kanten)seperatoren
Die Aussage ist dieselbe.
Anschaulich: Seperatoren sind die Mengen X, die den Zusammenhang zerstören.)

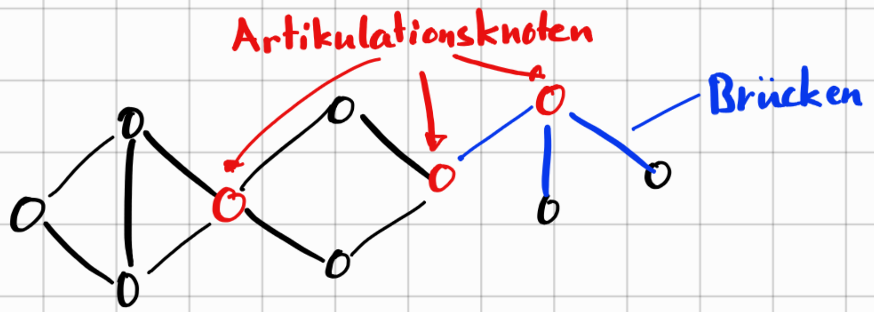
Es gilt immer:

(Knoten-)Zusammenhang ≤ Kanten-Zusammenhang ≤ minimaler Grad

Artikulationsknoten & Brücken

Definition: Sei $G = (V, E)$ ein zusammenhängender Graph.
Ein Knoten $v \in V$ heisst *Artikulationsknoten* (engl. cut vertex)
gdw. $G[V \setminus \{v\}]$ nicht zusammenhängend ist

Definition: Sei $G = (V, E)$ ein zusammenhängender Graph.
Ein Kante $e \in E$ heisst *Brücke* (engl. cut edge)
gdw. $G - e$ nicht zusammenhängend ist



Sei $G=(V,E)$ zsh.:

$\{x,y\}\in E$ Brücke $\Longrightarrow$ $\deg(x)=1$ oder x Artikulationsknoten

Beweisskizze: $\{x,y\}$ Brücke $\Rightarrow G'=(V,E\setminus\{x,y\})$ nicht zsh.

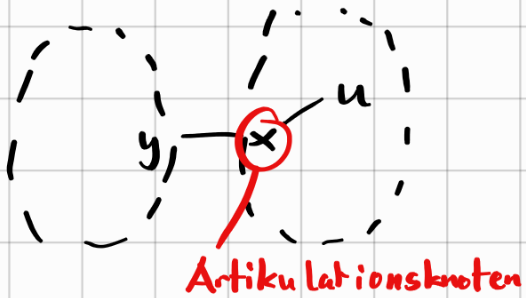
insbesondere $\nexists$ x-y Pfad in G'. (1)

Wenn $\deg(x)\neq 1 \Rightarrow \exists u \neq y$ ($\in V$) s.d. $\{x,u\}\in E' (=E\setminus\{x,y\})$ (2)

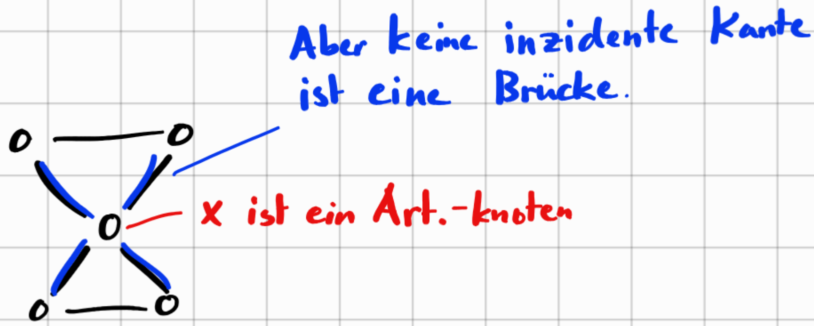
(1), (2) $\Rightarrow \nexists$ y-u Pfad in G'

$\Rightarrow$ Der einzige y-u Pfad in G führt über x

$\Rightarrow$ x ist ein Artikulationsknoten



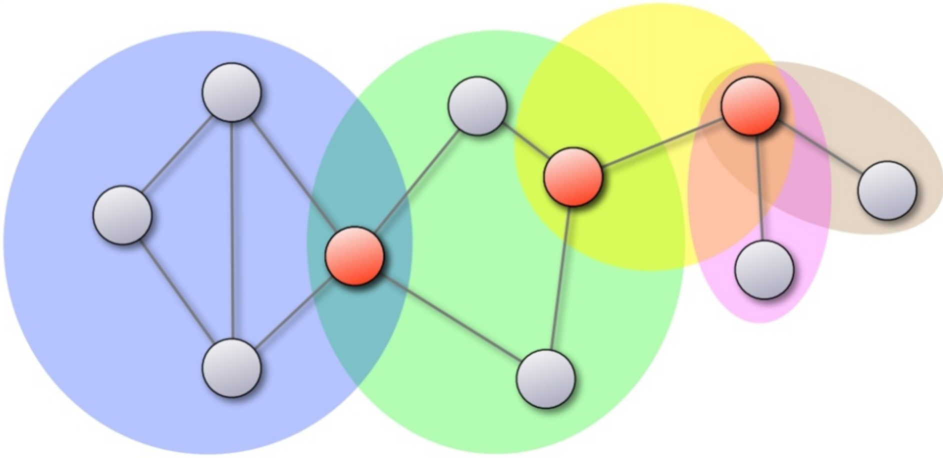
Gegenbeispiel für Umkehrung:



Definition: Sei $G = (V, E)$. Wir definieren eine Äquivalenzrelation auf E durch

$$e \sim f :\Longleftrightarrow \begin{cases} e = f, & \text{oder} \\ \exists \text{ Kreis durch } e \text{ und } f \end{cases}$$

Die Äquivalenzklassen nennen wir Blöcke.



Zur Erinnerung:

Kreis (Engl. cycle) " $\Longleftrightarrow$ " keine wiederholten Knoten. (ausser dass $v_0 = v_n$)

Zyklus (Engl. closed walk) " $\Longleftrightarrow$ " Knoten können mehrmals vorkommen

Aus dem DM Skript:

Definition 3.19. An *equivalence relation* is a relation on a set A that is reflexive, symmetric, and transitive.

Wir können einfach überprüfen, dass die Relation '$\sim$' reflexiv, transitiv und symmetrisch ist.

Finden von Artikulationsknoten und Brücken:

DFS-Low:

low[v] := kleinste dfs-Nummer, die man von v aus durch einen gerichteten Pfad aus (beliebig vielen) Baumkanten und maximal einer Restkante erreichen kann.

DFS-tree

Global variables low[], dfs[], T, int num

DFS-Visit(G,v)

num ← num+1 // step counter
dfs[v] ← num
low[v] ← dfs[v]
for all $\{v,w\} \in E$ do // for every neighbor of v
  if dfs[w] = 0 do // if unvisited ({v,w} Baumkante)
    T ← T + {v,w} // add Edge to T and call recursively (returns low[w])
    val ← DFS-Visit(G, w)
    low[v] ← min (low[v], val)
  else if $\{v,w\} \notin T$ // visited ({v,w} Restkante)
    low[v] ← min(low[v], dfs[w])
return low[v]

if not for the check we could reuse the 'Baumkante'

DFS-Low (G, s)

$\forall v \in V$: low[v] ← 0, dfs[v] ← 0, isArtVert[v] ← false
num ← 0
T ← $\emptyset$
DFS-Visit(G,s)
for all $v \in V$ do
  if low[v] ≥ dfs[v] then isArtVert[v] ← true
if deg(s) ≥ 2 do isArtVert[s] ← true
else isArtVert[s] ← false
return isArtVert

p. 39

Notice that the code in the script calculates isArtVert 'en passant'

Wir haben hierbei folgendes Lemma verwendet:

v ist Artikulationsknoten

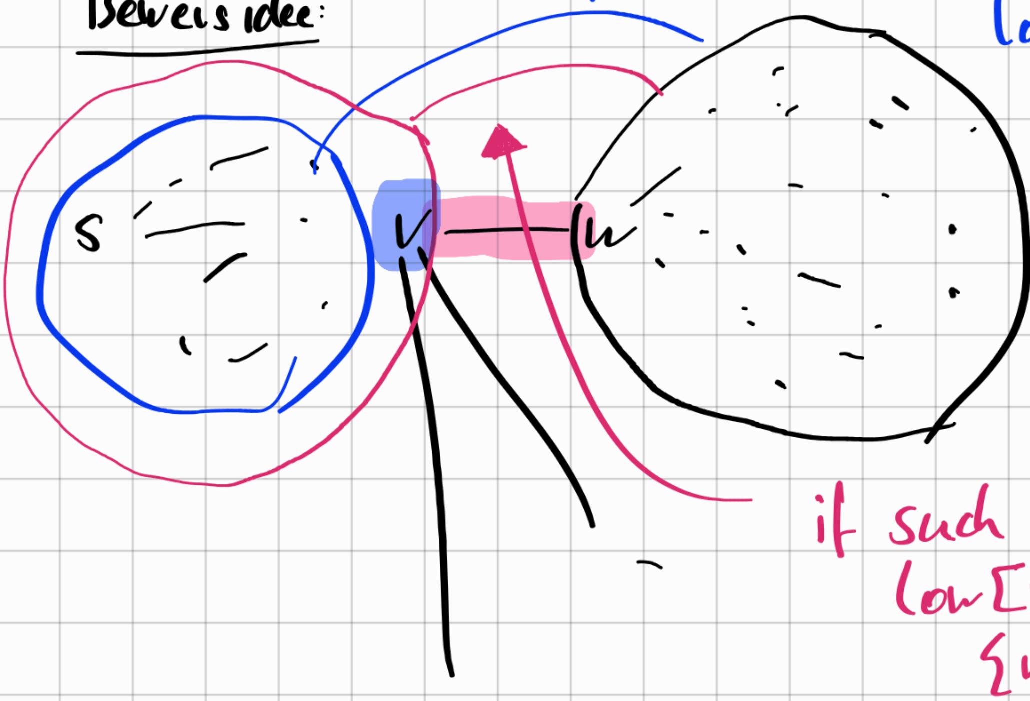
$\iff$ $v = s$ und s hat in T Grad mindestens zwei, oder

$v \neq s$ und es gibt $w \in V$ mit $\{v, w\} \in E(T)$ und $\mathtt{low}[w] \geq \mathtt{dfs}[v]$.

Um Brücken zu finden, müssten wir einfach bei der Extraction, folgendes Lemma verwenden:

Eine Baumkante e = (v,w) (v Elternknoten, w Kindknoten) ist genau dann eine Brücke, wenn $\mathrm{low}[w] > \mathrm{dfs}[v]$.

Beweisidee:



if such an edge exists, $low[v] < dfs[v]$ und v ist kein AV.
else
$low[v] \geq dfs[v]$ und v ist ein AV.

if such an edge exists
$low[w] \leq dfs[v]$ und $\{v,w\}$ ist keine Brücke
else
$low[w] > dfs[v]$ und $\{v,w\}$ ist eine Brücke.

Eulertour & Hamiltonkreis

Eine **Eulertour** ist ein geschlossener Weg (Zyklus) in $G=(V,E)$, der jede Kante genau einmal enthält.

Bmk: Eulertour $\hat{=}$ Eulerzyklus

Ein **Hamiltonkreis** ist ein Kreis in $G=(V,E)$, der jeden Knoten genau einmal enthält.

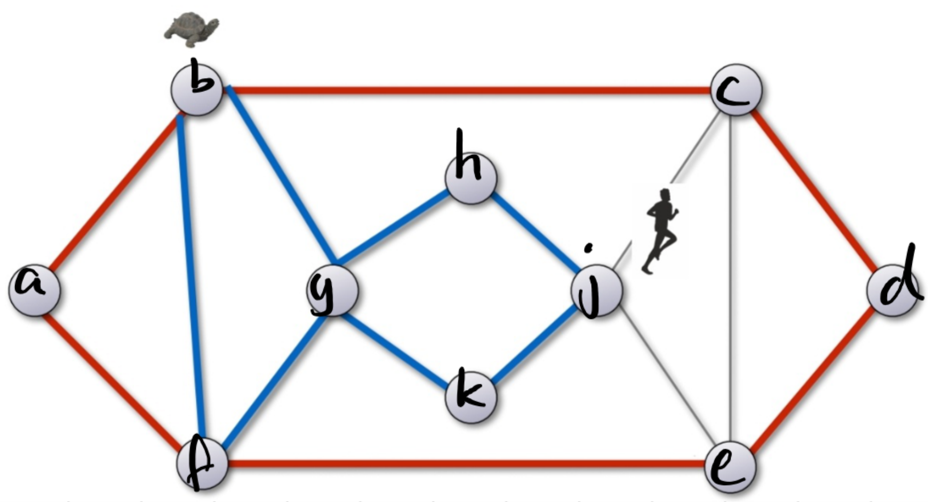
Eulertour finden (angenommen, dass eine exst.):

EULERTOUR(G, v_{start})

// *Schneller Läufer*
$W \leftarrow$ RANDOMTOUR$_G(v_{\text{start}})$
// *Langsamer Läufer*
$v^{\text{langsam}} \leftarrow$ Startknoten von W.
while v^{langsam} ist nicht letzter Knoten in W **do**
    $v \leftarrow$ Nachfolger von v^{langsam} in W
    if $N_G(v) \neq \emptyset$ **then**
        $W' \leftarrow$ RANDOMTOUR$_G(v)$
        // *Ergänze* $W = W_1 + \langle v \rangle + W_2$ *um die*
        // *Tour* $W' = \langle v'_1 = v, v'_2, \ldots, v'_{\ell-1}, v'_\ell = v \rangle$
        $W \leftarrow W_1 + W' + W_2$
    $v^{\text{langsam}} \leftarrow$ Nachfolger von v^{langsam} in W
return W

RANDOMTOUR$_G(v_{\text{start}})$

$v \leftarrow v_{\text{start}}$
$W \leftarrow \langle v \rangle$
while $N_G(v) \neq \emptyset$ **do**
    Wähle v_{next} aus $N_G(v)$ beliebig.
    Hänge v_{next} an die Tour W an.
    $e \leftarrow \{v, v_{\text{next}}\}$
    Lösche e aus G.
    $v \leftarrow v_{\text{next}}$
return W



Zuerst
W=[a,b,c,d,e,f]

Mit dem langs. Läufer fügen wir
[b,g,h,j,k,g,f,b]
an der Stelle von b in W
ein.

Wie schon in AnD gesehen:

Satz: Ein zusammenhängender Graph G = (V, E) enthält eine Eulertour

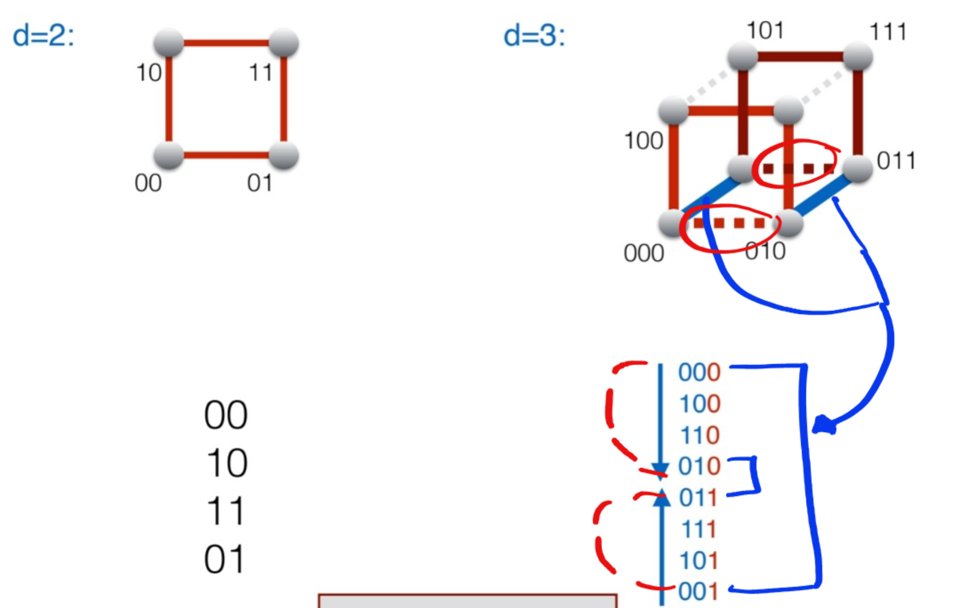
gdw. der Grad jedes Knotens gerade ist.

… und eine solche kann man in O(|E|) Zeit finden

Für Hamiltonkreise, gibt es keine solch einfache Vorgehensweise.

Satz: Seien m,n ≥ 2.
Ein n×m Gitter enthält einen Hamiltonkreis gdw n*m gerade ist.

Satz: Ein bipartiter Graph $G = (A \cup B, E)$ mit $|A| \neq |B|$ kann keinen Hamiltonkreis enthalten.



00
10
11
01

analog für d ≥ 4

Satz von Dirac

Jeder Graph G = (V, E) mit $|V| \geq 3$ und Minimalgrad $\delta(G) \geq |V|/2$ enthält einen Hamiltonkreis.

III. Aufwärmübung

Aufgabe 4 – *Eine generelle Eigenschaft von Graphen*

Zeigen Sie, dass jeder Graph G mit $n \geq 2$ Knoten zwei Knoten $v \neq w$ enthält, sodass $deg(v) = deg(w)$.

Hinweis: Für ein gegebenes n, was ist der grösstmögliche Grad den ein Knoten haben kann?

Aufgabe 5 – *Algorithmus*

Beschreiben Sie einen Algorithmus der das folgende Problem löst: Gegeben ist die Eingabe bestehend aus einen Graphen $G = (V, E)$ mit n Knoten (gehen Sie davon aus, dass der Graph als Adjazenzliste gegeben ist). Ihr Algorithmus soll "Ja" ausgeben, falls G ein Baum ist und "Nein" andernfalls.

Wie immer wenn Sie einen Algorithmus beschreiben gehöhrt zu einer vollständigen Lösung: eine klare Beschreibung des Algorithmus, ein Korrektheitsbeweis und eine Laufzeitanalyse.

Hinweis: Für diese Aufgabe dürfen Sie das Statement aus Aufgabe 6 ohne Beweis verwenden.