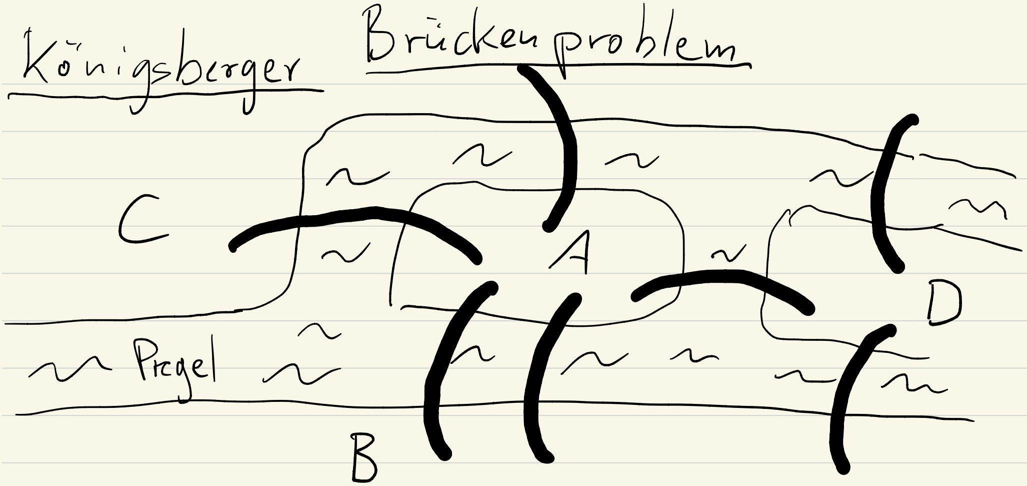
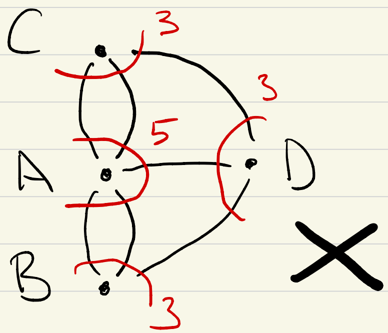
Königsberger Brückenproblem



Euler, bedeutender Mathematiker, *1707

jede Brücke genau einmal überqueren?

Abstraktion: "Graph"



Knoten: A, B, C, D

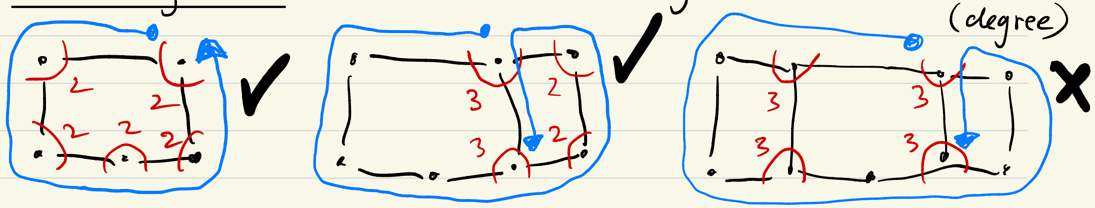
$\simeq$ Stadtteile

Kanten: Verbindungen zw. Knoten

$\simeq$ Brücken

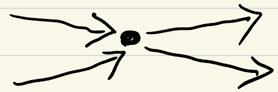
Eulerweg: durchlaufe jede Kante genau einmal (Eulerian walk)

Knotengrad: Anzahl anliegender Kanten (degree)



Beobachtung: Wenn Eulerweg existiert, dann ≤ 2 Knotengrade ungerade

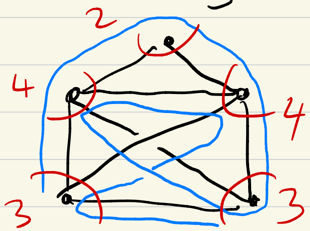
Beweis: ausser für Start- und Endknoten,



#hin = #weg

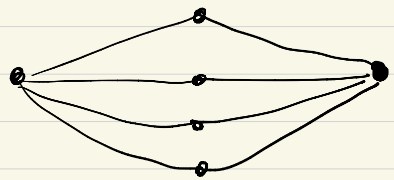
Knotengrad = #hin + #weg = 2 #hin

Haus vom Nikolaus: zeichnen ohne Absetzen



Beobachtung: möglich $\Leftrightarrow$ $\exists$ Eulerweg

Hamiltonpfad: besuche jeden Knoten genau einmal



Eulerweg: ✓

Hamiltonpfad: ✗

Algorithmen für Eulerweg/Hamiltonpfad

brute-force: alle möglichen Kanten-/Knoten-reihenfolgen durchprobiert $\rightsquigarrow$ Laufzeit $\Omega(n!)$ (n Knoten)

besser?

Eulerweg: ja! $O(n+m)$ (m Kanten)

Hamiltonpfad: wohl nicht!

Vermutung: polynomielle Laufzeit unmöglich!

($P \neq NP$ Vermutung)

Graphen: zentral für Informatik

mathematisches Modell für Netzwerke

- Computer-N. — internet
- Soziale-N. — "six degree of separation" Psychologe Milgram 1967
- Transport. N — schnellster Weg von A nach B
- Neuronale. N. — (künstliche) Intelligenz

Definition: Graph $G = (V, E)$

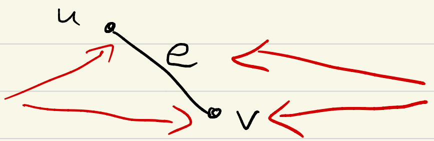
- Knotenmenge V (vertices)
- Kantenmenge E (edges)

jede Kante ist ungeordnetes Paar von Knoten

$u \overset{e}{\bullet\!\!-\!\!\!-\!\!\bullet} v$ entspricht $e = \{u, v\} \in E$

Begriffe:

adjazent/ benachbart



inzident/ anliegend

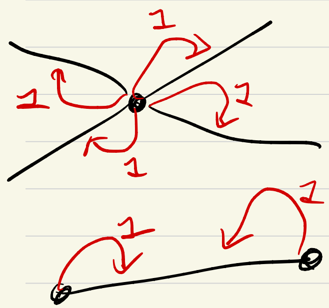
$\deg(u)$ = Knotengrad von u

Kurzform: $e = uv$ statt $e = \{u, v\}$

Handschlaglemma: $\displaystyle\sum_{v \in V} \deg(v) = 2|E|$

Beweis:

jeder Knoten verteilt 1 Franken an jede inzidente Kante



$\rightsquigarrow$ jede Kante erhält insgesamt genau 2 Franken

Weg: Folge von benachbarten Knoten (walk)

$$\underbrace{\overset{\text{Start-knoten}}{v_0} - v_1 - v_2 - \dots - v_{\ell-1} - \overset{\text{End knoten}}{v_\ell}}_{\text{Länge } \ell}$$

Pfad: Weg ohne wiederholte Knoten

Zyklus: Weg mit $v_0 = v_\ell$, $\ell \geq 2$ (closed walk)

Beobachtung: Weg ist Zyklus $\Leftrightarrow$ Endknoten inzident zu geraden Anzahl Kanten im Weg

Beweis: Besuche des Endknoten im inneren Teil des Wegs ändern Parität nicht

$\rightsquigarrow$ Parität gerade

$\Leftrightarrow$ Endknoten zweimal am Rand des Weges

$\Leftrightarrow$ Startknoten = Endknoten

u erreicht v: $\exists$ Weg zwischen u und v (reachable)

Äquivalenzrelation (symm., reflexiv, transitiv)

Zusammenhangskomponente: Äquivalenzklasse dieser Relation

(ZHK; connected component)

Graph zusammenhängend: genau eine ZHK

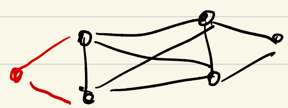
jeder Knoten erreicht jeden anderen

Eulerzyklus: Zyklus mit allen Kanten genau einmal

Behauptung: $\exists$ Eulerzyklus

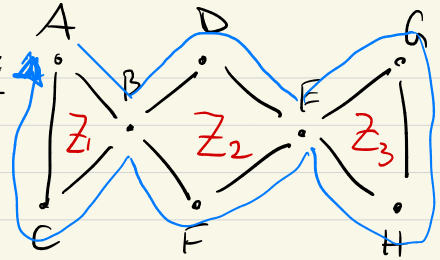
$\Leftrightarrow$ zusammenhängend & alle Knotengrade gerade

Beweis: "$\Rightarrow$" wie für Eulerweg



"$\Leftarrow$" per Algorithmus (Rest der Vorlesung)

Beispiel:



Z_1: A ~~B~~ C A

Z_2: B D ~~E~~ F B

Z_3: E G H E

Ansatz:

- berechne iterativ Zyklen; ohne wiederholte Kanten; bis alle Kanten aufgebraucht
- verschmelze Zyklen

Zyklen existieren? Wie Z. finden?

Walk(u): (finde langen Weg von u ohne wiederholte Kanten)

if $\exists v.\ uv \in E$, nicht markiert

  markiere uv

  Walk(v)

Eigenschaften:

1. Walk(u) markiert Weg W mit Startknoten u
2. Keine Kante wiederholt
3. Endknoten von W hat alle Kanten markiert

Beweis von 1: folgt von 2

Im Beispiel: alle Kanten unmarkiert zu anfangs

Walk(B): B A C B D E F B

Walk(E): E G H E

Invariante: $\forall v \in V$. Anzahl unmarkierter Kanten an v gerade

Behauptung:

1. I. wird von Walk(u) aufrecht erhalten
2. falls I. vor Walk(u) gilt, dann ist W ein Zyklus

Beweis von 2:

zu zeigen: Endknoten v von W hat gerade Anzahl Kanten in W

	vor Walk(u)	nach Walk(u)
unmarkierte Kanten an v	gerade (I.)	0 (Eigenschaft 3)

Walk(u) markiert gerade Anzahl Kanten an v

⇝ W hat gerade Anzahl Kanten an v

Laufzeit:

anstatt Zyklen iterativ zu suchen verwende Rekursion um alle Kanten in Zyklen einzuteilen

Euler(G):

(finde Eulerzyklus wenn existent)

- Leere Liste Z, alle Kanten unmarkiert
- EulerWalk(u_0) für $u_0 \in V$ beliebig
- return Z

Euler Walk(u):

⚠ auch nicht in Rekursion markiert

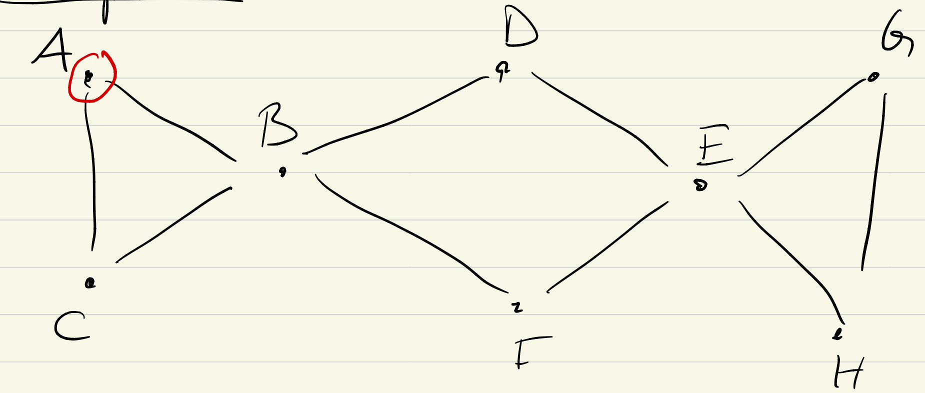
for $uv \in E$, nicht markiert

 markiere uv

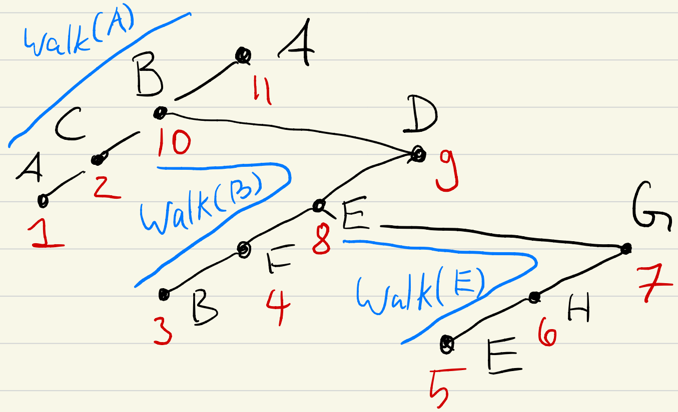
 Euler Walk(v)

$Z \leftarrow (Z, u)$

Beispiel:



Rekursionsbaum



Z: A C B F E H G E D B A

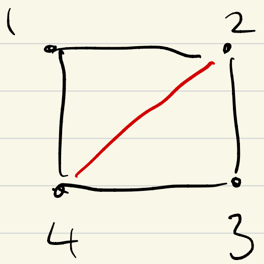
Laufzeit? Welche Datenstruktur für Graphen?

$V = \{1, \dots, n\}$, $G = (V, E)$, $m = |E|$

Adjazenzmatrix

$A = (A_{uv})_{u,v \in V}$

$$A_{uv} = \begin{cases} 1 & \text{falls } uv \in E \\ 0 & \text{sonst} \end{cases}$$

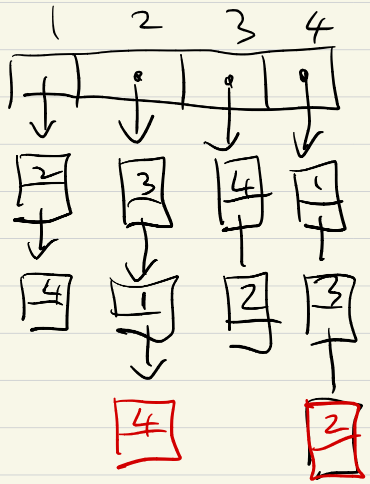


$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & & & & \\ 2 & & & & \\ 3 & & & & \\ 4 & & & & \end{array} \quad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Adjazenzliste: Array von Listen

$Adj[u]$ = Liste der Nachbarn von u

(beliebige Reihenfolge



Laufzeit von Operationen auf A

- teste ob $uv \overset{?}{\in} E$: $O(1)$ —— $O(1 + \min\{deg(u), deg(v)\})$
- durchlaufe inzidente Kanten: $O(n)$ —— $O(1 + deg(u))$

Laufzeit von EulerWalk:

pro rekursivem Aufruf: $O(n)$

$\leq m$ rekursive Aufrufe

total: $O(n \cdot m)$

besser? ja!

mit dieser Datenstruktur: Laufzeit $O(n+m)$ für EulerWalk möglich