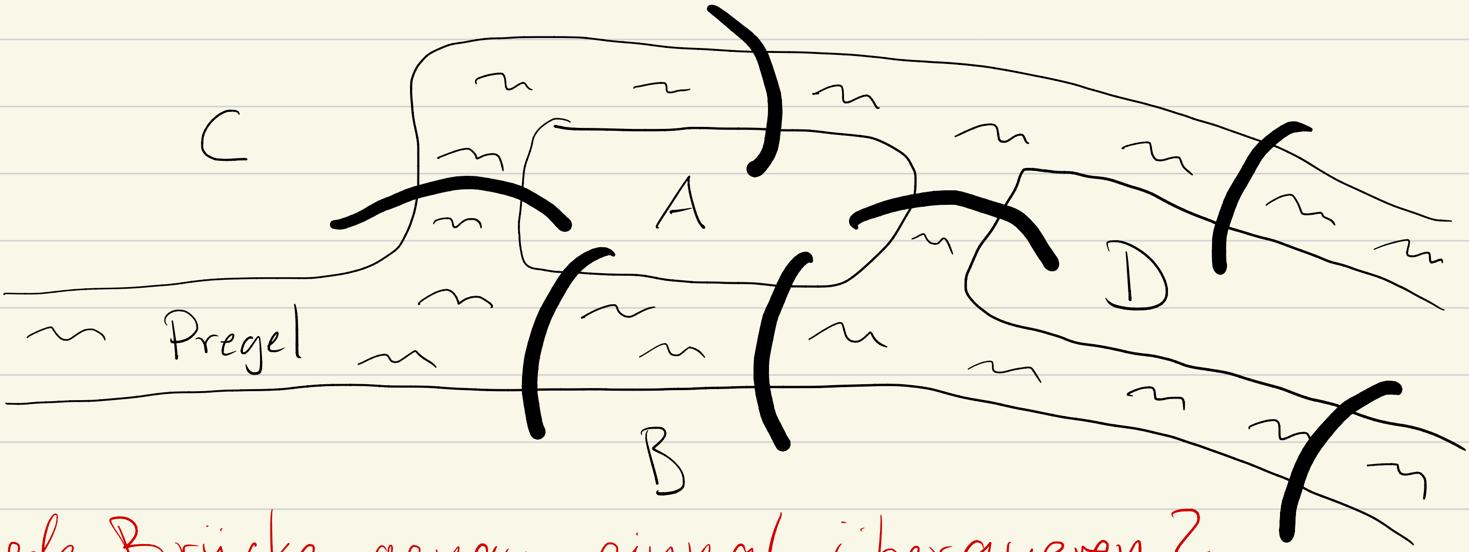
heute: Einführung in wichtiges Konzept der Informatik
vorab: anschauliches Beispiel und historischer Ursprung

Königsberger Brückenproblem

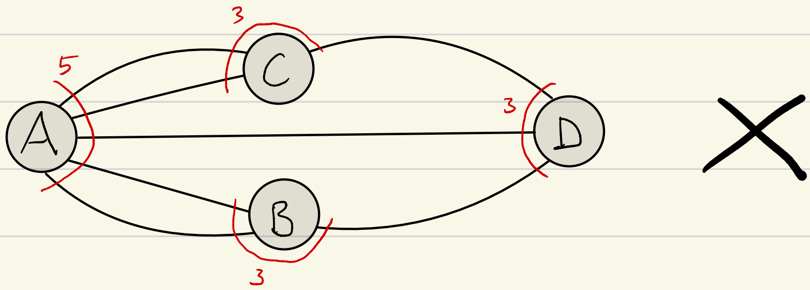
Leonhard Euler, bedeutender Mathematiker, * 1707, Basel



jede Brücke genau einmal überqueren?

Abstraktion: "Graph"

genaue Lage der Brücken nicht relevant

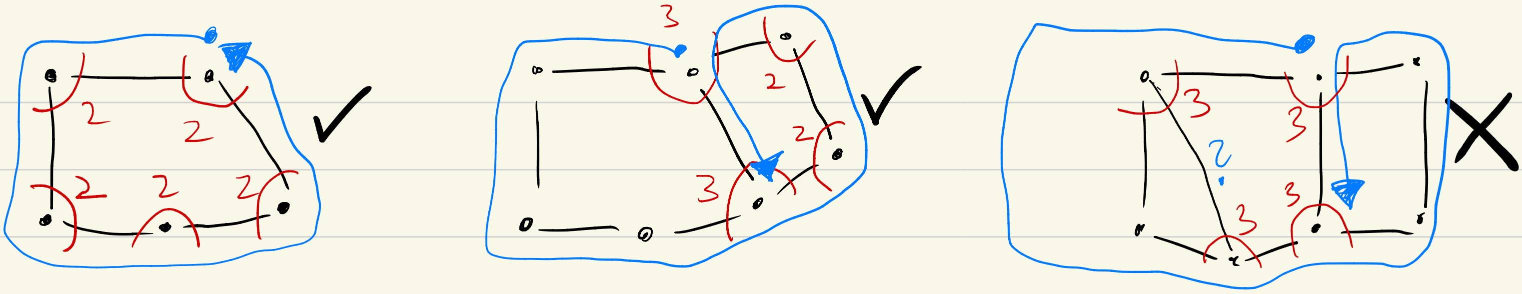


Knoten: A, B, C, D $\simeq$ Stadtteile

Kanten: Knotenverbindungen $\simeq$ Brücken

Eulerweg: (Eulerian walk)

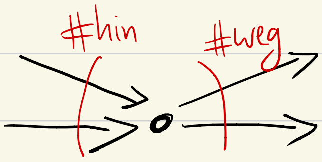
durchlaufe jede Kante genau einmal



<u>Knotengrad</u>: Anzahl anliegender Kanten (degree)

<u>Beobachtung</u>: Wenn Eulerweg existiert,
dann ≤ 2 Knotengrade ungerade

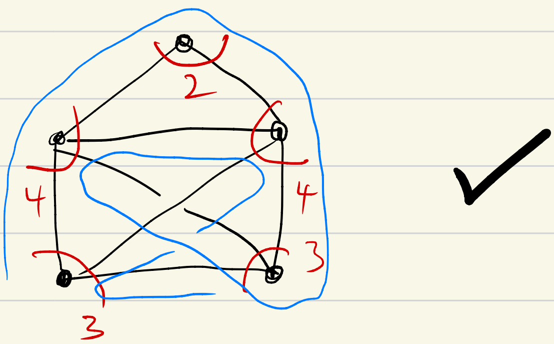
<u>Beweis</u>: ausser für Start- und Endknoten gilt



$\#\text{hin} = \#\text{weg}$

$$\text{Knotengrad} \overbrace{=}^{\text{da Eulerweg}} \#\text{hin} + \#\text{weg} = \underbrace{2 \cdot \#\text{hin}}_{\text{gerade}}$$

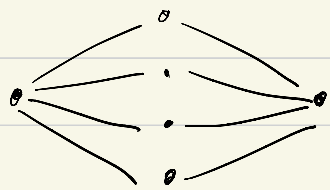
<u>Haus vom Nikolaus</u>



zeichnen ohne Absetzen

<u>Beobachtung</u>: möglich $\Leftrightarrow$ $\exists$ Eulerweg

Hamiltonpfad: besuche jeden **Knoten** genau einmal



Eulerweg: ✓

Hamiltonpfad: ✗

jeder mittlere Knoten braucht ≥ 1 Kante;
bis auf 2 Knoten sogar ≥ 2 Kanten
$\rightsquigarrow$ ein äusserer Knoten hat Grad > 2

Algorithmen für Eulerweg / Hamiltonpfad

n Knoten, m Kanten

brute-force: alle möglichen Kanten-/Knotenreihenfolgen durchprobieren $\rightsquigarrow$ **Laufzeit:** $\Omega(n!)$

geht es besser?

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

Hamiltonpfad: polynomielle Laufzeit unmöglich unter $P \neq NP$ Vermutung

(mehr dazu in späterem Semester)

Graphen: zentral für Informatik

mathematisches Modell für Netzwerke, z. B.

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

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

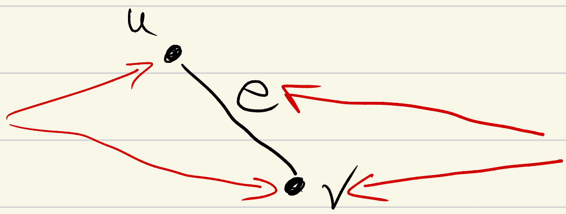
- Knotenmenge V (vertices)
- Kantenmenge E (edges)

jede Kante ist ungeordnetes Paar zweier Knoten $u \neq v$

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

Begriffe:

u, v adjazent / benachbart



e inzident / anliegend zu v

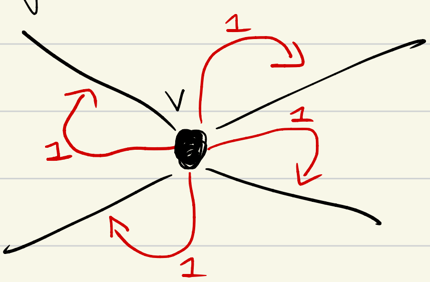
$\deg(u)$ = Knotengrad von u (Anzahl Nachbarn)

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

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

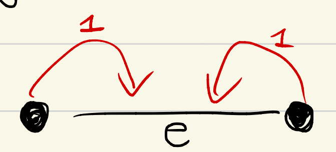
Beweis:

jeder Knoten v



$\simeq$

jede Kante e



jeder Knoten verteilt 1 Franken an jede inzidente Kante

$\rightsquigarrow$ jede Kante erhält insgesamt 2 Franken

Pause

Weg: Folge von benachbarten Knoten (walk)

Start-Knoten $\underbrace{v_0 - v_1 - v_2 - \cdots - v_{\ell-1} - v_\ell}_{\text{Länge } \ell}$ Endknoten

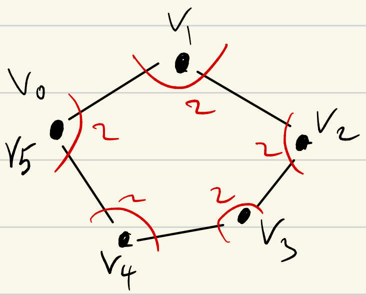
Knoten und Kanten im Weg dürfen sich wiederholen

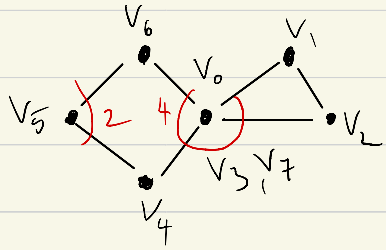
Pfad: Weg ohne wiederholte Knoten

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

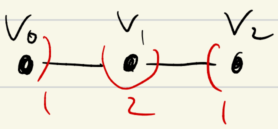
Beispiele

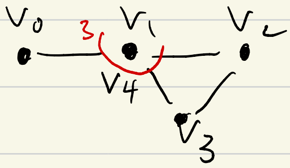
Zyklen:





nicht-Zyklen:





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

(Beweis in Aufschrieb)

u erreicht v: ∃ Weg zwischen u und v (reachable)

Äquivalenzrelation (symmetrisch, 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

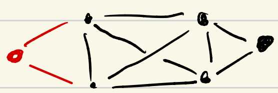
Behauptung: ∃ Eulerzyklus

$\Longleftrightarrow$ alle Knotengrade gerade und alle Kanten in einer ZHK

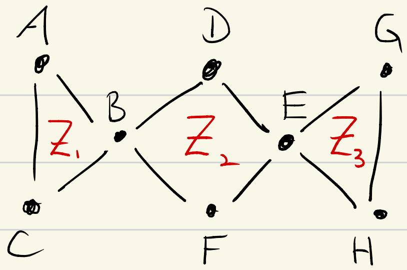
isolierte Knoten können wir ignorieren

Beweis: "$\Rightarrow$" wie vorher für Eulerweg

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



Beispiel:



Z_1 : $A\,\cancel{B}\,C\,A$

$+\ Z_2$: $B\,D\,\cancel{E}\,F\,B$

$+\ Z_3$: $E\,G\,H\,E$

Ansatz:

- berechne iterative Zyklen, ohne wiederholte Kanten, bis alle Kanten aufgebraucht
- verschmelze Zyklen

Zyklen existieren? Wie finden?

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

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

markiere uv

Walk(v)

Im Beispiel: keine Kanten markiert zu anfangs

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

Walk(E): E G H E

am Ende sind alle Kanten markiert

Eigenschaften

besuchen immer benachbarte Knoten

1. Walk(u) markiert Weg W mit Startknoten u
2. jede Kante höchstens einmal markiert markierte Kante bleibt markiert
3. Endknoten von W hat alle Kanten markiert sonst wäre Walk(u) weiter gelaufen

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

Behauptung:

1. Inv wird von Walk(u) aufrecht erhalten
2. falls Inv vor Walk(u) gilt, dann ist W 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 (Inv)	0 (Eigenschaft 3)

Walk(u) markiert gerade Anzahl Kanten an v

$\rightsquigarrow$ W hat gerade Anzahl Kanten an v $\rightsquigarrow$ Zyklus

Beweis von 1: folgt von 2 (in Zyklus hat jeder Knoten gerade Anzahl Kanten)

Pause

Laufzeit

bisher: polynomielle Schranke klar

wie nächsten Startknoten für Walk(·) wählen?

Idee: vorherige Berechnungen wiederverwenden mittels Rekursion

Euler(G): (finde Eulerzyklus in G wenn existent)

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

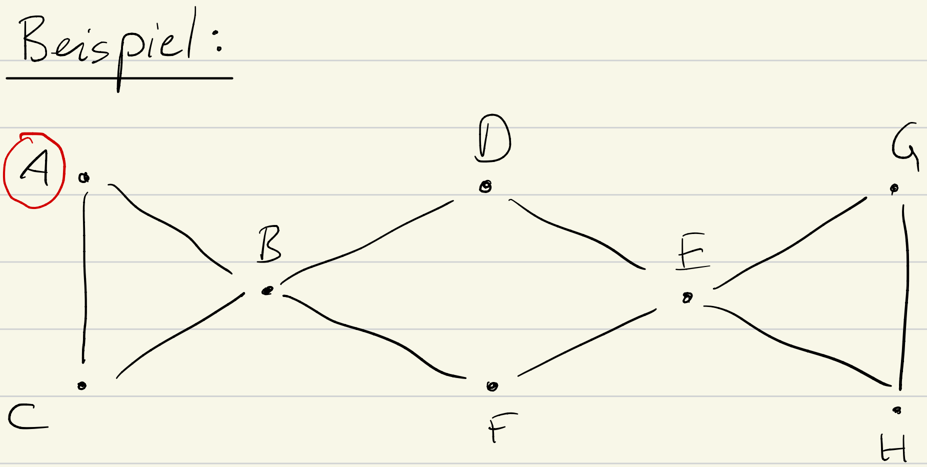
Euler Walk(u): ⚠ auch nicht in Rekursion markiert!

for $uv \in E$, nicht markiert

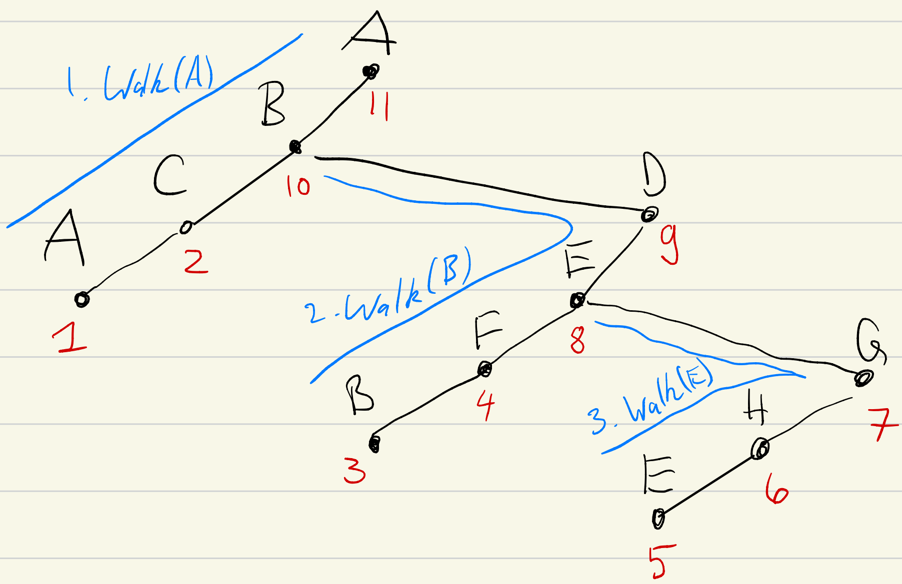
    markiere Kante uv
    EulerWalk(v)

$Z \leftarrow (Z, u)$

Beispiel:



Rekursionsbaum



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

Korrektheit (Skizze):

rekursiver Algorithmus $\simeq$ iteriere walk(·) geeignet (siehe Beispiel)

wähle nächsten Startknoten für walk(·):

gehe rückwärts im zuletzt gefundenen Zyklus bis Knoten mit unmarkierter Kante gefunden

Laufzeit: $O(n+m)$ mit geeigneter Datenstruktur für Graph