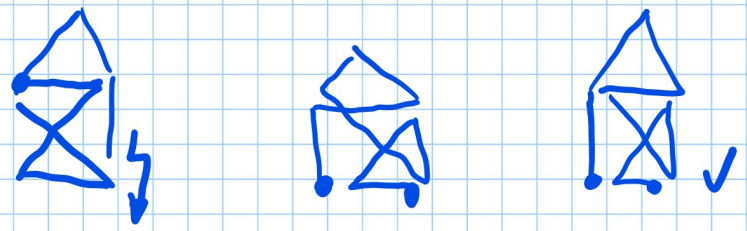
Haus vom Nikolaus

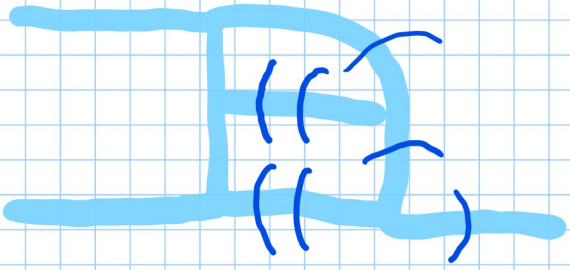


Wann / warum funktioniert das?

Leonhard Euler (1707 – 1783) Basel St. Petersburg

Begründer der Graphentheorie

Königsberger Brückenproblem



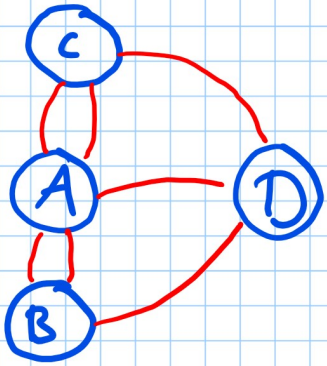
∃ Rundweg, welcher jede Brücke genau 1x benutzt?

Euler (1736): nein

Abstraktion: Graph

Menge Knoten $V = \{v_1, \dots, v_n\}$

Menge Kanten E



gerichteter Graph

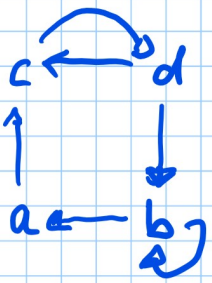
$E \subseteq V \times V = \{ (u, v) : u \in V, v \in V \}$

ungerichteter Graph

$E \subseteq \{ \{u, v\} : u \in V, v \in V \}$

Beispiele

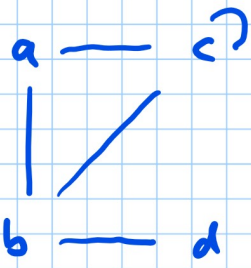
gerichtet



$V = \{a, b, c, d\}$

$E = \{(a,c), (b,a), (b,b), (c,d), (d,c)\}$

ungerichtet



$E = \{\{a,b\}, \{a,c\}, \{b,c\}, \{b,d\}, \{c,e\}\}$

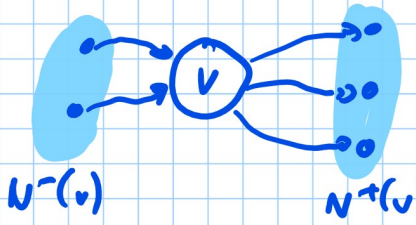
gerichtete Graphen:

Vorgänger $N^-(v) := \{u \in V \mid (u,v) \in E\}$

Nachfolger $N^+(v) := \{u \in V \mid (v,u) \in E\}$

Eingangsgrad $\deg^-(v) := |N^-(v)|$

Ausgangsgrad $\deg^+(v) := |N^+(v)|$



$\deg^-(v) = 2$

$\deg^+(v) = 3$

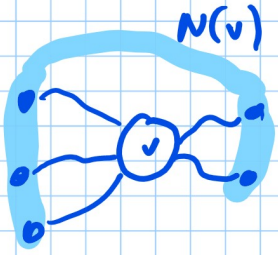
$\deg^-(v) = 1$

$\deg^+(v) = 1$

Ungerichtete Graphen

Nachbarschaft $N(v) = \{ u \in V \mid \{u, v\} \in E \}$

Grad $\deg(v) = |N(v)|$, Schleifen zählen 2



$\deg(v) = 5$

$\deg(v) = 2$

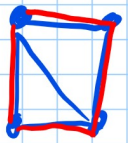
Vollständiger Graph: ungerichtet mit $E = \{ \{u, v\}, u \in V, v \in V, u \neq v \}$

Weg: Sequenz $p = \langle v_1, \dots, v_k \rangle$ von Knoten aus V

so dass $\exists$ Kante $v_i \to v_{i+1}$ $\forall i = 1 \dots k-1$

Pfad (einfacher Pfad): Weg $p = \langle v_1, \dots, v_k \rangle$ mit

$v_i \neq v_j$ $\forall 1 \leq i < j \leq k$



Zyklus: Weg $v_1 \overset{+}{\rightsquigarrow} v_k$ mit $v_1 = v_k$

Einfacher Zyklus: Zyklus mit $v_i \neq v_j$ $\forall 1 < i < j < k$

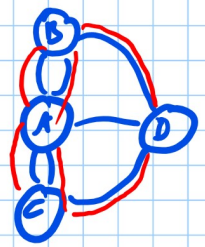
Zusammenhang

ungerichteter Graph heißt zusammenhängend

gerichteter Graph heißt stark zusammenhängend

wenn $\forall u \in V, v \in V, u \neq v$ $\exists$ Weg $u \overset{*}{\rightsquigarrow} v$

Zurück zu Euler



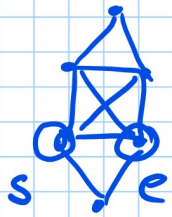
∃ kein Zyklus (ungerichteter, zusammenhängender Graph)

Eulerscher Kreis: Zyklus, welcher jede Kante eines zshgd. Graphen genau 1x enthält.

Euler 1736: ∃ Eulerscher Kreis ⇔ jeder Knoten hat geraden Grad.

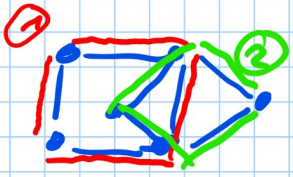
?

⇒ Erweiterung



"⇒": Eulerscher Kreis betritt und verlässt jeden Knoten auf je einer Kante

"⇐": Konstruktiv: Besuche Knoten $v_1 \dots v_k$ bis keine Kante mehr.
Jeder Knoten gerader Grad ⇒ $v_1 = v_k$
Alle Knoten besucht? ja: fertig
nein: starte bei unbesuchter Kante.



Handschlaglemma

gerichtet: $\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$

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

Abschätzungen

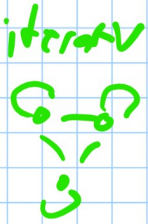
allgemein $|E| \in \Theta(|V|^2) =: \Theta(V^2)$

Zusammenhängend $|E| \in \Omega(V)$

vollständig $|E| = \frac{|V|(|V|-1)}{2} = \Theta(V^2)$

maximal $|E| = |V|^2$ gerichtet

$|E| = \sum_{i=1}^{V} i = \frac{|V|(|V|+1)}{2}$ ungerichtet

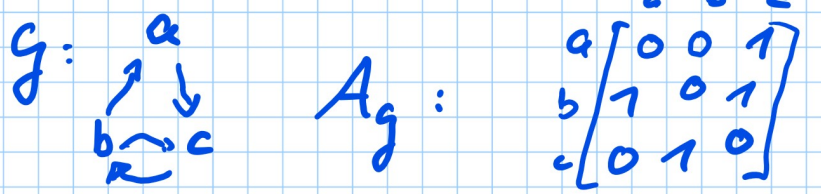


Repräsentation von Graphen $G = (V, E)$

(a) Adjazenz-Matrix M: $V \hat{=} \{1, 2, \ldots, n\}$

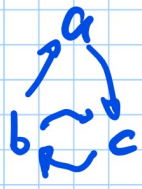
$A = (a_{ij})_{i,j=1\ldots n}$ $a_{ij} = \mathbb{1}\{(i,j) \in E\}$

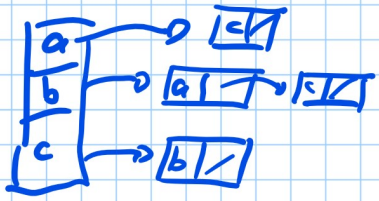
Speicher $\Theta(|V|^2)$



(b) Adjazenzliste Adj

Adj[u] speichert $N^+(u)$

G: 

Adj(G): 

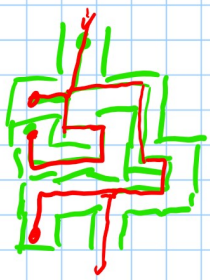
Speicher $\Theta(|V|+|E|)$

Laufzeiten einfacher Operationen

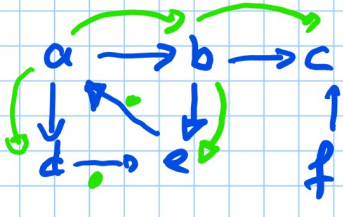
G gerichtet, $|V|=n$

	Matrix	Liste
Nachfolger von v	$\Theta(n)$	$\Theta(\deg^+ v)$
$u \in V$ ohne Nachfolger	$\Theta(n^2)$	$\Theta(n)$
$(u,v) \in E$?	$\Theta(1)$	$\Theta(\deg^+ u)$
Kante (u,v) einfügen	$\Theta(1)$	$\Theta(1)$
Kante (u,v) löschen	$\Theta(1)$	$\Theta(\deg^+ u)$

Graphen durchlaufen / durchsuchen



Tiefensuche z.B. Labyrinth



(alphabetische Reihenfolge)

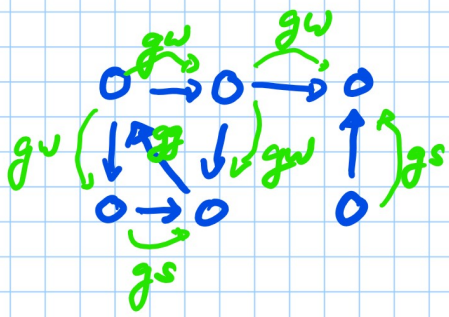
a, b, c, e, d, f

Farben (konzeptuell)

weiss: Knoten noch unbesucht

grau: Knoten in Bearbeitung

schwarz: Knoten besucht + fertig.



g→w: neu entdeckt (Baumkante)

g→g: Zyklus!

g→s: Vorwärtskante oder Querkante

Algorithmus DFS-Visit(v)

```
v.color ← grey
foreach w ∈ N+(v):
    if w.color = white:
        DFS-Visit(w)
v.color ← black
```

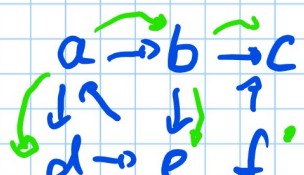
DFS-Visit(G)

```
foreach v ∈ V
    v.color ← white
foreach v ∈ V
    if v.color = white:
        DFS-Visit(v)
```

Laufzeit $\Theta(|V| + \sum_{v \in V} \deg^+(v)) = \Theta(|V| + |E|)$

(iterativer Algorithmus simuliert Rekursion mit einem Stack)

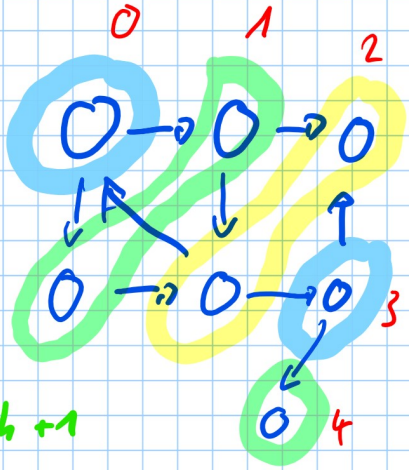
Breitensuche



a b d c e f

BFS-Visit (v)

```
Queue Q ← ∅
v.color ← grey
Q.enq(v)
While Q ≠ ∅ :
    w ← Q.deq()
    foreach c ∈ N+(w)
        if (c.color = white)
            c.color ← grey   { c.depth ← w.depth + 1
                               c.parent ← w
            Q.enq(c)
        elsif c.color = grey: cycle
    w.color ← black
```



BFS-Visit (Graph G)

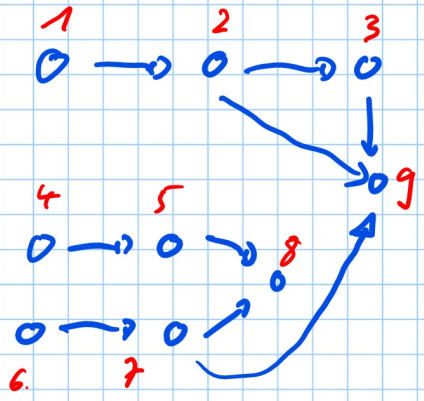
```
foreach u ∈ V
    u.color ← white
    u.parent ← null
    u.depth ← 0
foreach u ∈ V
    if u.color = white
        BFS-Visit(v)
```

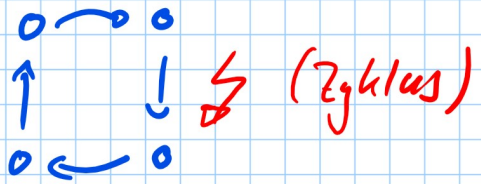
Topologisches Sortieren

Top. Sort. eines azykl. gerichteten Graphen $G = (V, E)$:

bijektive Abbildung $ord : V \to \{1, \dots, n\}$

s.d. $ord(v) < ord(w) \quad \forall (v,w) \in E$





Lemma: $G = (V, E)$ top. sortierbar

$\Leftrightarrow$ G kreisfrei

"$\Rightarrow$" G mit Kreis $\langle v_1 .. v_k \rangle$: $ord(v_i) < ord(v_{i+1})$

$\forall i = 1..k$ und

$ord(v_k) < ord(v_1)$ ↯

"$\Leftarrow$" Induktion über $|V| = n$

$n = 1$: klar

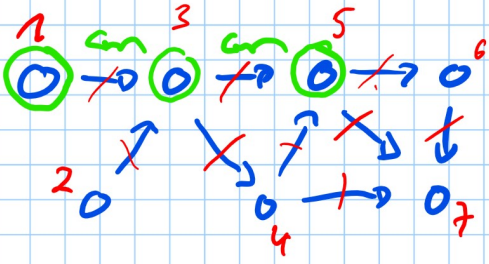
G_n sortierbar $\Rightarrow$ G_{n+1} sortierbar:

G_{n+1} enthält Knoten v_0 mit $deg^-(v) = 0$

verfolge Kanten rückwärts $\Rightarrow$ nach max Iterationen muss Schluss sein (Sonst Zyklus!)

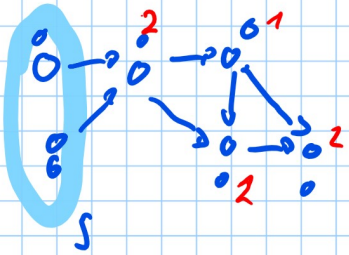
Graph ohne v_0 sortierbar → verwende Sortierung von G_n und setze $ord(v_i) \leftarrow ord(v_i)+1$ $\forall i$;
$ord(v_0) \leftarrow 1$.

Algorithmus Idee



$|V|$ Schritte pro Knoten
$\Theta(|V|^2)$

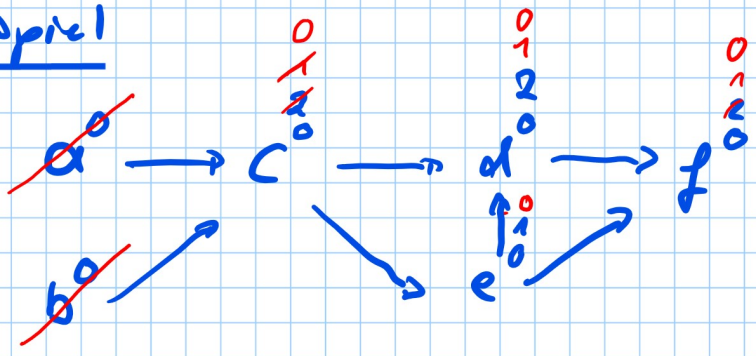
Verbesserung? → Eingangsgrade merken



Algorithmus Top. Sort

Stack $S \leftarrow \emptyset$
foreach $v \in V$: $A[v] \leftarrow 0$
foreach $(v,w) \in E$: $A[w] \leftarrow A[w]+1$
foreach $v \in V$ mit $A[v]=0$: $S.push(v)$
$i \leftarrow 1$
while $S \neq \emptyset$ do
  $v \leftarrow S.pop()$
  $ord[v] \leftarrow i++$
  foreach $(v,w) \in E$ [↔ foreach $w \in N^-(v)$]
    $A[w] \leftarrow A[w]-1$
    if $A[w]=0$
      $S.push(w)$
if $i = |V|+1$ return ord else "cycle"

Beispiel



S
a
b
c
e
d
f