Von letztem Termin:

DFS-Visit(u): // rekursives Durch laufen

Pre-Visit(u)

markiere u

FOR $(u,v) \in E$, v nicht markiert

DFS-Visit(v)

Post-Visit(u)

DFS(G): // Rahmen programm

FOR $u \in V$, nicht markiert

DFS-Visit(u)

Anfangs- und Endzeit von DFS-Visit(u) aufzeichnen

Pre-Visit(u): pre[u] ← T ; T ← T+1

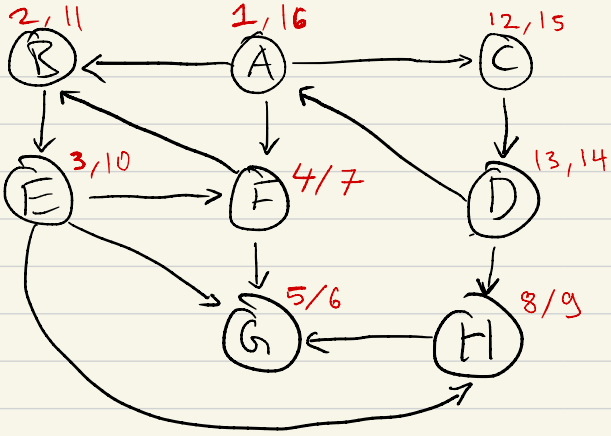
Post-Visit(u): post[u] ← T ; T ← T+1

(pre-/post order des Rekursionsbaum)

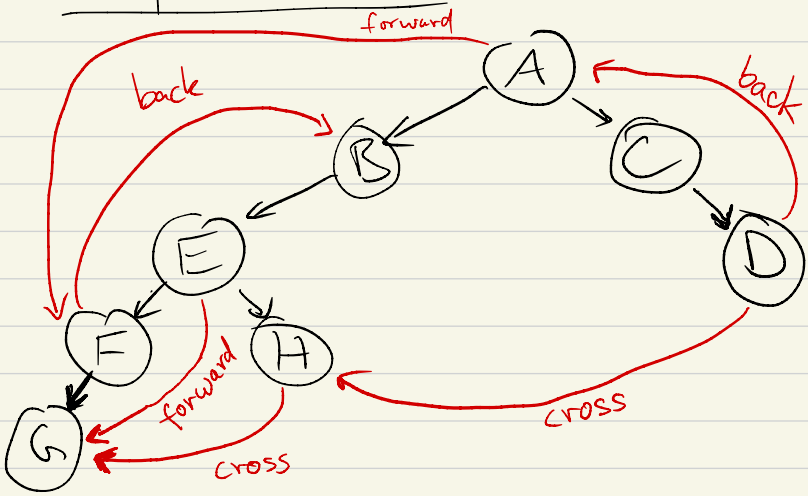
topologische Sortierung: post order, umgekehrt

Beispiel:

pre / post

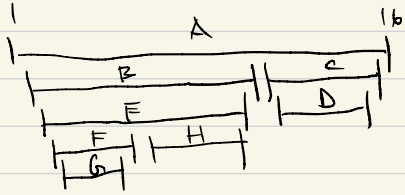


Tiefensuchbaum



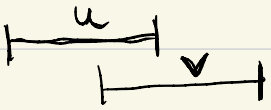
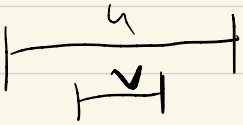
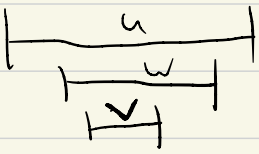
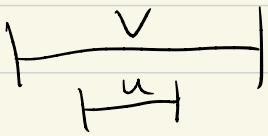
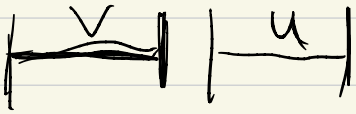
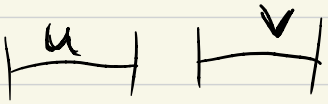
DFS-baum $\simeq$ Verschachtelung der Intervalle I_u

$$I_u = \{ pre[u], \ldots, post[u] \}$$



Klassifizierung der Kanten

Kante $(u, v) \in E$

I_u vs I_v	Klassifizierung
5. 	nicht möglich (Aufruf für v muss enden vor Aufruf für u)
3. 	forward oder DFS-Baum
4.  (w beliebig)	forward z.B. (A, F)
1. 	back z.B. (F, B)
2. 	cross z.B. (D, H)
6. 	nicht möglich (v noch nicht markiert als (u,v) betrachtet wurde)

azyklisch ④

unger. Z+K

Beobachtungen:

$\exists$ "back" Kante $\Rightarrow$ $\exists$ gerichteter Zyklus (1)

$(u,v) \in E$, nicht "back" $\Rightarrow$ $post[u] \geq post[v]$ (2)

(1), (2) $\rightsquigarrow$ $\nexists$ ger. Zyklus $\Leftrightarrow$ umgekehrte post order ist topologische Sortierung

$\Leftrightarrow \nexists$ "back" Kante

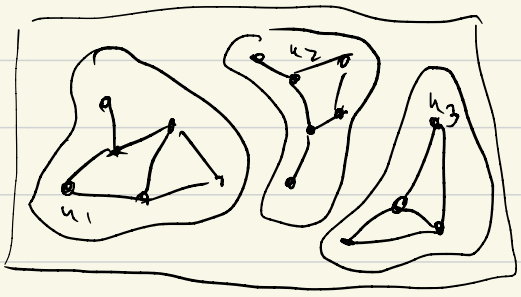
DFS auf unger. Graphen

"back" Kanten = "forward" Kanten (umgekehrt durchlaufen)

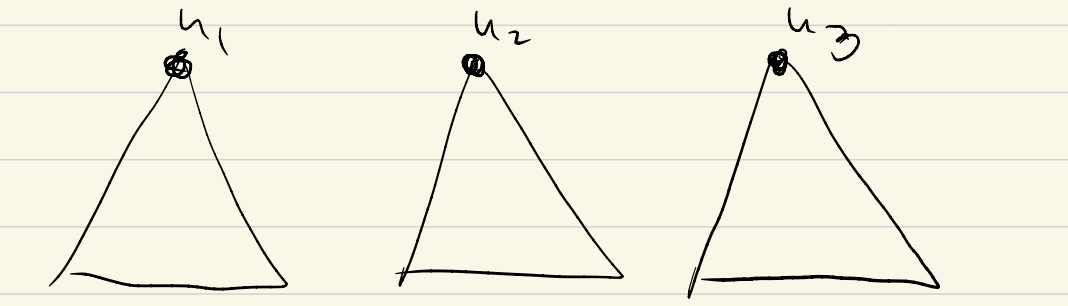
"cross" Kanten: nicht möglich

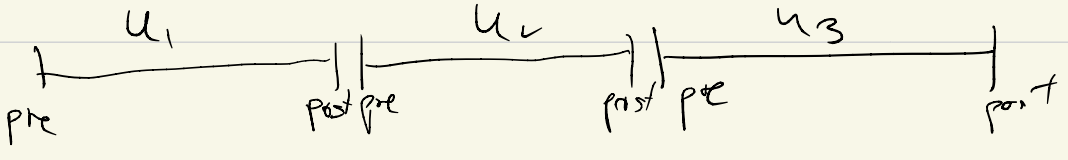
Zusammenhangskomponenten:

G

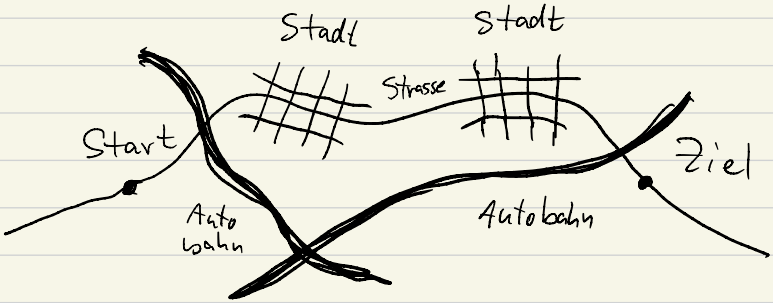


DFS Wald



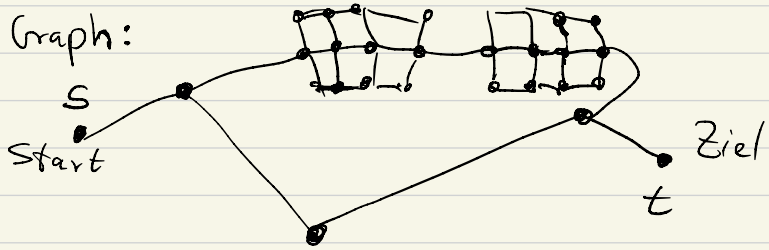


Strassen netzwerk



gesucht: Weg mit wenig Kreuzungen

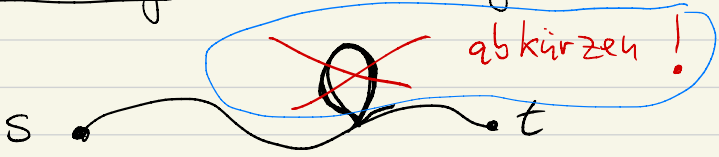
als Graph:



gesucht: Weg mit wenig Knoten / Kanten

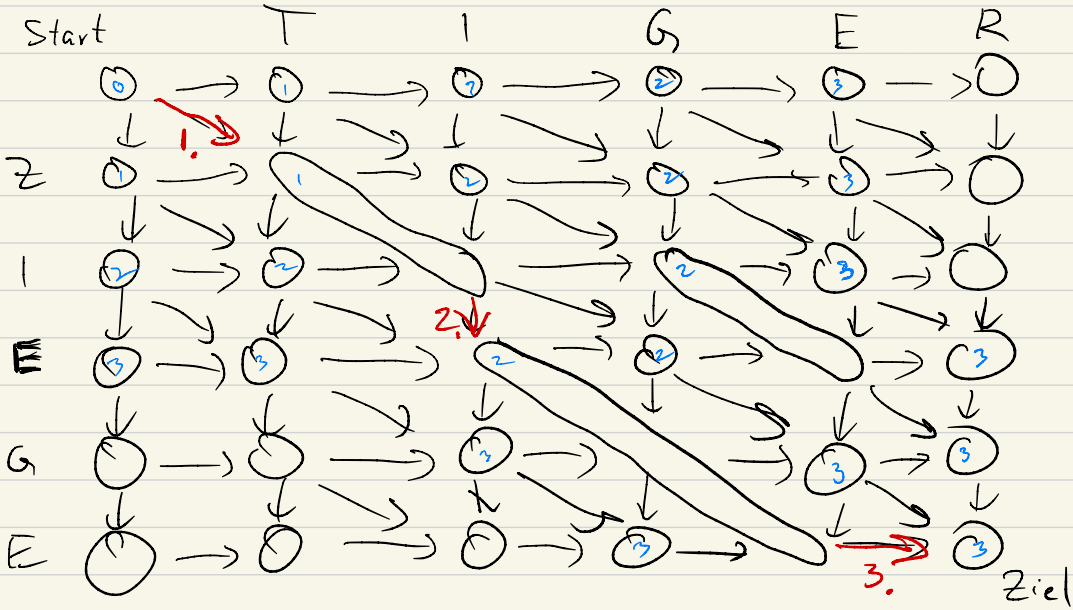
gerichtete Kanten möglich (Einbahnstrasse)

Beobachtung: Kürzester Weg ist immer Pfad



weiteres Beispiel:

minimale Editierdistanz



jede Kante steht für mögliche Editier-operation

gefundener Weg (Sequenz von Editierop.):

T I G E R
Z I G E R
Z I E G E R
Z I E G E

1. Tausche T und Z
2. Füge E hinzu
3. Entferne R

Formal: ger. Graph $G = (V, E)$

dist(u, v) = Distanz von u nach v in G

= Länge des kürzesten Wegs von u nach v

$dist(s, v) = 0 \iff v = s$

$dist(s, v) = 1 \iff v$ ist Nachfolger von s

$dist(s, v) = 2 \iff v$ ist Nachfolger eines Nachfolger von s, aber kein Nachfolger von s

sei $S_k = \{ v \in V \mid d(s,v) = k \}$ "level k"

angenommen wir kennen $S_0, S_1, \dots, S_{k-1}$

$v \in S_k \iff v$ ist Nachfolger von Knoten in S_{k-1} und $v \notin S_0 \cup \dots \cup S_{k-1}$

Algorithmus:

$S_0 \leftarrow \{s\}$

for $k = 1$ to $|V| - 1$:

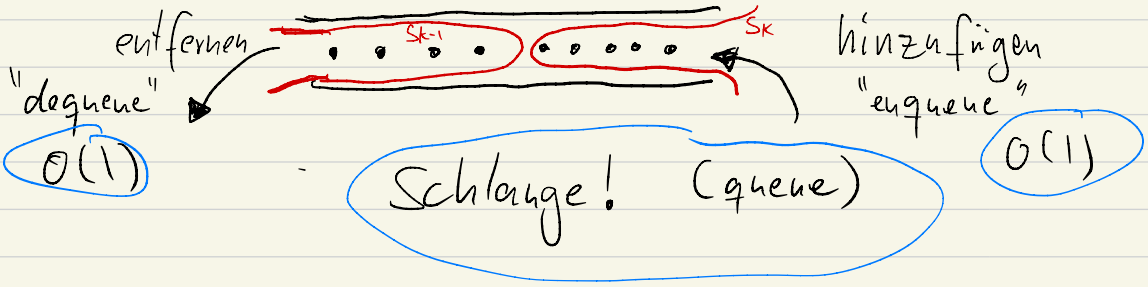
  for $u \in S_{k-1}$:

    for $(u, v) \in E$, $v \notin S_0 \cup \dots \cup S_{k-1}$

      $S_k \leftarrow S_k \cup \{v\}$

effiziente / elegante Implementierung

- markiere Knoten sobald Distanz bekannt
- geteilte Datenstruktur um
  - S_{k-1} abzuarbeiten
  - S_k aufzubauen



BFS(s): (Breitensuche)

$S \leftarrow \{s\}$

while $S \neq \emptyset$

$u \leftarrow$ dequeue(S)

if u nicht markiert

markiere u

for $(u,v) \in E$, v nicht markiert

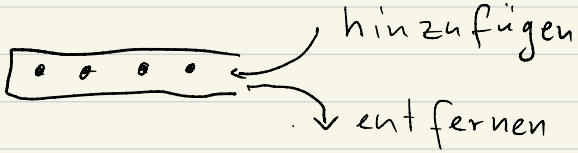
enqueue(S, v)

parent$[v] \leftarrow u$ falls parent$[v]$ noch nicht definiert

Knoten können in Schlange mehrmals auftreten

Wie passt DFS ins Bild:

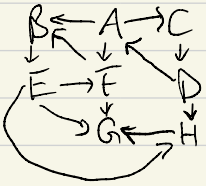
statt Schlange verwende Stapel (Stack)

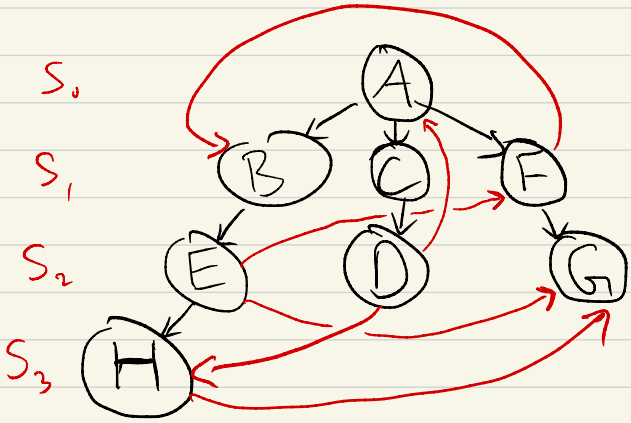


zuletzt hinzugefügt

≃ "tiefster" Knoten im Graph bislang

Breiten such baum





Beobachtung: Kante von S_k nach $S_{k'}$

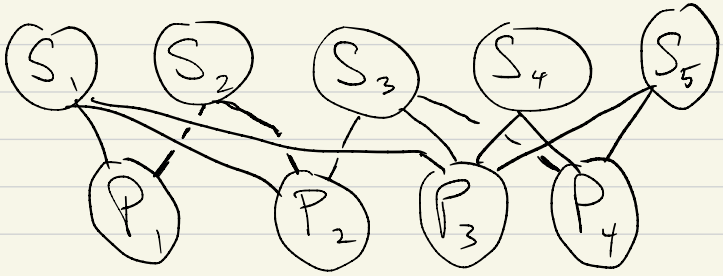
$\Rightarrow k' \leq k+1$

(ungerichtet: $k-1 \leq k' \leq k+1$)

Terminplanung

Studenten

Prüfungen

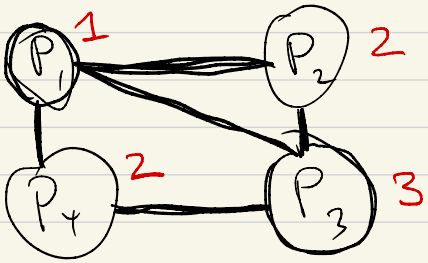


wieviele verschiedene Termine nötig?

welche Konflikte bestehen

besserer Graph:

"Konfliktgraph"

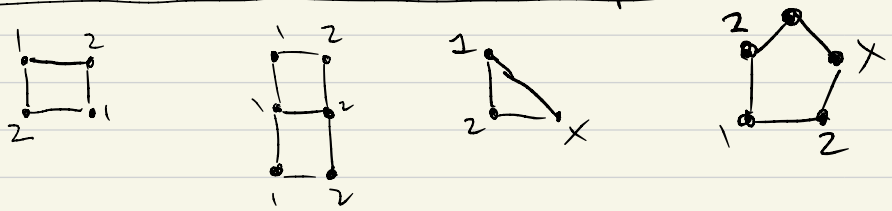


färbe Knoten

- an jeder Kante sind verschiedene Farben
- möglichst wenig verschiedene Farben

"Knoten färbungs problem"

Wann reichen zwei Farben (bipartit)?



Beobachtung: G bipartit

$\Rightarrow$ G enthält keinen ungeraden Zyklus

tatsächlich: "$\Leftarrow$" gilt auch

Beweis per Algorithmus!

- berechne BFS levels $S_0, S_1, \dots, S_{n-1}$ von s
- Farbe 1: S_0 S_2 $\cdots$ S_t (t gerade)

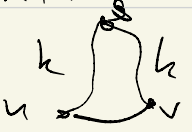
  Farbe 2: S_1 S_3 $\cdots$ S_t (t ungerade)

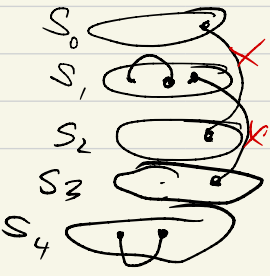
Behauptung: G enthält keinen ungeraden Zyklus
$\Rightarrow$ jede Kante hat verschiedene Farben

Beweis (indirekt): angenommen $uv \in E$, u,v gleiche Farbe

$\Rightarrow$ u,v im gleichen Level S_k

$\Rightarrow$ $\exists$ Zyklus der Länge $2k+1$





nicht behandelt:

Wann reichen 3 Farben?

- NP-vollständig zu entscheiden
- bester bekannter Algorithmus ist expo.
- polynomieller Alg. würde P = NP bedeuten