Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und
wissensbasierte Systeme

Institut für Informatik

Algorithmen und Datenstrukturen

Wintersemester 2018/19

20. Vorlesung

Tiefensuche und topologische Sortierung

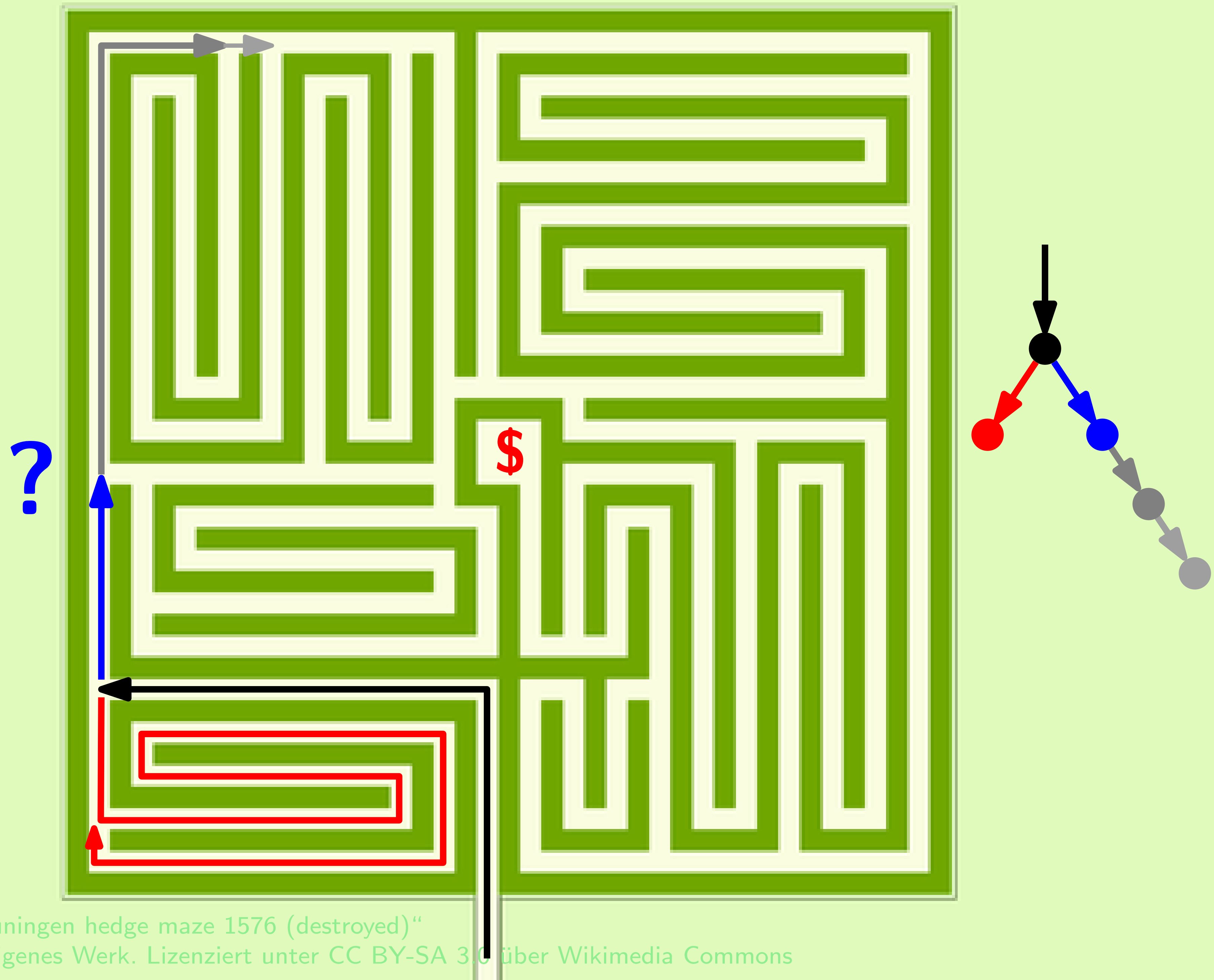
Prof. Dr. Alexander Wolff *Lehrstuhl für Informatik I*

Themen für den 3. Kurztest (Do, 24.01.18)

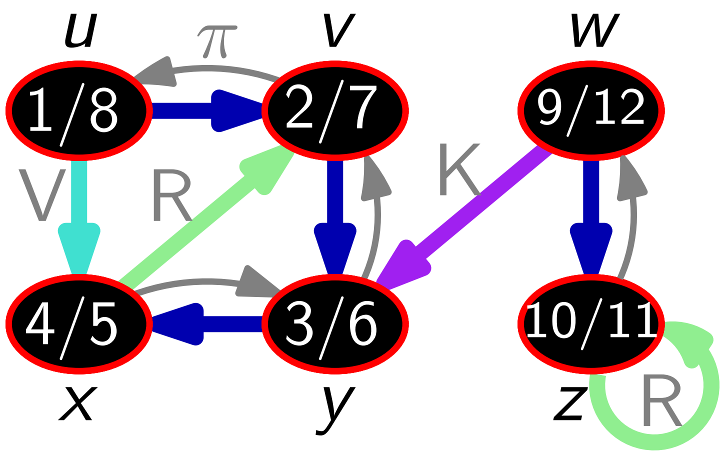
- Rot-Schwarz-Bäume (R-S-Eigenschaften, Höhe)
- Augmentieren von Datenstrukturen
- Amortisierte Analyse
- Nächstes Paar (Teile und Herrsche)
- Graphen und Breitensuche

Anmeldung

Ab sofort bis Di, 22.1., 13:00 Uhr.



Tiefensuche

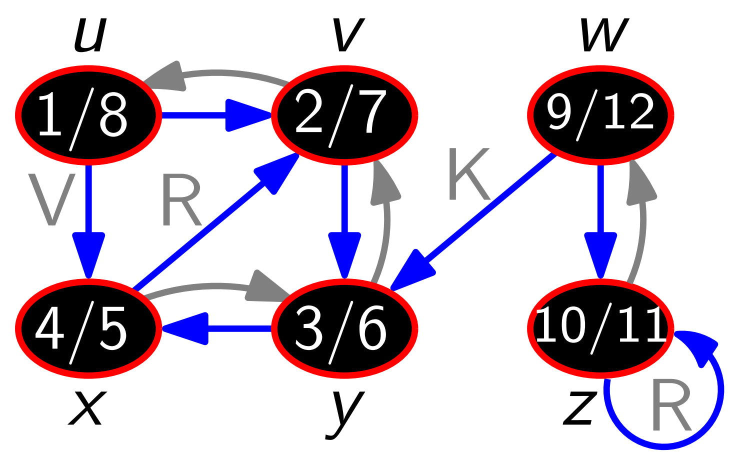


Eingabe: (un)gerichteter Graph G

Ausgabe:
- Besuchsintervalle $(u.d\,/\,u.f)$ (discovery time / finish time)
- DFS-Wald $\left(\xleftarrow{\pi}\right)$
- Klassifizierung der Graphkanten:

	Farbe Zielknoten:
● Baumkanten (Kanten von G_π) Kanten des DFS-Waldes (entgegen π gerichtet)	*weiss*
● Rückwärtskanten (R) Nicht-Baumkanten zu einem Vorgängerknoten	*grau*
● Vorwärtskanten (V) Nicht-Baumkanten zu einem Nachfolgerknoten	*schwarz und* start.d < ziel.d
● Kreuzkanten (K) Kanten, bei denen kein Endpunkt Vorgänger des anderen ist.	*schwarz und* start.d > ziel.d

Tiefensuche – Pseudocode



DFS(Graph $G = (V, E)$)
- **foreach** $u \in V$ **do**
  - $u.color = white$
  - $u.\pi = nil$
- $time = 0$ // globale Variable!
- **foreach** $u \in V$ **do**
  - **if** $u.color == white$ **then** DFSVisit(G, u)

DFSVisit(Graph G, Vertex u)
- $time = time + 1$
- $u.d = time$; $u.color = gray$
- **foreach** $v \in \mathrm{Adj}[u]$ **do**
  - **if** $v.color == white$ **then**
    - $v.\pi = u$; DFSVisit(G, v)
- $time = time + 1$
- $u.f = time$; $u.color = black$

Laufzeit von DFS?

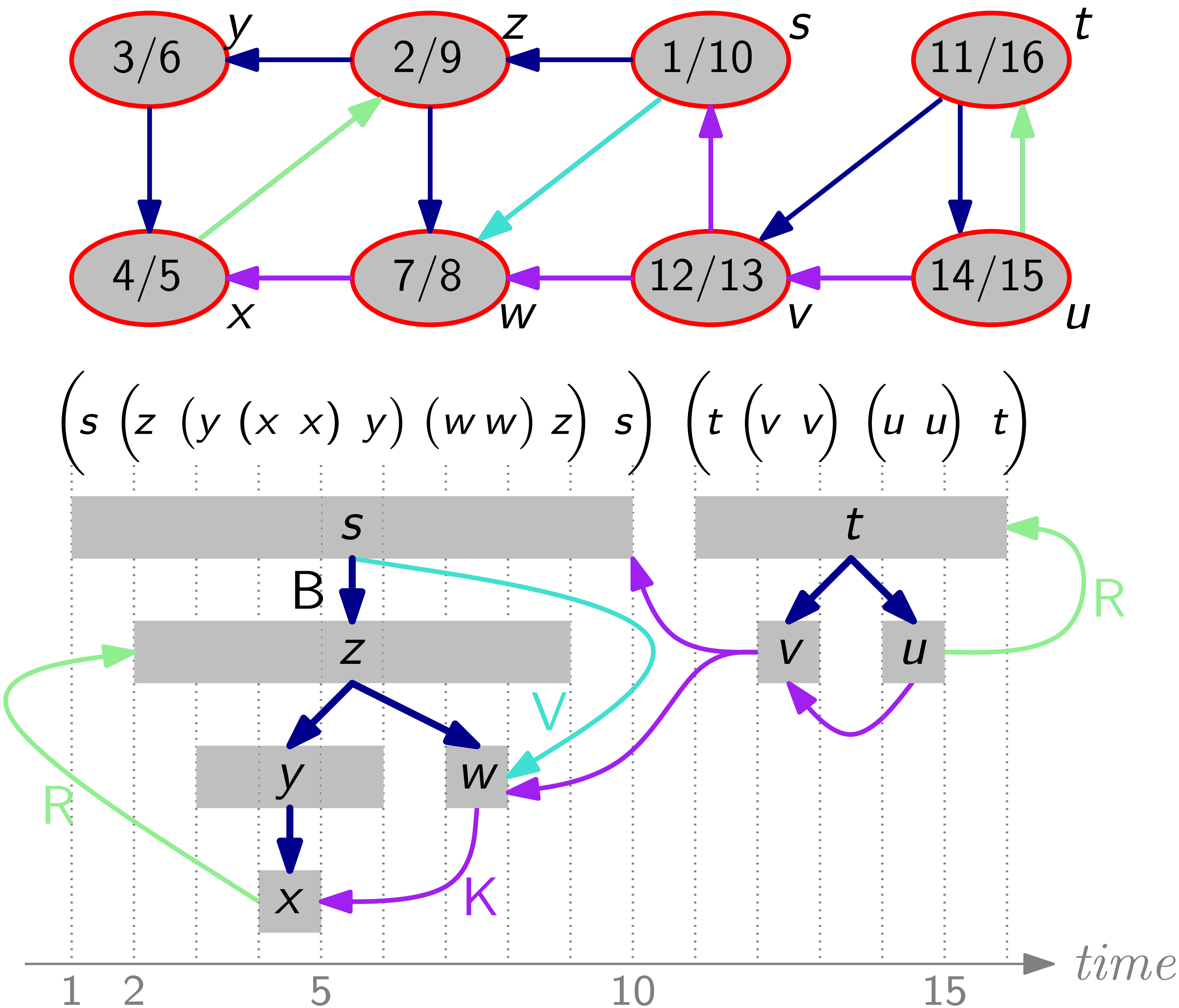
- DFSVisit wird nur für weiße Knoten aufgerufen.
- In DFSVisit wird der neue Knoten sofort grau gefärbt.

$\Rightarrow$ DFSVisit wird für jeden Knoten genau $1\times$ aufgerufen.

- DFS ohne **if** $O(V)$ Zeit
  DFSVisit ohne Rek. $O(\deg u)$ (out-)

DFS gesamt	$O(V + E)$ Zeit

Tiefensuche – Eigenschaften



Tiefensuche – Analyse

DFSVisit(Graph G, Vertex u)
  $time = time + 1$
  $u.d = time$; $u.color = gray$
  foreach $v \in \mathrm{Adj}[u]$ **do**
    if $v.color == white$ **then**
      $v.\pi = u$; DFSVisit(G, v)
  $time = time + 1$
  $u.f = time$; $u.color = black$

Satz. (Klammerntheorem)

Nach DFS(G) gilt für $\{u, v\} \in \binom{V}{2}$ genau eine der Bedingungen

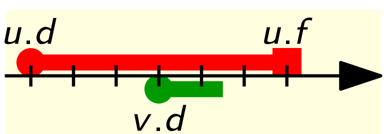
(i) Besuchsintervalle disjunkt und
Baumkanten enthalten weder u-v- noch v-u-Weg.

(ii) $[u.d, u.f] \subset [v.d, v.f]$ und Baumkanten enthalten v-u-Weg.

(iii) Wie (ii), nur umgekehrt.

Beweis. Wir betrachten zwei Fälle.

1. Fall: $u.d < v.d$.



A) $v.d < u.f$, d.h. v wurde entdeckt, als u noch grau war.
$\Rightarrow$ v ist *Nachfolger* von u, d.h. es gibt einen u-v-Weg.

Wegen $u.d < v.d$ gilt: v wurde später als u entdeckt.

$\Rightarrow$ alle Kanten, die v verlassen, sind erforscht;
v wird schwarz, *bevor* DFS zu u zurückkehrt und u
schwarz macht $\Rightarrow$ $[v.d, v.f] \subset [u.d, u.f]$, d.h. (ii) ✓

Tiefensuche – Analyse

```
DFSVisit(Graph G, Vertex u)
  time = time + 1
  u.d = time; u.color = gray
  foreach v ∈ Adj[u] do
    if v.color == white then
      v.π = u; DFSVisit(G, v)
  time = time + 1
  u.f = time; u.color = black
```

Satz. (Klammerntheorem)

Nach DFS(G) gilt für $\{u, v\} \in \binom{V}{2}$ genau eine der Bedingungen

(i) Besuchsintervalle disjunkt und Baumkanten enthalten weder u-v- noch v-u-Weg.

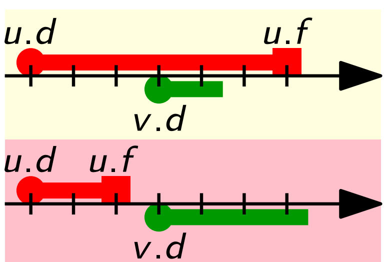
(ii) $[u.d, u.f] \subset [v.d, v.f]$ und Baumkanten enthalten v-u-Weg.

(iii) Wie (ii), nur umgekehrt.

Beweis. Wir betrachten zwei Fälle.

1. Fall: $u.d < v.d$. ✓ 2. Fall: $v.d < u.d$. Symmetrisch! ✓

(Vertausche im Beweis $u \leftrightarrow v$.)



A) $v.d < u.f$. ✓

B) $u.f < v.d$. ✓

Laut Code gilt außerdem $u.d < u.f < v.d < v.f$

$\Rightarrow [u.d, u.f] \cap [v.d, v.f] = \emptyset$

$\Rightarrow$ Keiner der beiden Knoten wurde entdeckt, während der andere noch grau war, d.h. keiner Nachf. des anderen.

(i) $\Uparrow$

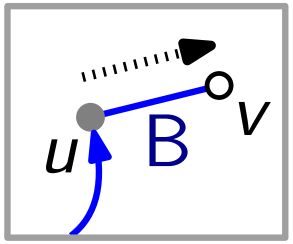
Tiefensuche in ungerichteten Graphen

Satz. G ungerichtet
$\Rightarrow$ G hat nur Baum- und Rückwärtskanten.

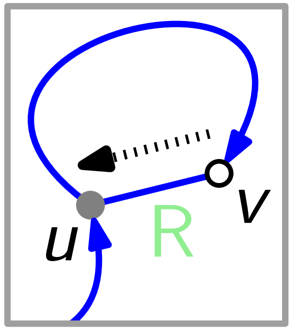
Beweis. Sei uv (kurz für $\{u, v\}$) eine beliebige Kante von G.

O.B.d.A. gilt $u.d < v.d$.

Dann entdeckt DFS v und färbt v schwarz, bevor u schwarz gefärbt wird (da $v \in \mathrm{Adj}[u]$).



- Falls DFS uv zum ersten Mal von u nach v überschreitet, ist v zu diesem Zeitpunkt *weiss*. Dann ist uv Baumkante.

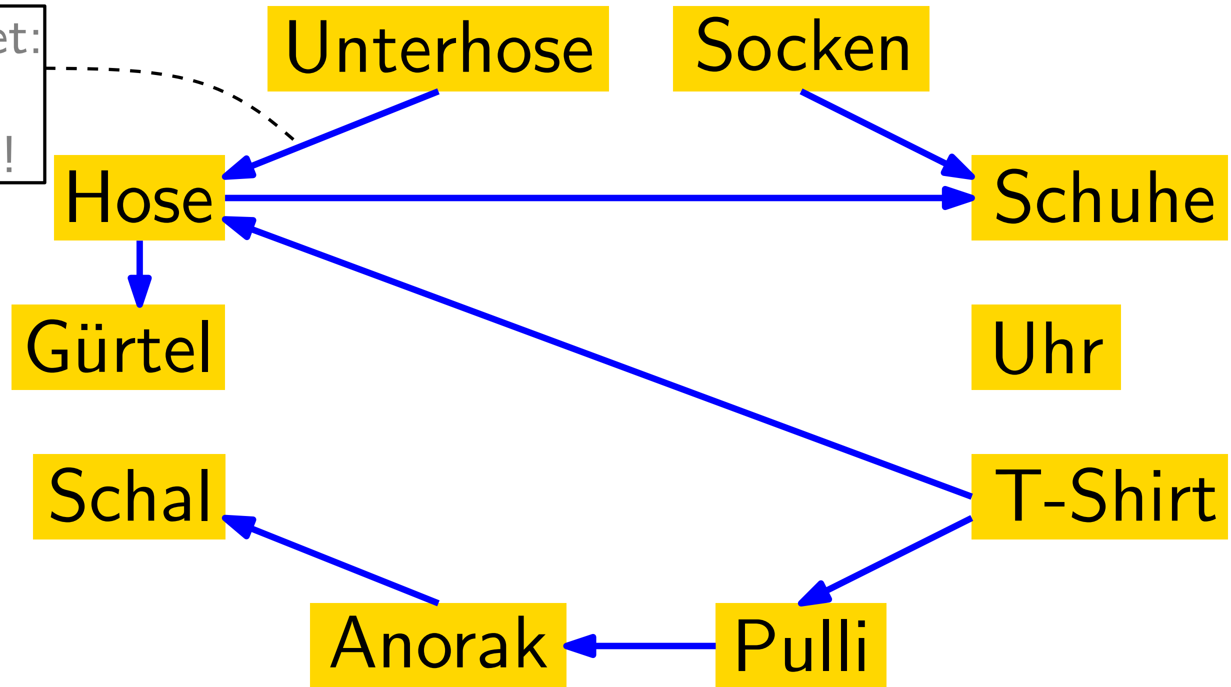


- Andernfalls wird uv zum ersten Mal von v nach u überschritten. Dann ist uv R-Kante, da u dann schon (und immer noch) *grau* ist.

$\square$

Ablaufplanung

Kante bedeutet:
Unterhose *vor*
Hose anziehen!

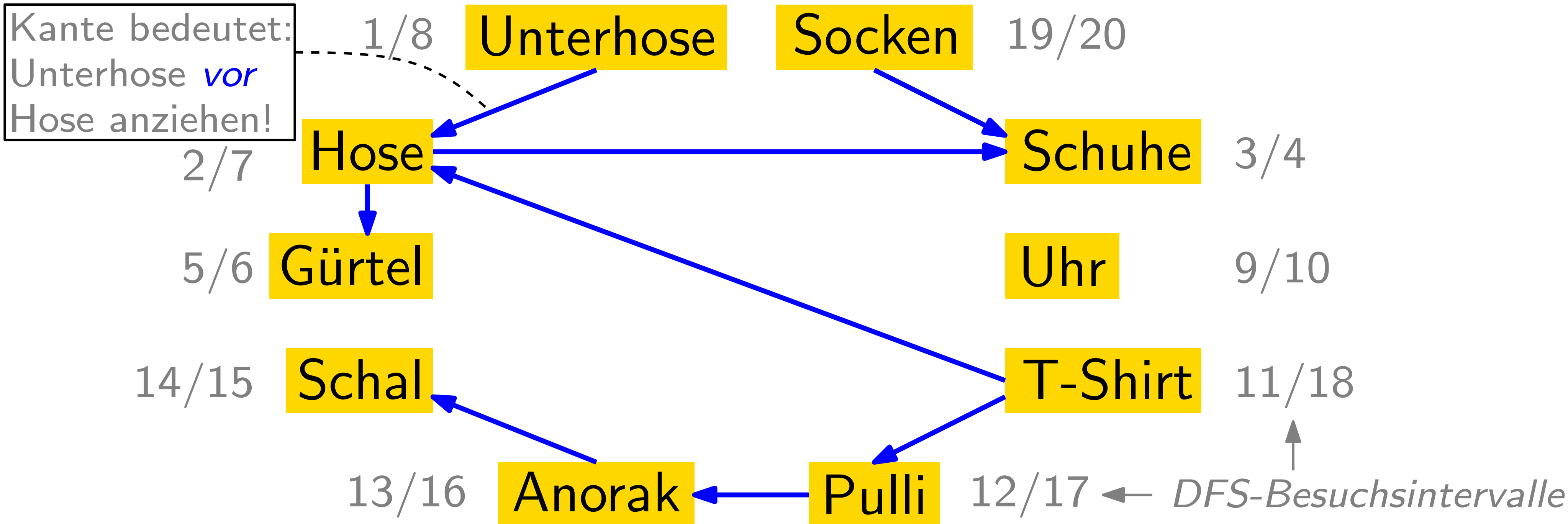


Aufgabe: Finde Ablaufplan –

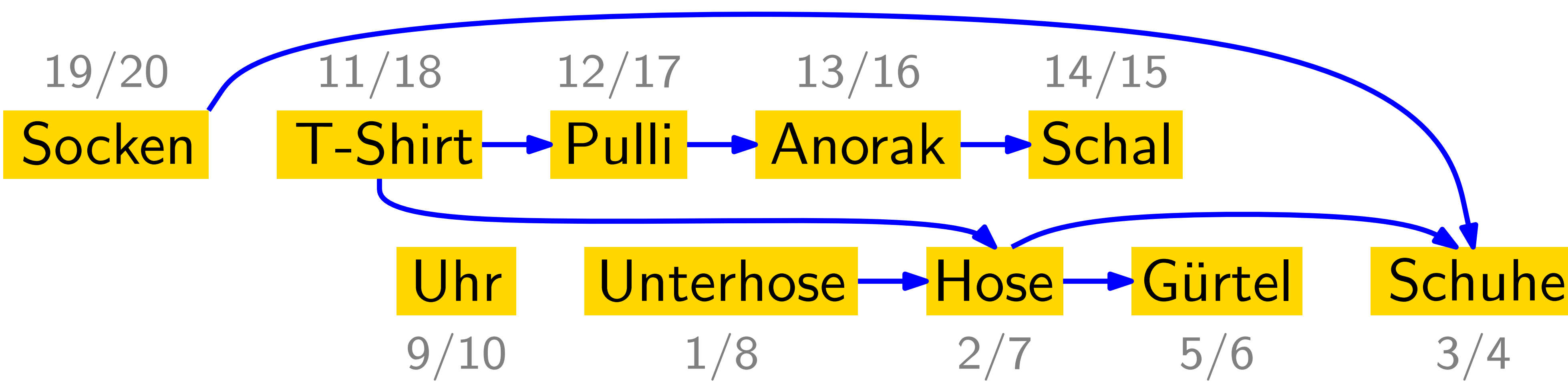
d.h. Reihenfolge der Knoten, so dass alle Einschränkungen erfüllt sind (z.B. T-Shirt vor Pulli).

Topologische Sortierung: Lineare Ordnung der Knoten, so dass aus $(u, v) \in E$ folgt: u kommt vor v.

Ablaufplanung



Idee: Nutze Tiefensuche! ⇒ Alle Kanten sind nach rechts gerichtet.
Sortiere Knoten nach absteigenden f-Zeiten.



Topologisch sortieren

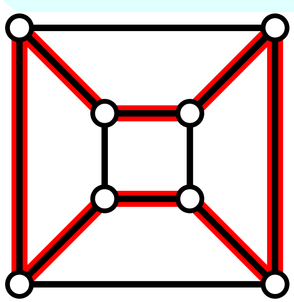
Topologische Sortierung: Lineare Ordnung der Knoten, so dass aus $(u, v) \in E$ folgt: u kommt vor v.

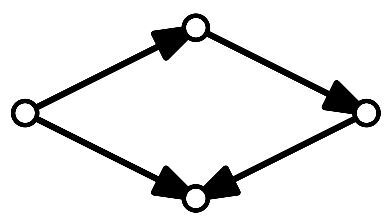
TopologicalSort(DirectedGraph G)
  L = **new** List()
  DFS(G) mit folgender Änderung:
    Wenn ein Knoten schwarz gefärbt wird,
    häng ihn *vorne* an die Liste L an.
  return L

Laufzeit?
$O(V + E)$

Korrekt?
Wann funktioniert's?

Def. Ein (gerichteter) Graph ist *kreisfrei*, wenn er keinen (gerichteten) Kreis enthält.

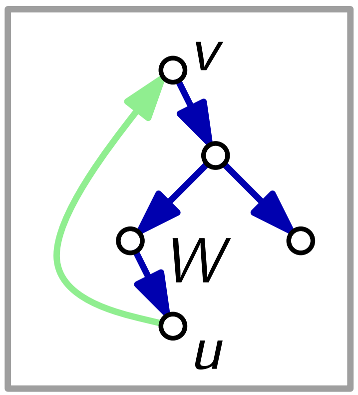

✗


✓

Kreisfrei $\Leftrightarrow$ keine R-Kanten

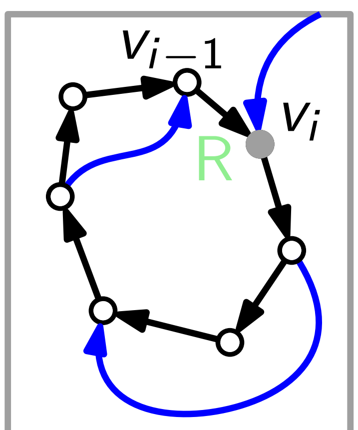
Lem. Ein gerichteter Graph G ist kreisfrei
$\Leftrightarrow$ DFS(G) liefert keine Rückwärtskanten.

Beweis. „$\Rightarrow$" Sei G kreisfrei.



Angenommen DFS(G) liefert R-Kante (u, v).
Dann ist u Nachfolger von v im DFS-Wald.
D.h. G enthält einen gerichteten v-u-Weg W.
Aber dann ist $W \oplus (u, v)$ ein gerichteter Kreis. ↯

„$\Leftarrow$" DFS(G) liefere keine R-Kanten.



Ang. G enthält trotzdem Kreis $C = \langle v_1, \dots, v_k \rangle$.
Sei v_i der 1. Knoten in C, den DFS(G) erreicht.
Es gibt einen Weg von v_i nach v_{i-1} in G.
$\Rightarrow$ DFS gelangt zu v_{i-1}, solange v_i grau ist.
$\Rightarrow$ (v_{i-1}, v_i) ist R-Kante. ↯ $\square$

Korrektheit von TopologicalSort

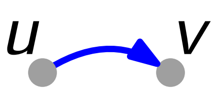
Satz. Sei G ein gerichteter kreisfreier Graph. Dann liefert TopologicalSort(G) eine topologische Sortierung von G.

Beweis. Sei $L = \langle v_n, v_{n-1}, \dots, v_1 \rangle = \text{TopologicalSort}(G)$.

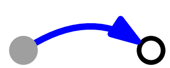
Dann gilt $v_1.f < v_2.f < \dots < v_n.f$.

Sei (u, v) Kante von G. Zu zeigen: $v.f < u.f$

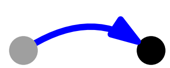
Welche Farbe hat v, wenn DFS (u, v) überschreitet?



– v grau $\Rightarrow$ (u, v) ist R-Kante ↯ Widerspruch zu Lemma: G kreisfrei!



– v weiß $\Rightarrow$ v Nachfolger von u $\Rightarrow$ $v.f < u.f$ ✓



– v schwarz $\Rightarrow$ $u.f$ noch nicht gesetzt, $v.f$ gesetzt

$\Rightarrow$ $v.f < u.f$ ✓

$\square$

Vergleich Durchlaufstrategien für Graphen

	Breitensuche	Tiefensuche
Laufzeit	$O(V+E)$	$O(V+E)$
Ergebnis	BFS-Baum, d.h. kürzeste Wege	d- und f-Werte, z.B. für top. Sortierung
Datenstruktur	Schlange	Rekursion bzw. Stapel
Vorgehen	nicht-lokal	lokal