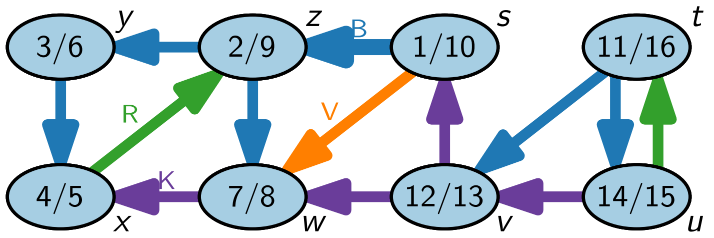
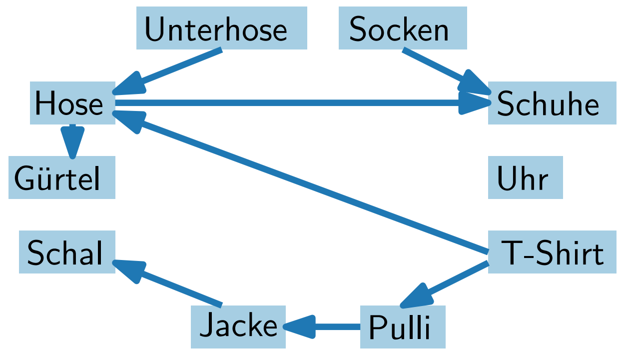
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Algorithmen und Datenstrukturen

Vorlesung 20: Tiefensuche und topologische Sortierung



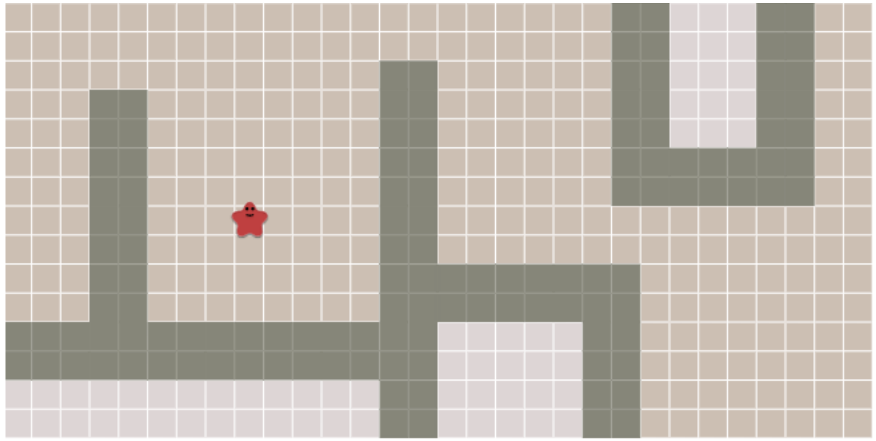


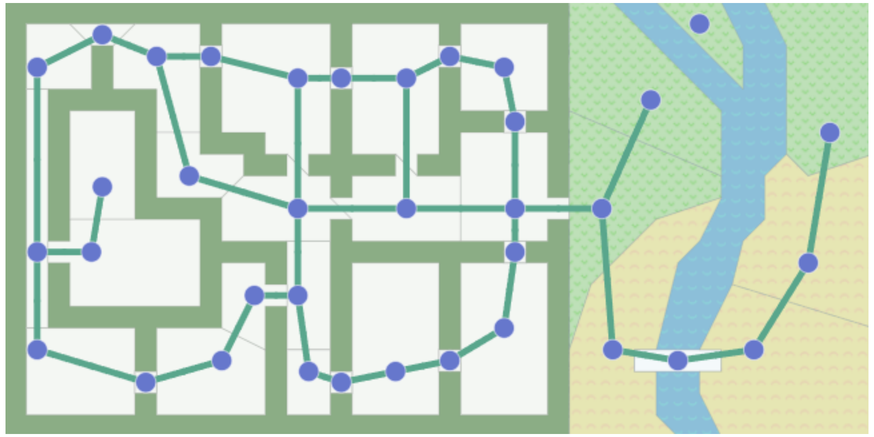
Alexander Wolff

Wintersemester 2025

Wie durchlaufe ich einen Graphen?

Wie finde ich heraus, welche Knoten von einem Startknoten s aus erreichbar sind?



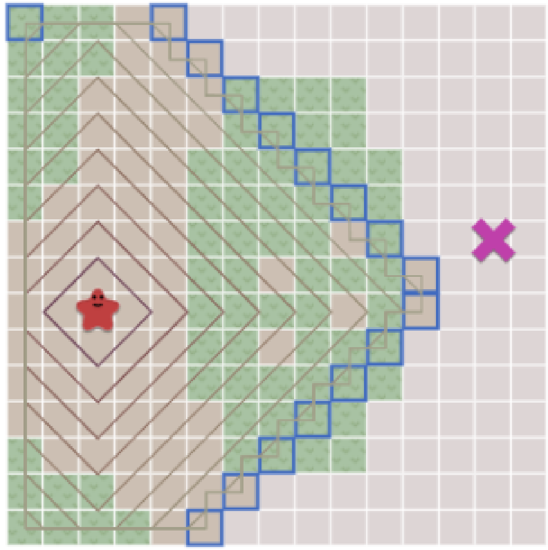


Amit Patel, "Introduction to the A^* Algorithm", Red Blob Games, 2014, https://www.redblobgames.com/pathfinding/a-star/introduction.html

1. wellenförmige Ausbreitung ab s

   Breitensuche (breadth-first search, BFS)

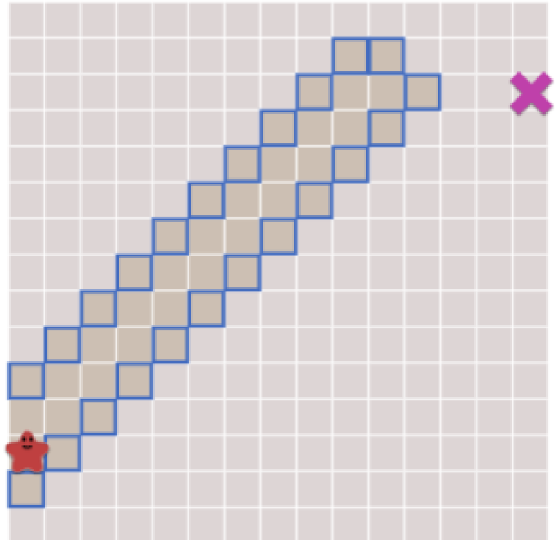
   vorletztes Mal

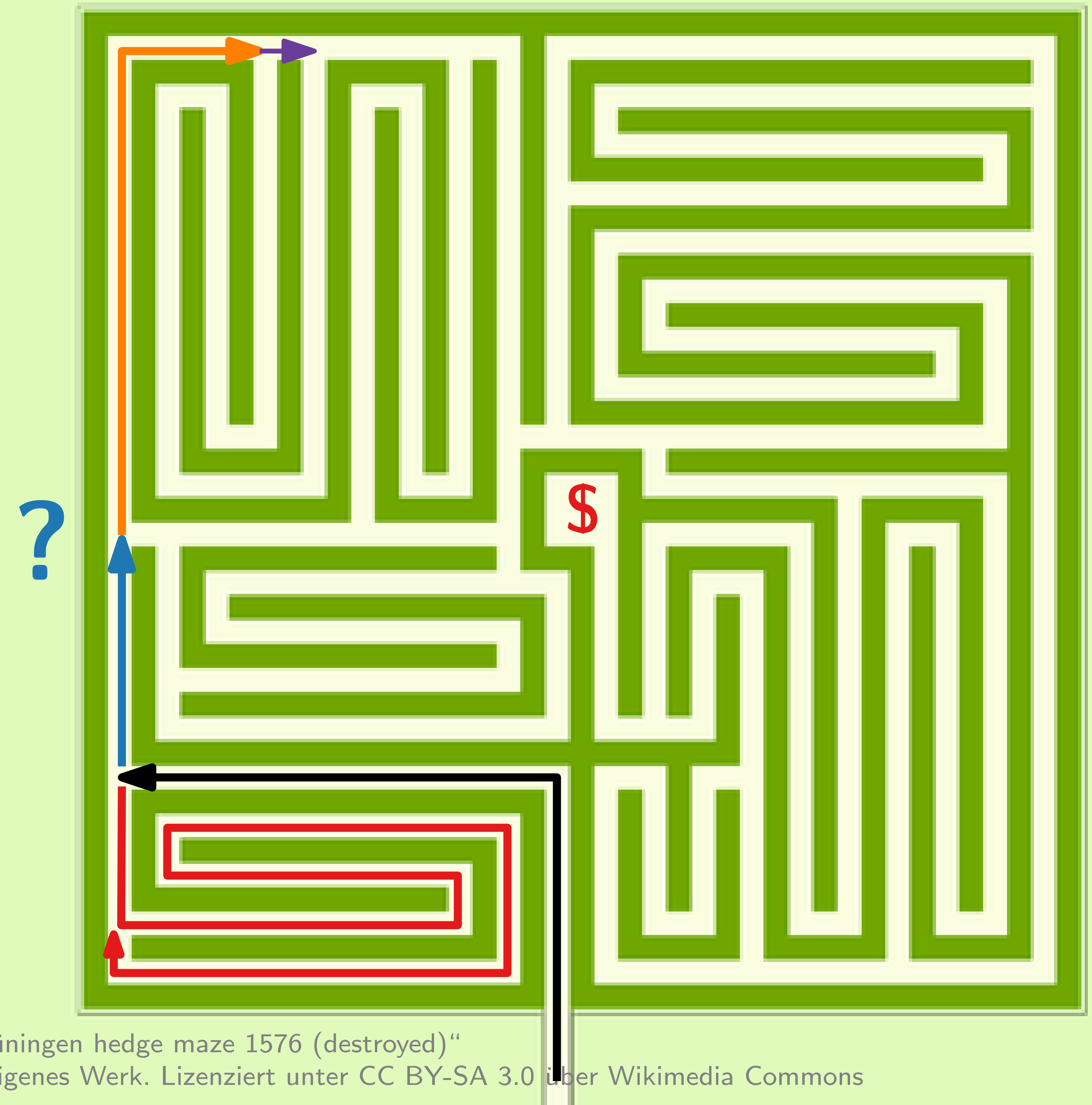


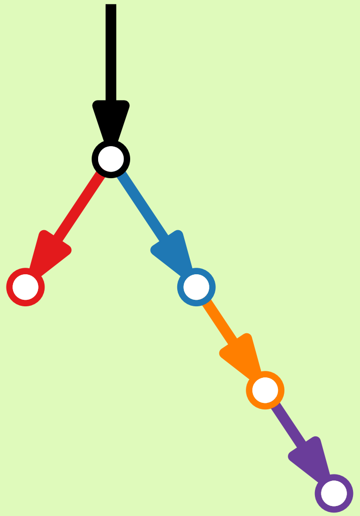
2. von s möglichst schnell weit weg

   Tiefensuche (depth-first search, DFS)

   heute

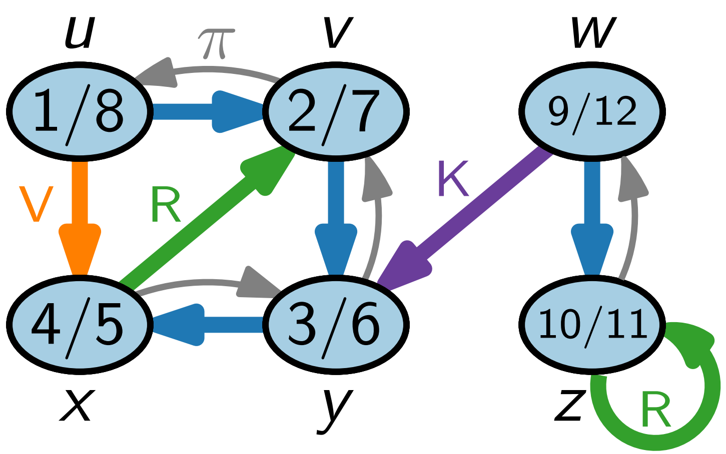






„Maze-01 Grüningen hedge maze 1576 (destroyed)“

Tiefensuche



Eingabe: (un)gerichteter Graph G

Ausgabe:

- Besuchsintervalle $(u.d/u.f)$ — discovery time, finish time
- DFS-Wald $(\xleftarrow{\pi})$
- Klassifizierung der Graphkanten:
  - Baumkanten (Kanten von G_π)
    Kanten des DFS-Waldes (entgegen π gerichtet)
  - Rückwärtskanten (R)
    Nicht-Baumkanten zu einem Vorgängerknoten
  - Vorwärtskanten (V)
    Nicht-Baumkanten zu einem Nachfolgerknoten
  - Kreuzkanten (K)
    Kanten, bei denen kein Endpunkt Vorgänger des anderen ist

Farbe Zielknoten:

weiß

rot

blau und start.$d <$ ziel.d

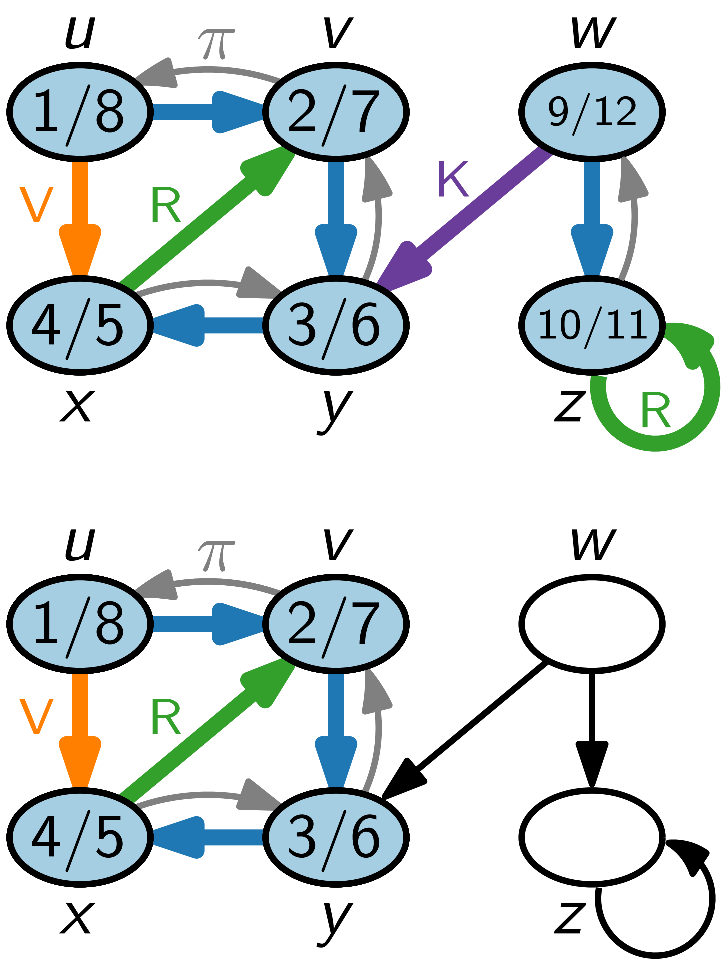
blau und start.$d >$ ziel.d

Tiefensuche – Pseudocode

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

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

$time = 8$



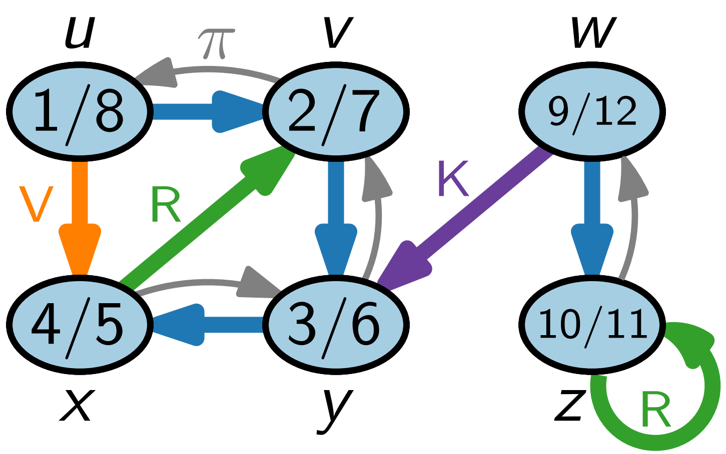
Für jeden Knoten u von G ist
- $u.d$ der Zeitpunkt der Entdeckung,
- $u.f$ der Abschluss-Zeitpunkt;

Besuchsintervall von u ist $[u.d, u.f]$.

Tiefensuche – Pseudocode

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

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



Laufzeit von DFS?

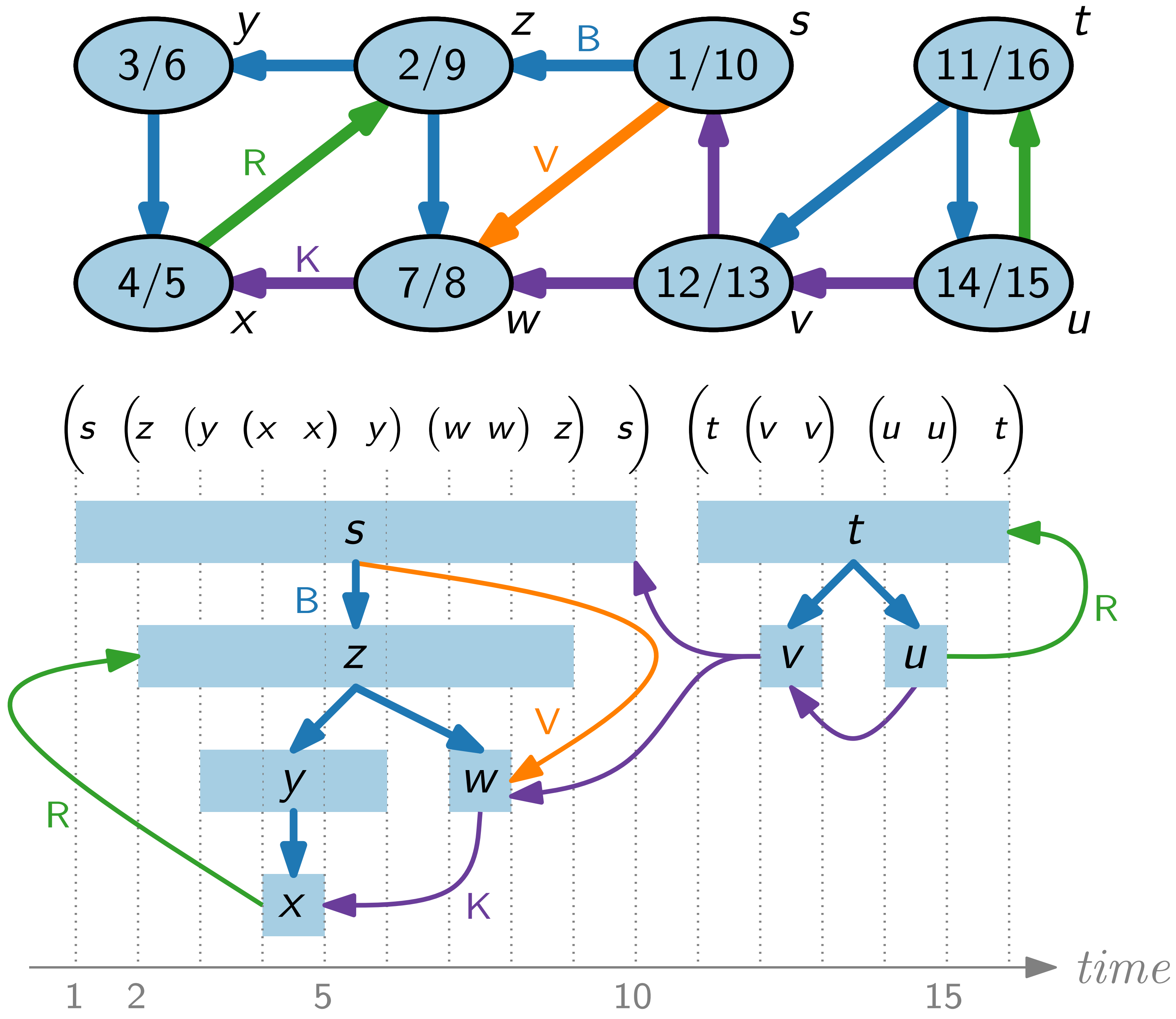
- DFSVisit wird nur für weiße Knoten aufgerufen.
- In DFSVisit wird der neue Knoten sofort rot gefärbt.

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

- DFS ohne **if** $\mathcal{O}(V)$ Zeit

DFS ohne **if**	$\mathcal{O}(V)$ Zeit
DFSVisit ohne Rek.	$\mathcal{O}((\text{out})\deg(u))$
DFS gesamt	$\mathcal{O}(V + E)$ Zeit

Tiefensuche – Eigenschaften



Tiefensuche – Analyse

Satz. (Klammerntheorem)

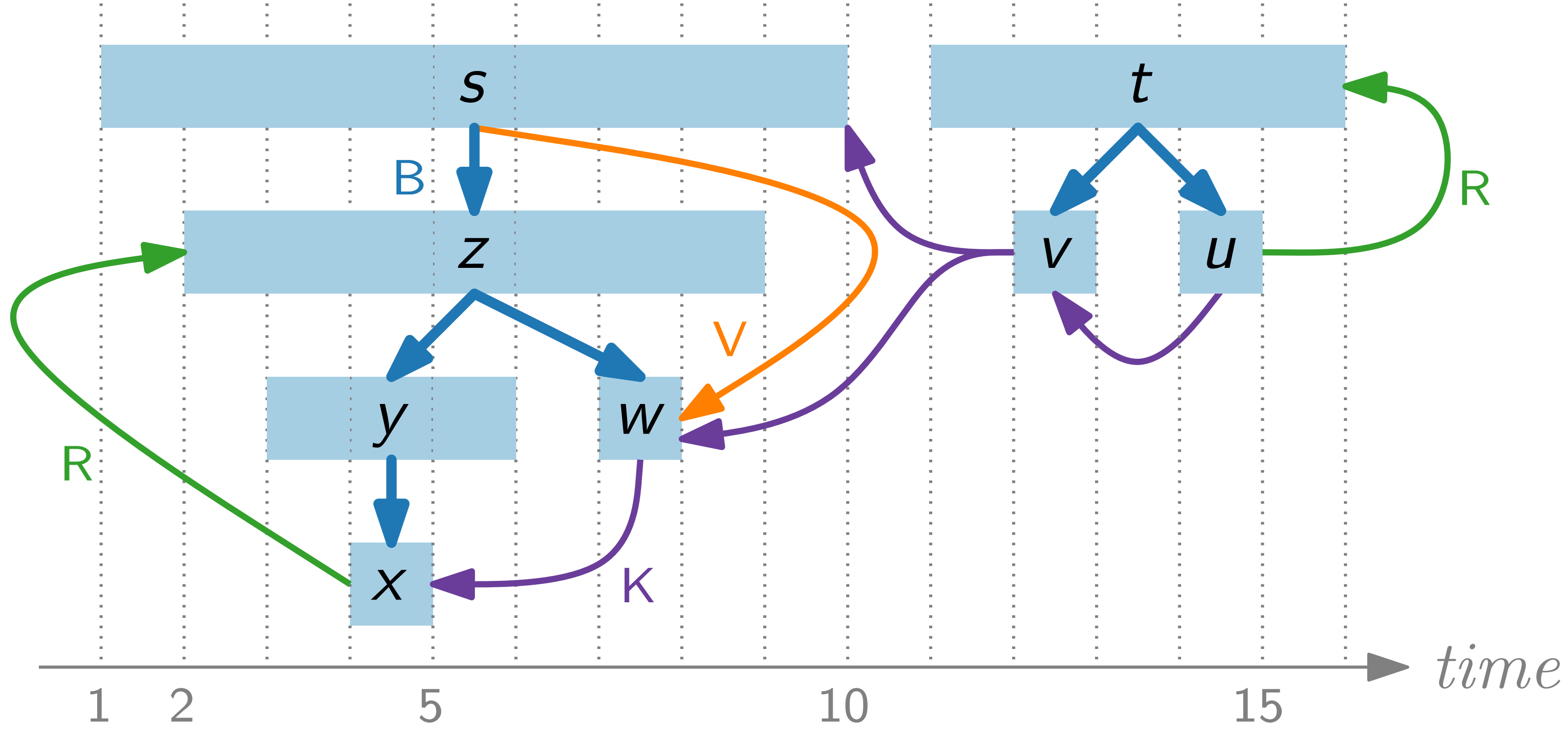
d.h. für jedes Paar $\{u, v\}$ von Knoten (mit $u \neq v$)

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.



Tiefensuche – Analyse

Satz. (Klammerntheorem)

d.h. für jedes Paar $\{u, v\}$ von Knoten (mit $u \neq v$)

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.

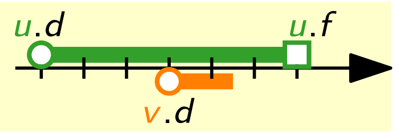
(ii) $[u.d, u.f] \subset [v.d, v.f]$ und Baumkanten enthalten v-u-Weg.

(iii) Wie (ii), nur umgekehrt.

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

Beweis. Wir betrachten zwei Fälle.

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



A) $v.d < u.f$, d.h. v wurde entdeckt, als u noch rot 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 blau, **bevor** DFS zu u zurückkehrt und u blau macht.

$\Rightarrow$ $[v.d, v.f] \subset [u.d, u.f]$, d.h. (iii) ✓

Tiefensuche – Analyse

Satz. (Klammerntheorem)

d.h. für jedes Paar $\{u, v\}$ von Knoten (mit $u \neq v$)

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.

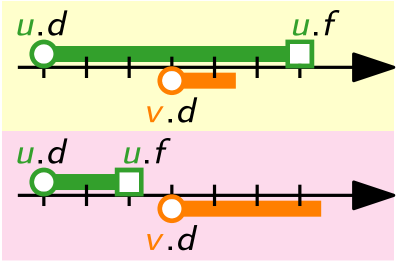
(iii) Wie (ii), nur umgekehrt.

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

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 rot war, d.h. keiner ist Nachfolger des anderen. $\Rightarrow$ (i)

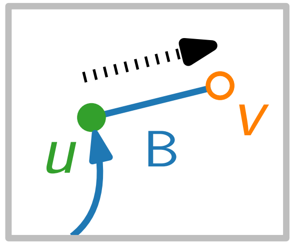
Tiefensuche in ungerichteten Graphen

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

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

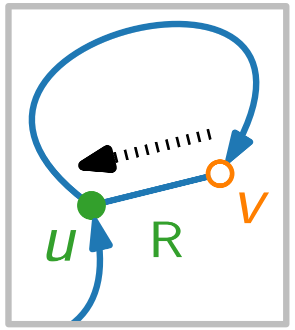
O.B.d.A. gilt $u.d < v.d$.

Dann entdeckt DFS v und färbt v blau,
bevor u blau 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.

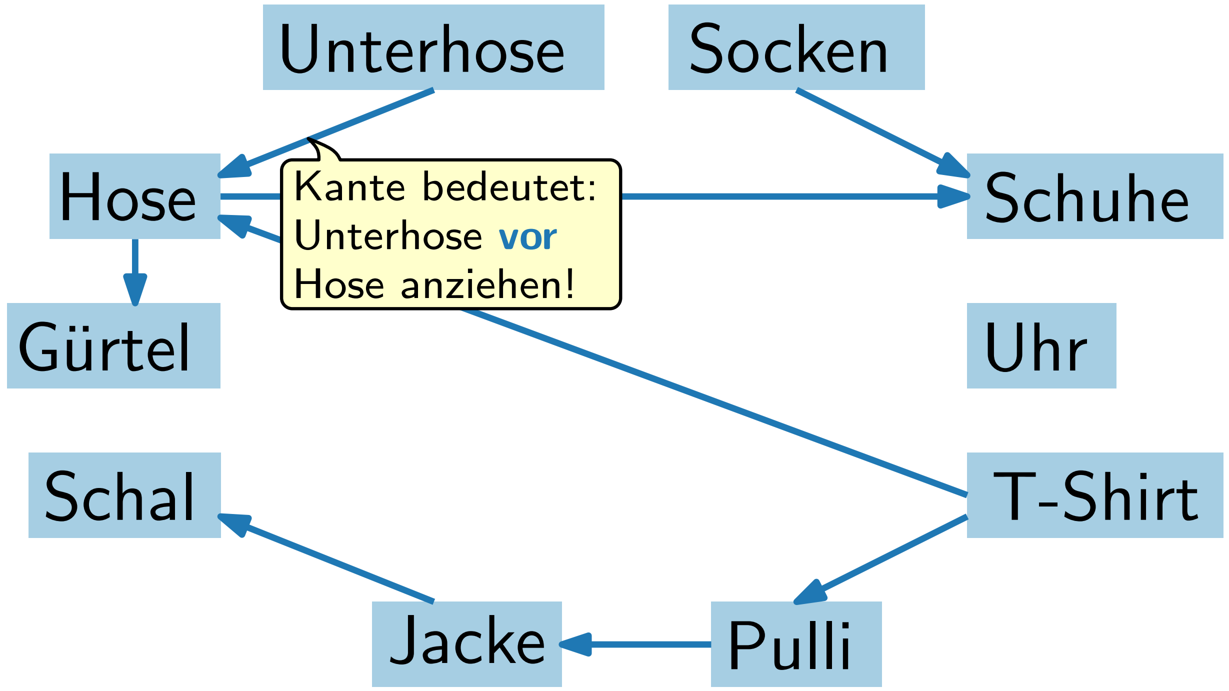
  Dann ist uv Baumkante.



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

$\square$

Ablaufplanung

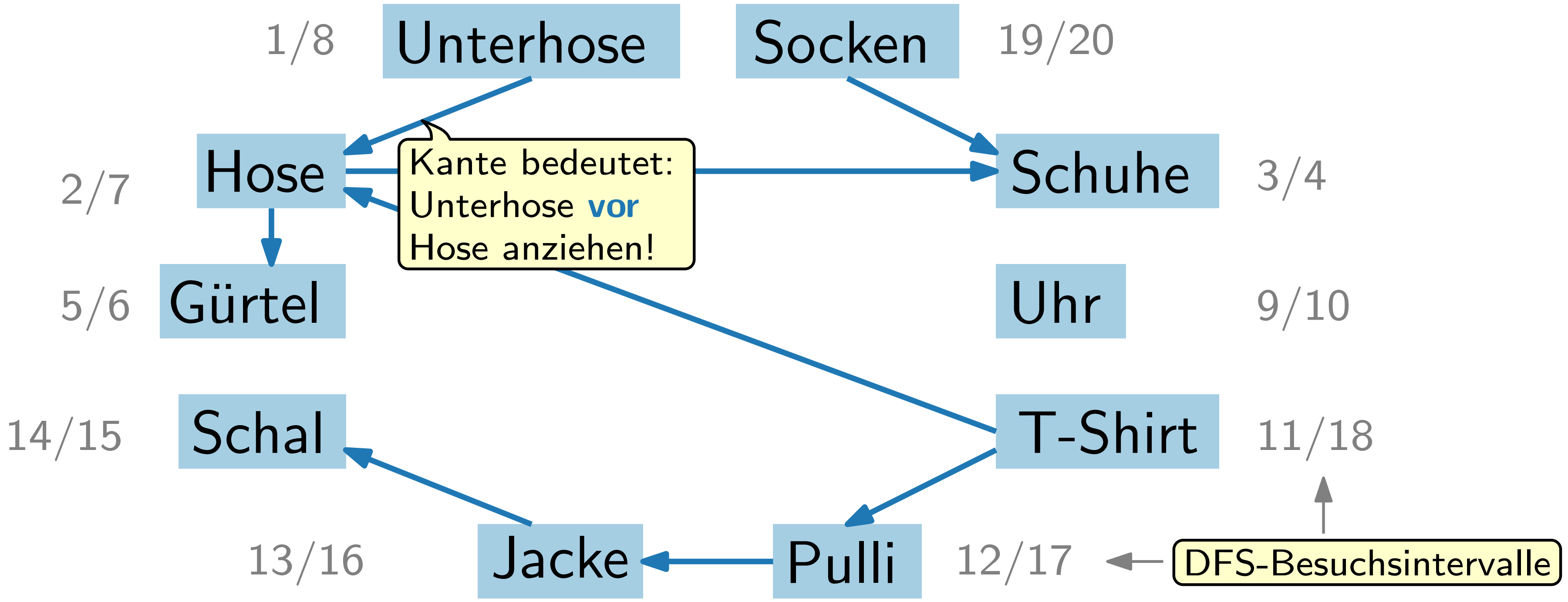


Aufgabe: Finde Ablaufplan –

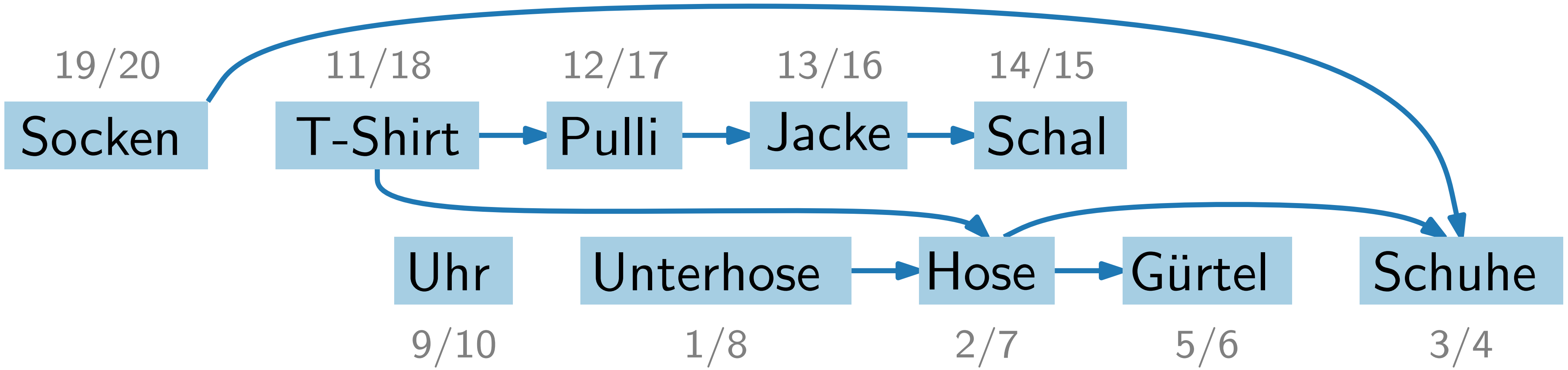
d.h. Reihenfolge der Knoten, so dass alle Einschränkungen erfüllt sind (z.B. T-Shirt vor Pulli).

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

Ablaufplanung



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



Topologisch sortieren

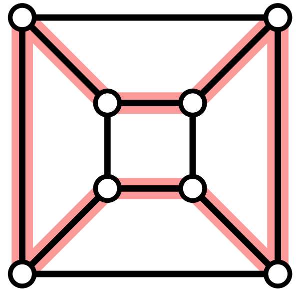
Topologische Sortierung: Lineare Ordnung der Knoten, so dass aus $(u, v) \in E$ folgt: u kommt vor v.

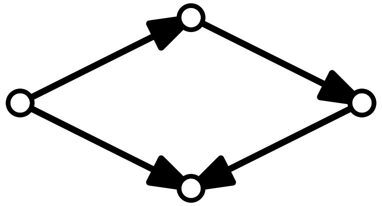
$\textsc{TopologicalSort}$(DirectedGraph G)
- L = **new** $\textsc{List}()$
- DFS(G) mit folgender Änderung:
  - Wenn ein Knoten blau gefärbt wird, häng ihn **vorne** an die Liste L an.
- **return** L

Laufzeit?
$\mathcal{O}(V + E)$

Korrekt?
Wann funktiont's?

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

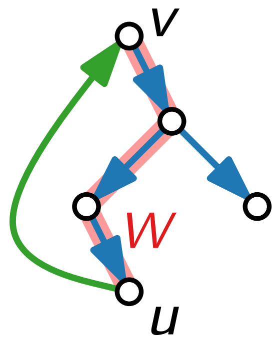




Kreisfrei $\Leftrightarrow$ keine R-Kanten

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

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



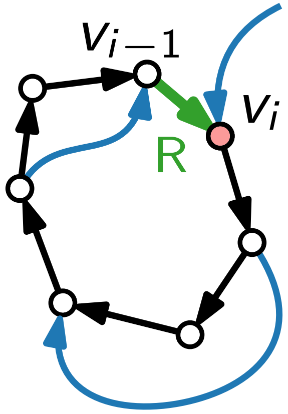
Angenommen DFS(G) liefert R-Kante (u, v).

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 rot ist.

$\Rightarrow$ (v_{i-1}, v_i) ist R-Kante. ⚡ $\square$

Korrektheit von TopologicalSort

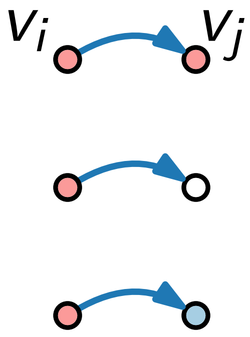
Satz. Sei G ein gerichteter kreisfreier Graph. Dann liefert TopologicalSort(G) eine topologische Sortierung von G.

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

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

Sei (v_i, v_j) Kante von G. Zu zeigen: $v_i.f > v_j.f$

Welche Farbe hat v_j, wenn DFS (v_i, v_j) überschreitet?



- v_j rot $\Rightarrow$ (v_i, v_j) ist R-Kante ⚡ Widerspruch zu Lemma: G kreisfrei!
- v_j weiß $\Rightarrow$ v_j Nachfolger von v_i $\Rightarrow$ $v_i.f > v_j.f$ ✓
- v_j blau $\Rightarrow$ $v_i.f$ noch nicht gesetzt, $v_j.f$ gesetzt
  $\Rightarrow$ $v_i.f > v_j.f$ ✓

□

Vergleich Durchlaufstrategien für Graphen

	Breitensuche	**Tiefensuche**
Laufzeit	$\mathcal{O}(V + E)$	$\mathcal{O}(V + E)$
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