CoNe Freiburg

IIF INSTITUT FÜR INFORMATIK FREIBURG

Graphentheorie
10 – Tiefensuche

Dr. Sven Köhler
Rechnernetze und Telematik
Technische Fakultät
Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

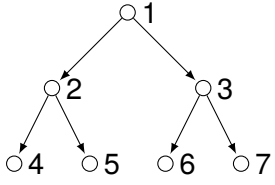
Tiefen- vs. Breitensuche

Ziel einer Suche:

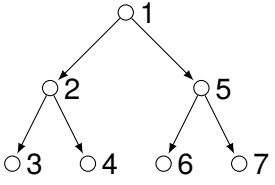
- Besuche alle erreichbaren Knoten

Strategien:

- Zuerst in die “Tiefe” gehen (DFS = Depth First Search)
- Zuerst in die “Breite” gehen (BFS = Breadth First Search)



Breitensuche



Tiefensuche

Tiefensuche – Vorbereitungen

DFS-Algorithmus markiert Knoten mit 3 Farben

- $\text{weiss} =$ unentdeckt
- $\text{grau} =$ entdeckt und “in Arbeit”
- $\text{schwarz} =$ abgearbeitet

und benutzt folgende Variablen

- $zeit =$ Zeitstempel $(0, 1, 2, 3, \ldots)$
- $d[v] =$ Discovery Time (wann wurde v grau gefärbt)
- $f[v] =$ Finishing Time (wann wurde v schwarz gefärbt)

Wir definieren Zeitinterval $I(v) = [d[v], f[v]]$.

DFS Algorithmus – Teil 1

Algorithmus 7.1 $\mathrm{DFS}(G)$

Eingabe: gerichteter Graph $G = (V, R)$

for each $v \in V$ **do**
    $farbe[v] := \text{weiss}$
    $\pi[v] := \text{nil}$

$R_\pi := \emptyset$
$zeit := 0$
for each $v \in V$ **do**
    if $farbe[v] = \text{weiss}$ **then**
        $\mathrm{DFS\text{-}VISIT}(v)$

return R_π

DFS Algorithmus – Teil 2

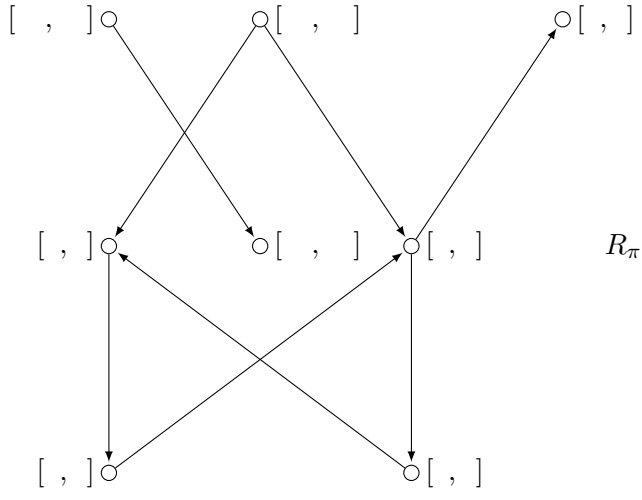
Algorithmus 7.2 DFS-VISIT(v)

$farbe[v] := \text{grau}$
$d[v] := zeit$
$zeit := zeit + 1$

for each $u \in N^+(v)$ **do** ▷ benutze $ADJ[v]$
  if $farbe[u] = \text{weiss}$ **then**
    $\pi[u] := v$ ▷ $\pi[u]$ speichert Vorgänger von u
    $R_\pi := R_\pi \cup \{r_{vu}\}$ ▷ r_{vu} ist die Kante von v nach u
    DFS-VISIT(u)

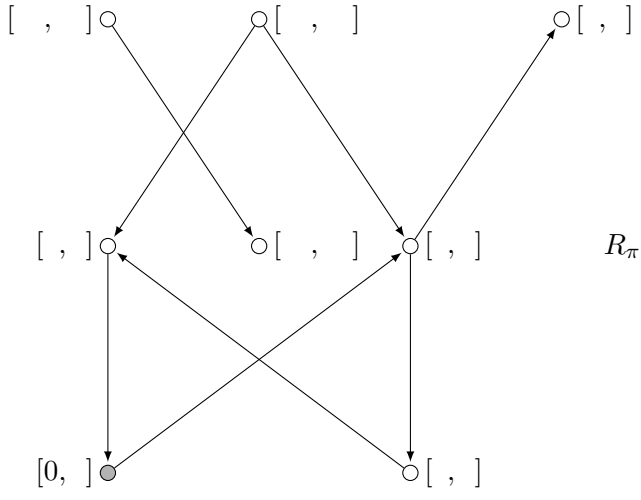
$farbe[v] := \text{schwarz}$
$f[v] := zeit$
$zeit := zeit + 1$

Tiefensuche – Beispiel



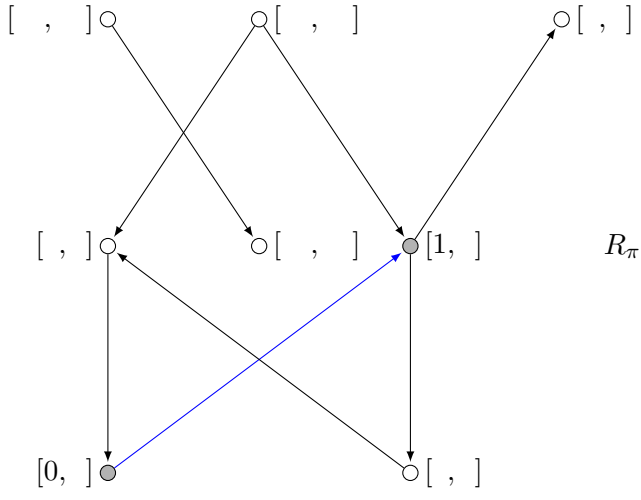
$R_\pi =$ blue edges

Tiefensuche – Beispiel



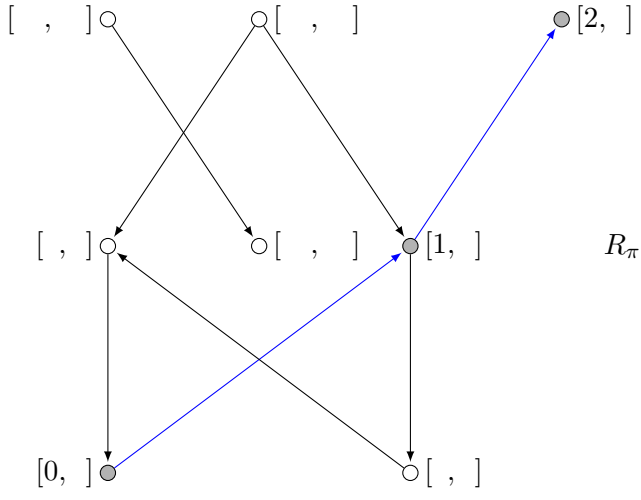
$R_\pi =$ blue edges

Tiefensuche – Beispiel



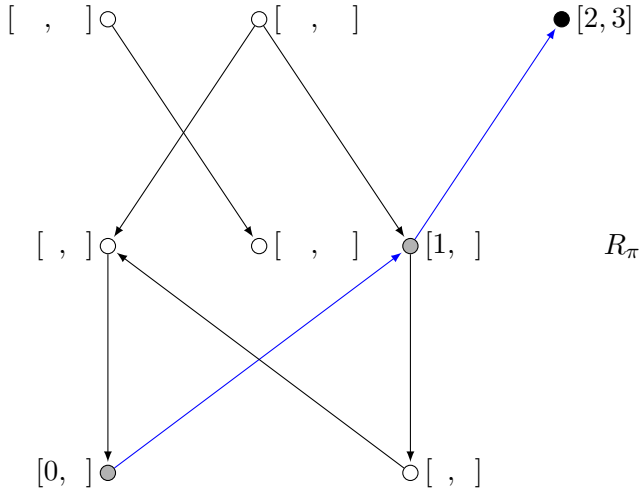
$R_\pi =$ blue edges

Tiefensuche – Beispiel



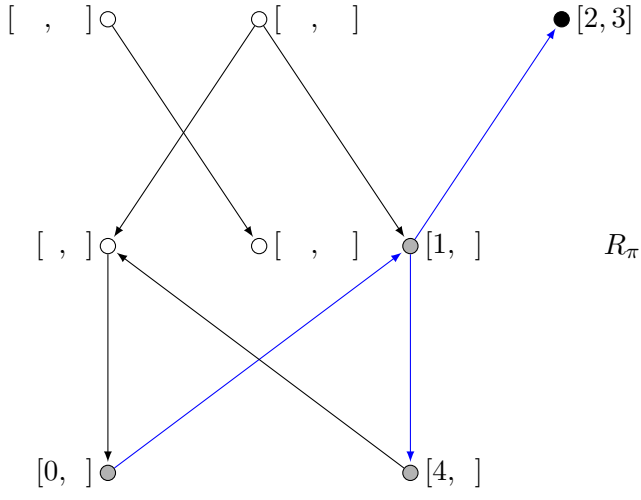
$R_\pi =$ blue edges

Tiefensuche – Beispiel



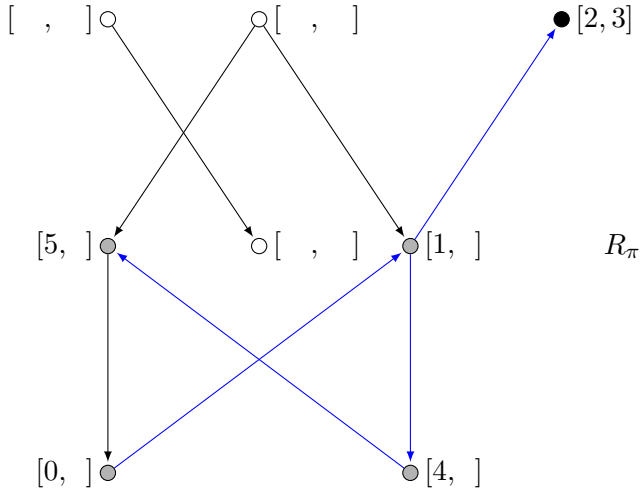
$R_\pi =$ blue edges

Tiefensuche – Beispiel



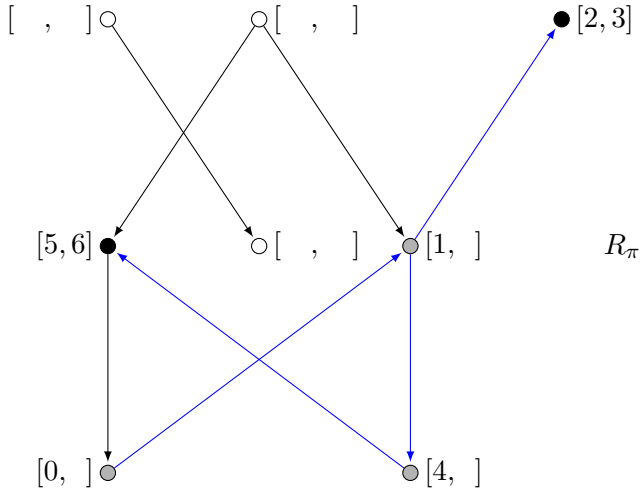
$R_\pi =$ blue edges

Tiefensuche – Beispiel



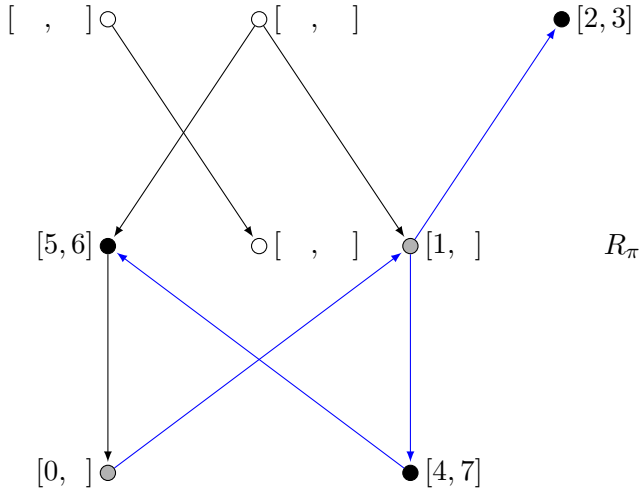
$R_\pi =$ blue edges

Tiefensuche – Beispiel



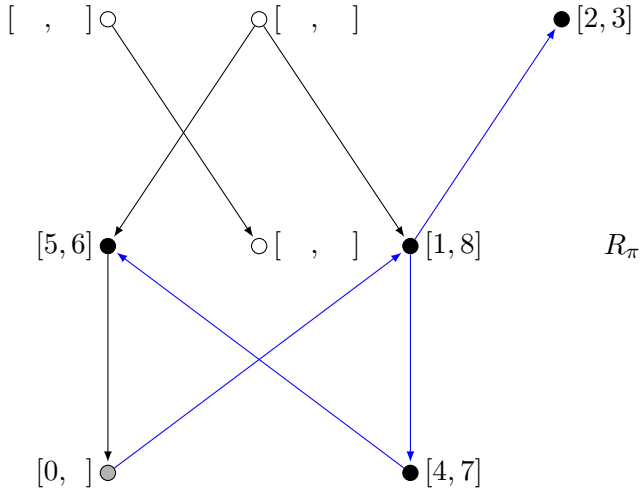
$R_\pi =$ blue edges

Tiefensuche – Beispiel



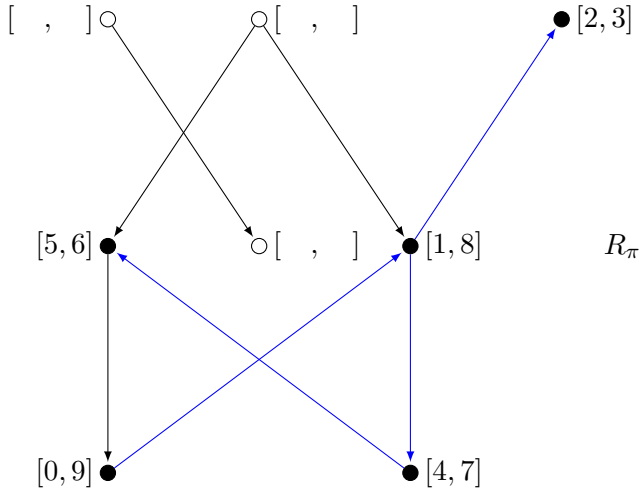
$R_\pi =$ blue edges

Tiefensuche – Beispiel



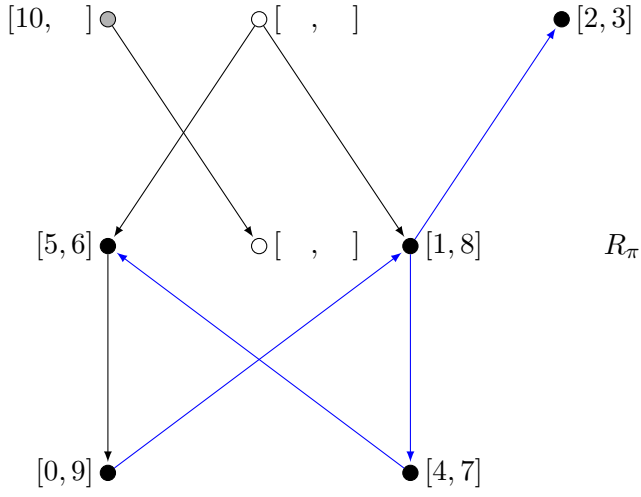
$R_\pi =$ blue edges

Tiefensuche – Beispiel



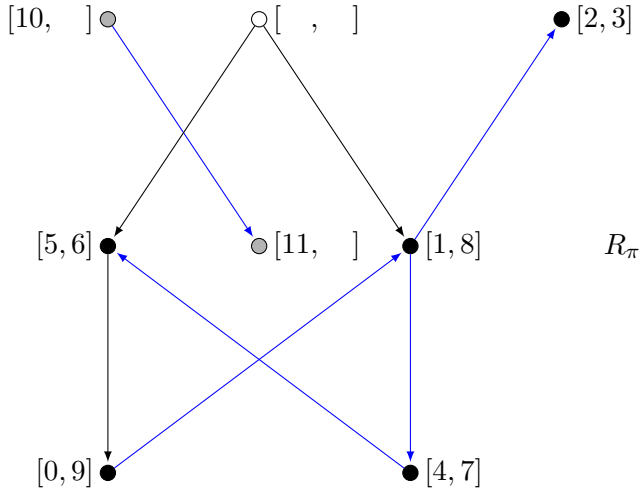
$R_\pi =$ blue edges

Tiefensuche – Beispiel



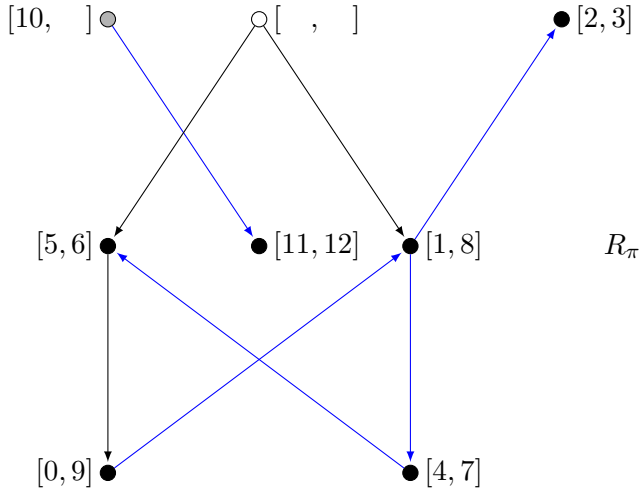
$R_\pi =$ blue edges

Tiefensuche – Beispiel



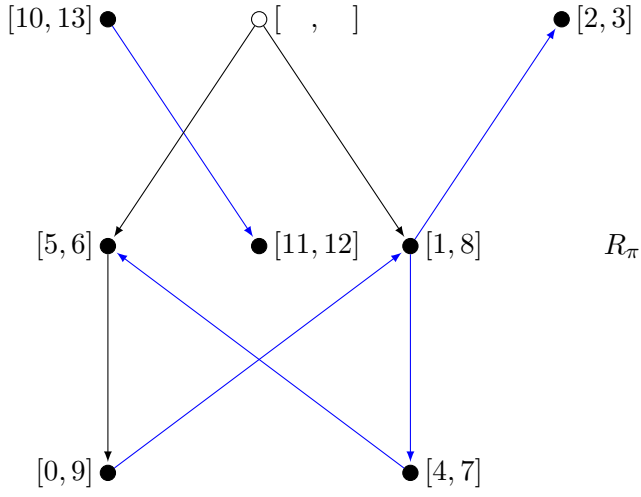
R_π = blue edges

Tiefensuche – Beispiel



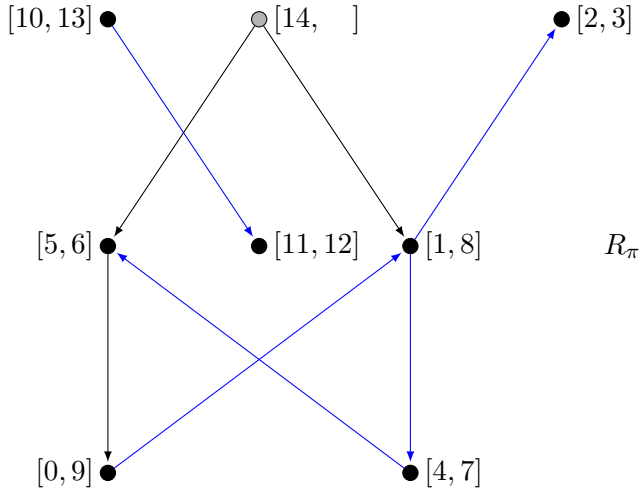
$R_\pi =$ blue edges

Tiefensuche – Beispiel



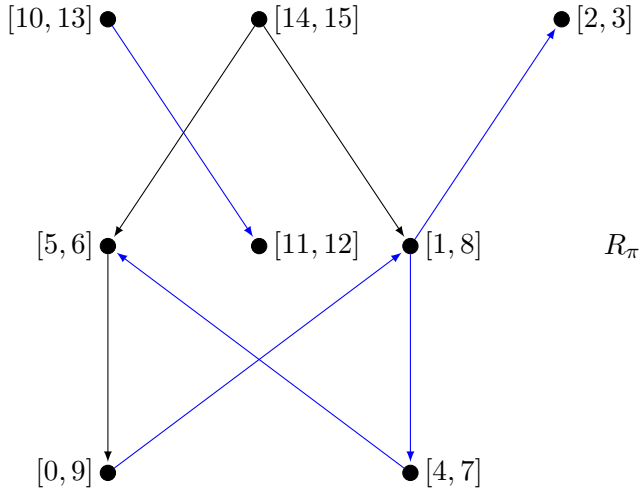
$R_\pi =$ blue edges

Tiefensuche – Beispiel



R_π = blue edges

Tiefensuche – Beispiel



$R_\pi =$ blue edges

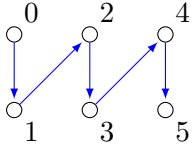
Tiefensuche

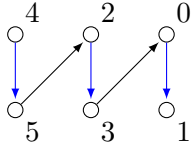
Satz

Sei G ein gerichteter Graph in Adjazenzlisten-Repräsentation.
Dann hat $\mathrm{DFS}(G)$ Laufzeit $\mathcal{O}(n+m)$.

Beobachtung

R_π ist nicht eindeutig bestimmt.





Tiefensuche

Satz 7.1

$G_\pi = (V, R_\pi)$ ist ein Wald.
Jede schwache Zusammenhangskomponente von G_π ist ein Wurzelbaum.

Beweis, Teil 1.

Es gilt $g^-(v) \leq 1$ für alle $v \in V$, da $\pi[v]$ höchstens einmal gesetzt wird.

Für alle $r_{vu} \in R_\pi$ gilt $d[v] < d[u]$. Daraus folgt: G_π ist kreisfrei.

Sei T eine schwache Zusammenhangskomponente von G_π.
Es existiert ein $s \in T$ mit $g^-(s) = 0$.
Zu zeigen: $E_{G_\pi[T]}(s) = T$

Tiefensuche

Beweis, Teil 2.

Sei T eine schwache Zusammenhangskomponente von G_π.
Sei $s \in T$ mit $g^-(s) = 0$. Zu zeigen: $E_{G_\pi[T]}(s) = T$

Betrachte den zu $G_\pi[T]$ zugeordneten Graphen und betrachte darin Weg P von s nach $u \in T$.

Zu zeigen: alle Kanten in P zeigen in Richtung u.

Induktion über Kanten von P:

- Kante 1 zeigt in Richtung u, da $g^-(s) = 0$
- da Kante i in Richtung u zeigt und $g^-(v) \leq 1$ für alle $v \in T$ zeigt Kante $i + 1$ zeigt ebenfalls in Richtung u

Damit $E_{G_\pi[T]}(s) = T$ und $G_\pi[T]$ ist s-Wurzelbaum. □

Tiefensuche – Intervallsatz

Satz 7.2 (Intervallsatz)

Seien $u, v \in V$ mit $d[u] < d[v]$. Dann gilt entweder

- $I(v) \cap I(u) = \emptyset$ oder
- $I(v) \subset I(u)$ und v ist Nachfahre von u in G_π.

Zur Erinnerung: $I(v) = [d[v], f[v]]$

Beweis.

Tiefensuche – Intervallsatz

Satz 7.2 (Intervallsatz)

Seien $u, v \in V$ mit $d[u] < d[v]$. Dann gilt entweder

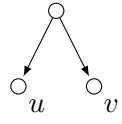
- $I(v) \cap I(u) = \emptyset$ oder
- $I(v) \subset I(u)$ und v ist Nachfahre von u in G_π.

Zur Erinnerung: $I(v) = [d[v], f[v]]$

Beweis.

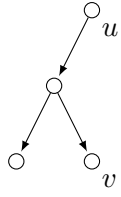
Fall 1: $d[v] > f[u]$

- Es gilt $d[u] < f[u] < d[v] < f[v]$



Fall 2: $d[v] < f[u]$

- v wird entdeckt wenn u grau ist
- $\delta^+(v)$ wird erforscht, bevor DFS-VISIT zu u zurückkehrt
- Damit $f[v] < f[u]$



□

Tiefensuche – Satz vom weißen Weg

Satz 7.4 (Satz vom weißen Weg)

Ein Knoten v ist genau dann ein Nachfahre von u in G_π, wenn es zum Zeitpunkt $d[u]$ einen Weg P von u nach v gibt, der neben u nur weiße Knoten berührt.

Beweis.

Tiefensuche – Satz vom weißen Weg

Satz 7.4 (Satz vom weißen Weg)

Ein Knoten v ist genau dann ein Nachfahre von u in G_π, wenn es zum Zeitpunkt $d[u]$ einen Weg P von u nach v gibt, der neben u nur weiße Knoten berührt.

Beweis.

"$\Rightarrow$": Betrachte Weg P von u nach v in G_π. Alle von P berührten Knoten werden (zeitlich) nach $d[u]$ erkundet. P ist weißer Weg zum Zeitpunkt $d[u]$.

Tiefensuche – Satz vom weißen Weg

Satz 7.4 (Satz vom weißen Weg)

Ein Knoten v ist genau dann ein Nachfahre von u in G_π, wenn es zum Zeitpunkt $d[u]$ einen Weg P von u nach v gibt, der neben u nur weiße Knoten berührt.

Beweis.

"$\Rightarrow$": Betrachte Weg P von u nach v in G_π. Alle von P berührten Knoten werden (zeitlich) nach $d[u]$ erkundet. P ist weißer Weg zum Zeitpunkt $d[u]$.

"$\Leftarrow$": O.B.d.A. wähle Weg P so, dass er Rekursion von $\mathrm{DFS\text{-}VISIT}(u)$ folgt. Induktion über die Länge von P zeigt: $\mathrm{DFS\text{-}VISIT}(v)$ wird aufgerufen. $\square$

Tiefensuche – Satz vom weißen Weg

Satz 7.4 (Satz vom weißen Weg)

Ein Knoten v ist genau dann ein Nachfahre von u in G_π, wenn es zum Zeitpunkt $d[u]$ einen Weg P von u nach v gibt, der neben u nur weiße Knoten berührt.

Beweis.

"$\Rightarrow$": Betrachte Weg P von u nach v in G_π. Alle von P berührten Knoten werden (zeitlich) nach $d[u]$ erkundet. P ist weißer Weg zum Zeitpunkt $d[u]$.

"$\Leftarrow$": O.B.d.A. wähle Weg P so, dass er Rekursion von $\mathrm{DFS\text{-}VISIT}(u)$ folgt. Induktion über die Länge von P zeigt: $\mathrm{DFS\text{-}VISIT}(v)$ wird aufgerufen. □

Korollar 7.5

Alle Knoten einer starken Zusammenhangskomponente von G liegen im gleichen Wurzelbaum von G_π.

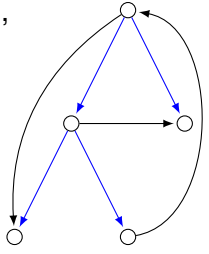
Tiefensuche – Kreise finden

Um Kreise zu finden erweitern wir den DFS-Algorithmus.

Definition

Eine Kante $r = (v, u)$ ist eine

- tree edge, wenn $r \in R_\pi$,
- back edge, wenn $r \notin R_\pi$ und v ist Nachfahre von u in G_π,
- forward edge, wenn $r \notin R_\pi$ und v ist Vorfahre von u in G_π,
- cross edge, sonst.



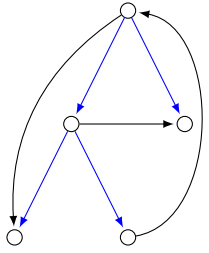
Tiefensuche – Kreise finden

Kanten können schon während der Ausführung des Algorithmus klassifiziert werden.

$\mathrm{DFS\text{-}VISIT}(v)$ ignoriert Kante (v, u) wenn u grau oder schwarz ist.
Genau dann wurde eine forward, back, oder cross edge gefunden!

Wenn $\mathrm{DFS\text{-}VISIT}(v)$ den Pfeil (v, u) untersucht, dann ist (v, u) eine

- tree edge, wenn u weiß ist,
- forward edge, wenn u schwarz ist und $d(v) < d(u)$ gilt,
- back edge, wenn u grau ist,
- cross edge, sonst.



Tiefensuche – Kreise finden

Satz 7.6
DFS findet genau dann back edges, wenn der Graph Kreise enthält.

Beweis.
"$\Rightarrow$":
Sei (u, v) eine back edge. Dann gibt es in G_π einen Weg P von v nach u.
Verlängern von P um (u, v) ergibt Kreis.

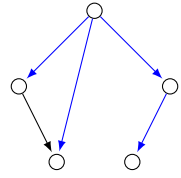
"$\Leftarrow$":
Sei C ein Kreis mit $s(C) = (v_0, v_1, v_2, \ldots, v_k = v_0)$.
O.B.d.A. sei v_1 der erste Knoten, der von $\mathrm{DFS\text{-}VISIT}$ besucht wird.
Nach dem Satz vom weißen Weg wird v_0 von $\mathrm{DFS\text{-}VISIT}(v_1)$ besucht.
Damit wird v_0 ein Nachfolger von v_1 und (v_0, v_1) eine back edge. □

Tiefensuche vs. Topologische Sortierung

Tiefensuche berechnet auch eine topologische Sortierung.
(nur wenn der Graph kreisfrei ist)

Algorithmus 7.3

$i := n$
Rufe $\mathrm{DFS}(G)$ auf
Wenn v schwarz wird: setze $\sigma(v) := i$ und $i := i - 1$
return σ



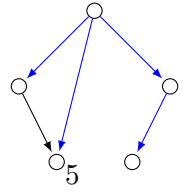
Satz 7.8
Alg. 7.3 liefert topologische Sortierung von kreisfreien Graphen in Zeit $\mathcal{O}(n + m)$.

Tiefensuche vs. Topologische Sortierung

Tiefensuche berechnet auch eine topologische Sortierung.
(nur wenn der Graph kreisfrei ist)

Algorithmus 7.3

$i := n$
Rufe $\mathrm{DFS}(G)$ auf
Wenn v schwarz wird: setze $\sigma(v) := i$ und $i := i - 1$
return σ



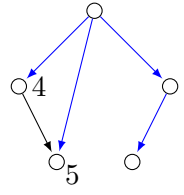
Satz 7.8
Alg. 7.3 liefert topologische Sortierung von kreisfreien Graphen in Zeit $\mathcal{O}(n + m)$.

Tiefensuche vs. Topologische Sortierung

Tiefensuche berechnet auch eine topologische Sortierung.
(nur wenn der Graph kreisfrei ist)

Algorithmus 7.3

$i := n$
Rufe $\mathrm{DFS}(G)$ auf
Wenn v schwarz wird: setze $\sigma(v) := i$ und $i := i - 1$
return σ



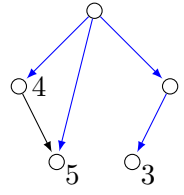
Satz 7.8
Alg. 7.3 liefert topologische Sortierung von kreisfreien Graphen in Zeit $\mathcal{O}(n + m)$.

Tiefensuche vs. Topologische Sortierung

Tiefensuche berechnet auch eine topologische Sortierung.
(nur wenn der Graph kreisfrei ist)

Algorithmus 7.3

$i := n$
Rufe $\mathrm{DFS}(G)$ auf
Wenn v schwarz wird: setze $\sigma(v) := i$ und $i := i - 1$
return σ



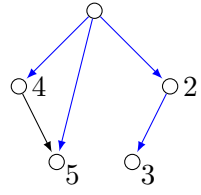
Satz 7.8
Alg. 7.3 liefert topologische Sortierung von kreisfreien Graphen in Zeit $\mathcal{O}(n + m)$.

Tiefensuche vs. Topologische Sortierung

Tiefensuche berechnet auch eine topologische Sortierung.
(nur wenn der Graph kreisfrei ist)

Algorithmus 7.3

$i := n$
Rufe $\mathrm{DFS}(G)$ auf
Wenn v schwarz wird: setze $\sigma(v) := i$ und $i := i - 1$
return σ



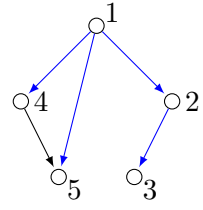
Satz 7.8
Alg. 7.3 liefert topologische Sortierung von kreisfreien Graphen in Zeit $\mathcal{O}(n + m)$.

Tiefensuche vs. Topologische Sortierung

Tiefensuche berechnet auch eine topologische Sortierung.
(nur wenn der Graph kreisfrei ist)

Algorithmus 7.3

$i := n$
Rufe $\mathrm{DFS}(G)$ auf
Wenn v schwarz wird: setze $\sigma(v) := i$ und $i := i - 1$
return σ



Satz 7.8
Alg. 7.3 liefert topologische Sortierung von kreisfreien Graphen in Zeit $\mathcal{O}(n + m)$.

Tiefensuche – Starke Zusammenhangskomponenten

Algorithmus 7.4 Berechnung der starken Zusammenhangskomponenten

Rufe $\mathrm{DFS}(G)$ auf ▷ berechnet $f[v]$
Sortiere Knoten nach $f[v]$
Berechne inversen Graphen G^{-1}
Rufe $\mathrm{DFS}(G^{-1})$ auf ▷ Knoten in absteigender Reihenfolge untersuchen

DFS-Bäume aus zweitem Durchlauf
sind starke Zusammenhangskomponenten von G

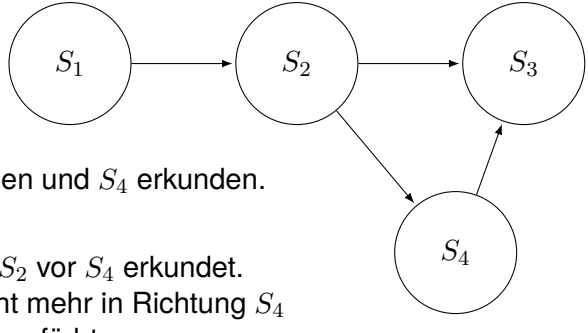
Beobachtung

Graph aus starken Zusammenhangskomponenten ist ein DAG.
(Gäbe es Kreise, so könnte man Zusammenhangskomponenten verschmelzen.)

Tiefensuche – Starke Zusammenhangskomponenten

Satz 7.9
Algorithmus 7.4 berechnet starke Zusammenhangskomponenten in Zeit $\mathcal{O}(n+m)$.

Beweisskizze.
Betrachte DAG aus starken Zusammengangskomponenten S_i.



$\mathrm{DFS}(G)$ wird S_2 irgendwann verlassen und S_4 erkunden.
$\Rightarrow \exists v \in S_2\ \forall u \in S_4 : f(v) > f(u)$

D.h. beim Lauf von $\mathrm{DFS}(G^{-1})$ wird S_2 vor S_4 erkundet.
In G^{-1} kann die Tiefensuche S_2 nicht mehr in Richtung S_4 verlassen. S_1 ist bereits schwarz eingefärbt.

Allgemein:
Gibt es eine Kante von S_i zu S_j, so wird S_i durch $\mathrm{DFS}(G^{-1})$ vor S_j erkundet. □