

ADS: Algorithmen und Datenstrukturen 2

Teil III

Peter F. Stadler & Konstantin Klemm

Bioinformatics Group, Dept. of Computer Science & Interdisciplinary Center for
Bioinformatics, **University of Leipzig**

20. April 2011

Traversierung

Durchlaufen eines Graphen, bei dem jeder vom gewählten Startknoten erreichbare Knoten (bzw. jede Kante) genau 1-mal aufgesucht wird. Jeweils nächster besuchter Knoten hat mindestens einen Nachbarn in der zuvor besuchten Knotenmenge.

Generische Lösungsmöglichkeit für Graphen $G = (V, E)$:

```
FOREACH v in V DO {markiere v als unbearbeitet};  
B={s}; // Menge besuchter Knoten, anfangs = Startknoten s  
markiere s als bearbeitet;  
WHILE es gibt unbearbeiteten Knoten v'  
  mit  $(v, v')$  in E und v in B  
  { B = B + {v'}; markiere v'; }
```

Realisierungen unterscheiden sich bezüglich Verwaltung der noch abzuarbeitenden Knotenmenge und Auswahl der jeweils nächsten Kante.

Breiten- und Tiefendurchlauf

Breitendurchlauf (Breadth First Search, BFS)

- ausgehend von Startknoten werden zunächst alle direkt erreichbaren Knoten bearbeitet
- danach die über mindestens zwei Kanten vom Startknoten erreichbaren Knoten, dann die über drei Kanten usw.
- es werden also erst die Nachbarn besucht, bevor zu den Söhnen gegangen wird.
- kann mit FIFO-Datenstruktur für noch zu bearbeitende Knoten realisiert werden.

Tiefendurchlauf (Depth First Search, DFS)

- ausgehend von Startknoten werden zunächst rekursiv alle Söhne (Nachfolger) bearbeitet; erst dann wird zu den Nachbarn gegangen
- kann mit Stack-Datenstruktur für noch zu bearbeitende Knoten realisiert werden
- Verallgemeinerung der Traversierung von Bäumen.

Breitensuche (BFS)

Bearbeite einen Knoten, der in n Schritten von u erreichbar ist, erst wenn alle Knoten abgearbeitet wurden, die in $n - 1$ Schritten erreichbar sind.

- gerichteter Graph $G = (V, E)$; Startknoten s ; Q sei FIFO-Warteschlange.
- zu jedem Knoten u werden der aktuelle Farbwert und der Vorgänger $p[u]$, von dem aus u erreicht wurde, gespeichert.
- p -Werte liefern nach Abarbeitung für zusammenhängende Graphen einen Spannbaum.

Breitensuche: Algorithmus

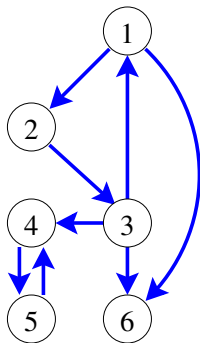
BFS(G, s)

```
FOREACH  $v$  in  $V$  DO {farbe( $v$ )=weiss;  $p[v]$ =null; }
farbe[ $s$ ]=grau; INIT( $Q$ );  $Q$ =ENQUEUE( $Q, s$ );
WHILE NOT (EMPTY( $Q$ )) DO
{
   $v$ =FRONT( $Q$ );
  FOREACH  $u$  in succ( $v$ ) DO
  {
    If farbe[ $u$ ]=weiss THEN
    { farbe[ $u$ ]=grau;  $p[u]$ = $v$ ;  $Q$ =ENQUEUE( $Q, u$ );}
  }
  DEQUEUE( $Q$ ); farbe[ $v$ ]=schwarz;
}
```

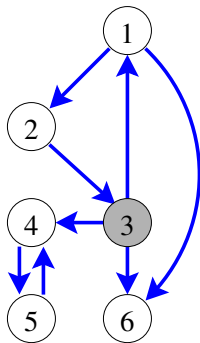
Farben:

weiss=unbearbeitet, grau=in Bearbeitung, schwarz=bearbeitet

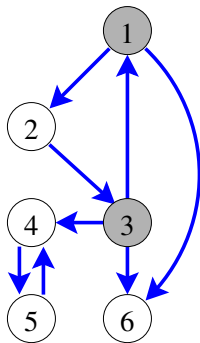
Breitensuche: Beispiel

Startknoten $s = 3$  $Q = []$

Breitensuche: Beispiel

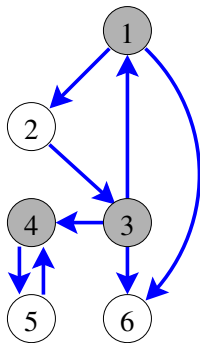
Startknoten $s = 3$  $Q = [3]$

Breitensuche: Beispiel

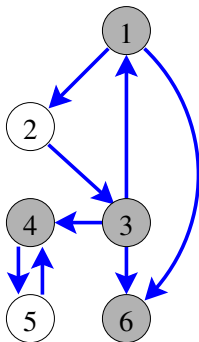
Startknoten $s = 3$ 

$$Q = [3, 1]$$

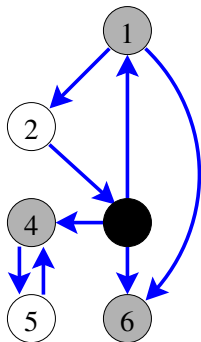
Breitensuche: Beispiel

Startknoten $s = 3$  $Q = [3, 1, 4]$

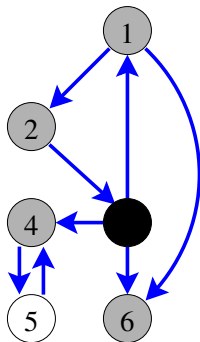
Breitensuche: Beispiel

Startknoten $s = 3$  $Q = [3, 1, 4, 6]$

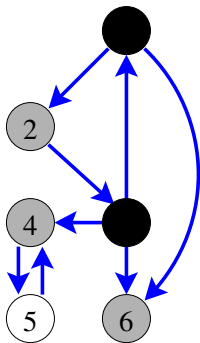
Breitensuche: Beispiel

Startknoten $s = 3$  $Q = [1, 4, 6]$

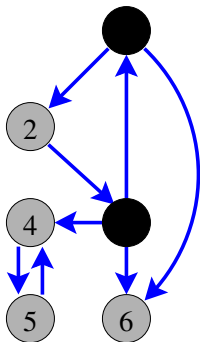
Breitensuche: Beispiel

Startknoten $s = 3$  $Q = [1, 4, 6, 2]$

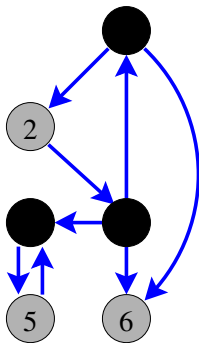
Breitensuche: Beispiel

Startknoten $s = 3$  $Q = [4, 6, 2]$

Breitensuche: Beispiel

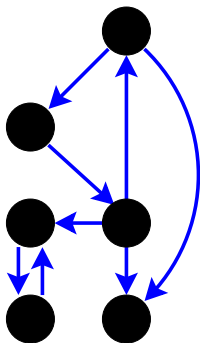
Startknoten $s = 3$  $Q = [4, 6, 2, 5]$

Breitensuche: Beispiel

Startknoten $s = 3$ 

$$Q = [6, 2, 5]$$

Breitensuche: Beispiel

Startknoten $s = 3$ 

$$Q = []$$

Tiefensuche (DFS)

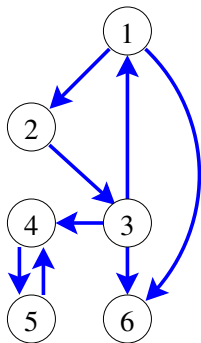
- Bearbeite einen Knoten v erst dann, wenn alle seine Söhne bearbeitet sind (außer wenn ein Sohn auf dem Weg zu v liegt)
- gerichteter Graph $G = (V, E)$;
- zu jedem Knoten v werden gespeichert: der aktuelle Farbwert $farbe[v]$, die Zeitpunkte $in[v]$ und $out[v]$, zu denen der Knoten im Rahmen der Tiefensuche erreicht bzw. verlassen wurde und der Vorgänger $p[v]$, von dem aus v erreicht wurde
- die in - bzw. out -Zeitpunkte ergeben eine Reihenfolge der Knoten analog zur Vor- bzw. Nachordnung bei Bäumen.

Tiefensuche: Algorithmus

```
DFS(G)
FOR EACH v in V do { farbe[v]=weiss; p[v]=null; }
zeit=0
for each v in V do {if farbe[v]=weiss then DFS-visit[v]}

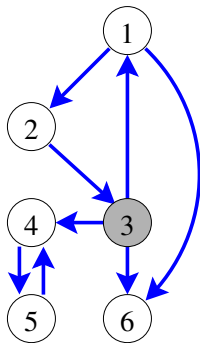
DFS-visit(G,v) // rekursive Methode zur Tiefensuche
farbe[v]=grau; zeit=zeit+1; in[v]=zeit;
FOR EACH u in succ(v) DO
{ IF farbe[u]=weiss THEN { p[u]=v; DFS-visit[u];}}
farbe[v]=schwarz; zeit=zeit+1; out[v]=zeit;
```

Tiefensuche: Beispiel

Startknoten $s = 3$ 

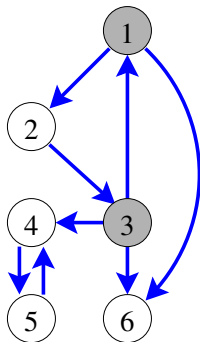
v	$\text{in}[v]$	$\text{out}[v]$
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

Tiefensuche: Beispiel

Startknoten $s = 3$ 

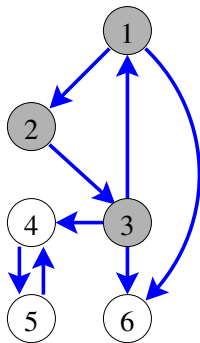
v	$\text{in}[v]$	$\text{out}[v]$
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

Tiefensuche: Beispiel

Startknoten $s = 3$ 

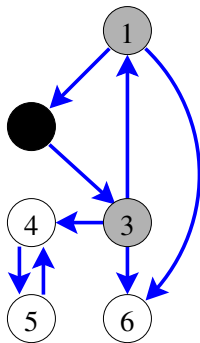
v	$\text{in}[v]$	$\text{out}[v]$
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

Tiefensuche: Beispiel

Startknoten $s = 3$ 

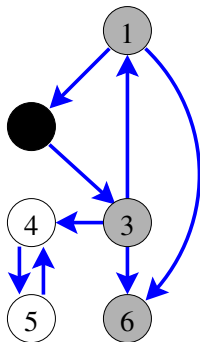
v	$\text{in}[v]$	$\text{out}[v]$
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

Tiefensuche: Beispiel

Startknoten $s = 3$ 

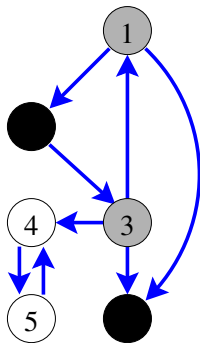
v	$\text{in}[v]$	$\text{out}[v]$
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

Tiefensuche: Beispiel

Startknoten $s = 3$ 

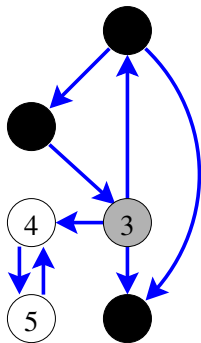
v	$\text{in}[v]$	$\text{out}[v]$
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

Tiefensuche: Beispiel

Startknoten $s = 3$ 

v	$\text{in}[v]$	$\text{out}[v]$
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

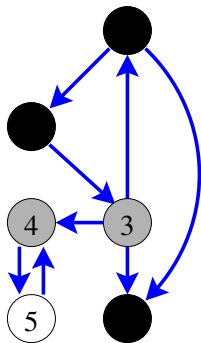
Tiefensuche: Beispiel

Startknoten $s = 3$ 

v	$in[v]$	$out[v]$
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

Tiefensuche: Beispiel

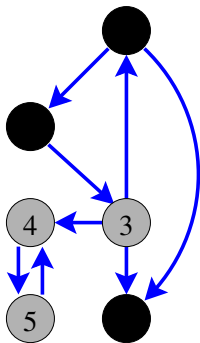
Startknoten $s = 3$



v	$in[v]$	$out[v]$
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

Tiefensuche: Beispiel

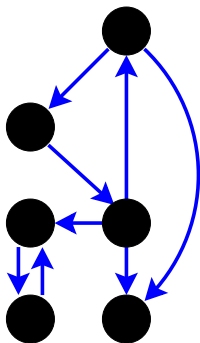
Startknoten $s = 3$



v	$\text{in}[v]$	$\text{out}[v]$
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

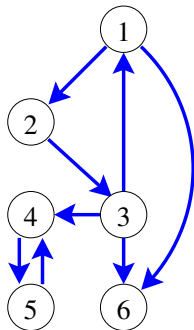
Tiefensuche: Beispiel

Startknoten $s = 3$

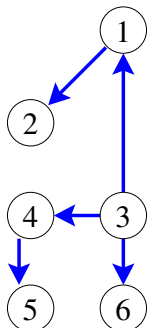


v	in[v]	out[v]
1	2	7
2	3	4
3	1	12
4	8	11
5	9	10
6	5	6

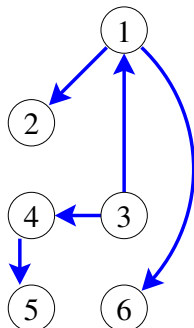
Bäume aus Breiten- und Tiefensuche



Graph G



BFS-Baum



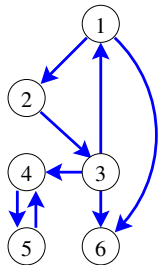
DFS-Baum

Starke Zusammenhangskomponenten

Ein gerichteter Graph $G = (V, E)$ heißt *stark zusammenhängend*, wenn für alle $u, v \in V$ gilt:
Es gibt einen Pfad von u nach v .

Eine *starke Zusammenhangskomponente* von G ist ein maximaler stark zusammenhängender Teilgraph von G .

Jeder Knoten eines Graphen ist in genau einer starken Zusammenhangskomponente enthalten. Wieso?

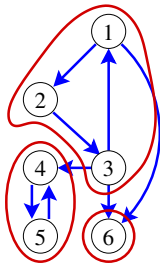


Starke Zusammenhangskomponenten

Ein gerichteter Graph $G = (V, E)$ heißt *stark zusammenhängend*, wenn für alle $u, v \in V$ gilt:
Es gibt einen Pfad von u nach v .

Eine *starke Zusammenhangskomponente* von G ist ein maximaler stark zusammenhängender Teilgraph von G .

Jeder Knoten eines Graphen ist in genau einer starken Zusammenhangskomponente enthalten. Wieso?



gegenseitige Erreichbarkeit = Äquivalenzrelation

Sei $G = (V, E)$ ein gerichteter Graph. Sind Knoten $u, v \in V$ gegenseitig erreichbar, schreiben wir $u \sim v$. Die so definierte Relation \sim auf V ist eine Äquivalenzrelation also

- symmetrisch
- transitiv und
- reflexiv

Die Knotenmengen der starken Zusammenhangskomponenten von G sind die Äquivalenzklassen von \sim .

Starke Zsh-Komponenten durch Tiefensuche

- Führe Tiefensuche (DFS) auf G aus.
- Für jeden Knoten v berechne dabei $l[v] =$ “frühester” Knoten, der von v erreichbar ist in der durch $\text{in}[]$ gegebenen Reihenfolge.
- Wenn alle Kindsknoten von v abgearbeitet sind und $l[v] = \text{in}[v]$, ist v die “Wurzel” einer starken Zusammenhangskomponente. Deren Knoten werden dann gleich ausgegeben und nicht mehr weiter betrachtet (denn jeder Knoten ist in nur einer Komponente).

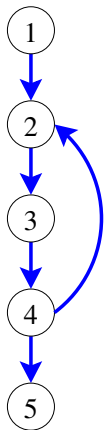
Algorithmus von Tarjan (1972)

```
Tarjan-visit(G,v)
farbe[v]=grau; zeit=zeit+1; in[v]=zeit; l[v]=zeit;
PUSH(S,v)
FOR EACH u in succ(v) DO {
  IF farbe[u]=weiss THEN
  { Tarjan-visit[u]; l[v]=min(l[v],l[u]) ;}
  ELSEIF u in S THEN l[v]=min(l[v],in[u]);
}

IF (l[v]=in[v]) {
  Ausgabe("starke ZshK:");
  DO { u=TOP(S); Ausgabe(u); POP(S); }
  UNTIL u=v;
}

farbe[v]=schwarz; zeit=zeit+1;
```

Tarjan: Beispiel



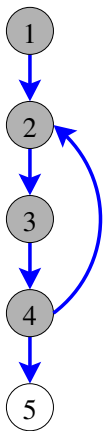
v	in[v]	l[v]
1	1	1
2	2	2
3	3	3
4	4	2
5	5	

Zshk: 5

Zshk: 2,3,4

Zshk: 1

Tarjan: Beispiel



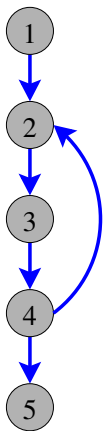
v	in[v]	l[v]
1	1	1
2	2	2
3	3	3
4	4	2
5	5	

Zshk: 5

Zshk: 2,3,4

Zshk: 1

Tarjan: Beispiel



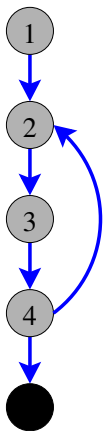
v	in[v]	l[v]
1	1	1
2	2	2
3	3	3
4	4	2
5	5	5

Zshk: 5

Zshk: 2,3,4

Zshk: 1

Tarjan: Beispiel



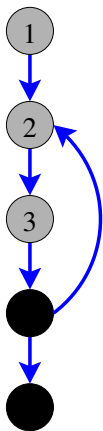
v	in[v]	l[v]
1	1	1
2	2	2
3	3	3
4	4	2
5	5	5

Zshk: 5

Zshk: 2,3,4

Zshk: 1

Tarjan: Beispiel



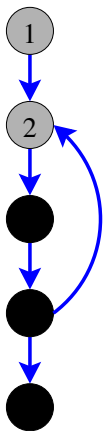
v	in[v]	l[v]
1	1	1
2	2	2
3	3	3
4	4	2
5	5	5

Zshk: 5

Zshk: 2,3,4

Zshk: 1

Tarjan: Beispiel



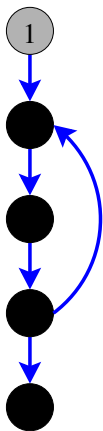
v	in[v]	l[v]
1	1	1
2	2	2
3	3	2
4	4	2
5	5	5

Zshk: 5

Zshk: 2,3,4

Zshk: 1

Tarjan: Beispiel



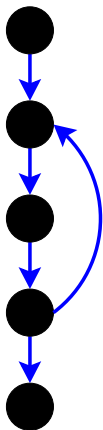
v	in[v]	l[v]
1	1	1
2	2	2
3	3	2
4	4	2
5	5	5

Zshk: 5

Zshk: 2,3,4

Zshk: 1

Tarjan: Beispiel



v	in[v]	l[v]
1	1	1
2	2	2
3	3	2
4	4	2
5	5	5

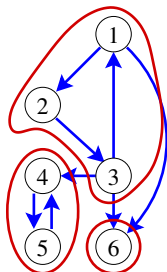
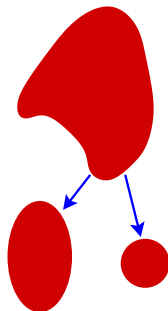
Zshk: 5

Zshk: 2,3,4

Zshk: 1

Komponentengraph G^*

Fasse alle Knoten jeweils einer starken Zusammenhangskomponente zu einem einzigen Knoten zusammen.
 Kante von Komponente A nach Komponente B , wenn es $u \in A$ und $v \in B$ gibt, so daß (u, v) eine Kante in G ist.

Graph G Komponentengraph G^*

Erlaubt z.B. schnellere Berechnung der transitiven Hülle von G .