

Kap. 6.3: Traversieren von Graphen

Kap. 6.4: Elementare Graphalgorithmen

Professor Dr. Petra Mutzel

Lehrstuhl für Algorithm Engineering, LS11

18. VO

8. Juni 2006

Motivation

Heute braucht es keine Motivation,
denn:

Spielereien

DFS

mit

Überblick

- Traversieren von Graphen:
 - Breitensuche (BFS) (Wdhlg.)
 - Tiefensuche (DFS)

- Elementare Graphalgorithmen:
 - Zusammenhangskomponenten
 - Kreise in Graphen

Kap. 6.3 Traversieren von Graphen

Traversieren: systematisches Durchwandern von Graphen

HIER: ungerichtete Graphen

Def.: Der **graphentheoretische Abstand** zweier Knoten u, v eines ungerichteten Graphen G ist die Länge des kürzesten Weges von u nach v , falls ein solcher existiert, sonst ∞ .

Achtung: hierbei werden keine vorgegebenen Kantenlängen bzw. Kantengewichte berücksichtigt. ⁴

Kap. 6.3.1 Breitensuche (BFS)

engl.: Breadth-first-search, BFS

Idee: besuche die Knoten nach aufsteigendem graphentheoretischen Abstand zu einem vorher festgelegten Startknoten.

1. Fall: Wir sehen v zum ersten Mal (von u aus): dann muss $\text{dist}(v)$ um genau 1 größer sein als $\text{dist}(u)$. Wir können v besuchen, nachdem wir alle bisher gesehenen Knoten besucht haben.
2. Fall: wir haben v schon gesehen \rightarrow nichts zu tun

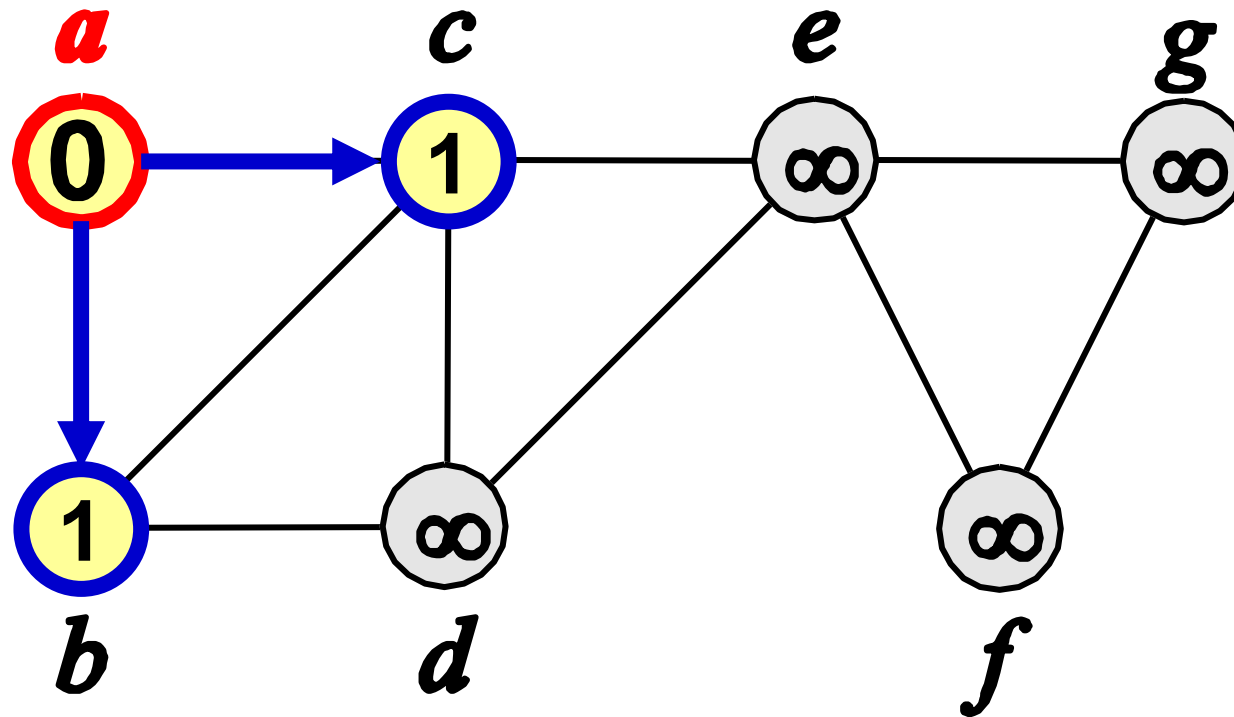
Breitensuche (BFS)

Methode:

- (1) Starte am Knoten $u:=s$. Sei $Q:=\emptyset$ eine Queue.
- (2) Für alle Knoten $v \in N(u)$ // erforsche Knoten u
- (3) Falls wir v zum ersten Mal sehen:
- (4) markiere v als „gesehen“
- (5) $\text{dist}(v):=\text{dist}(u)+1$; merke Vorgänger;
- (6) hänge v hinten an Q an.
- (7) Sei u der nächste Knoten in Q . Gehe zu (2)

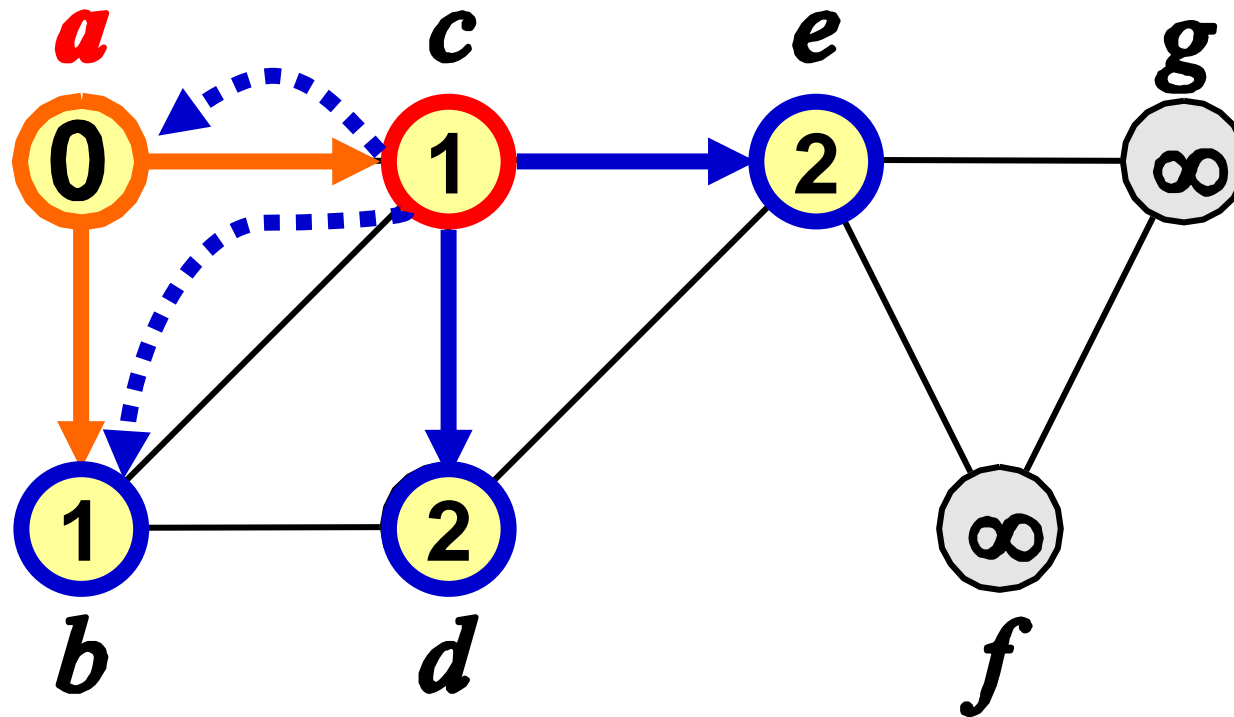
$\text{dist}(v)$ enthält den graphentheoretischen Abstand von u nach v ; $\pi(v)$ den Vorgänger des kürz. Weges

Beispiel für BFS



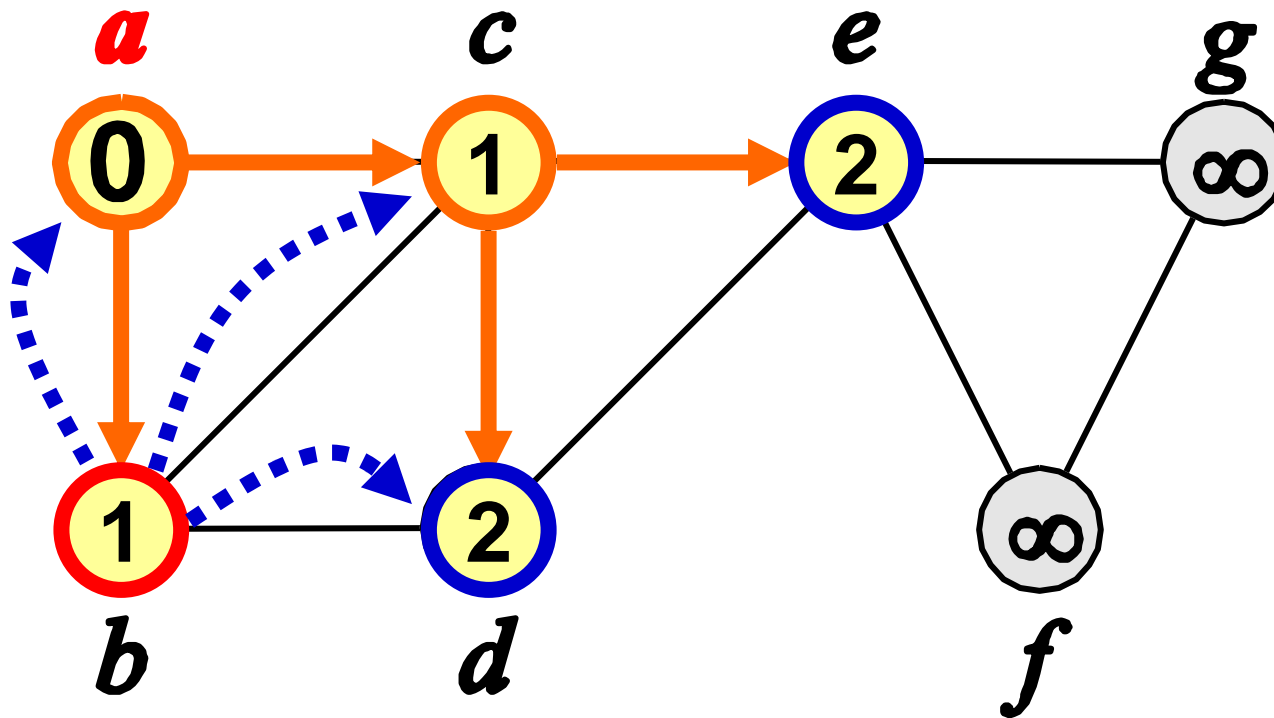
Q *a* *c* *b*

Beispiel für BFS

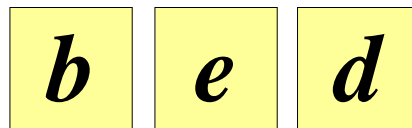


Q c b e d

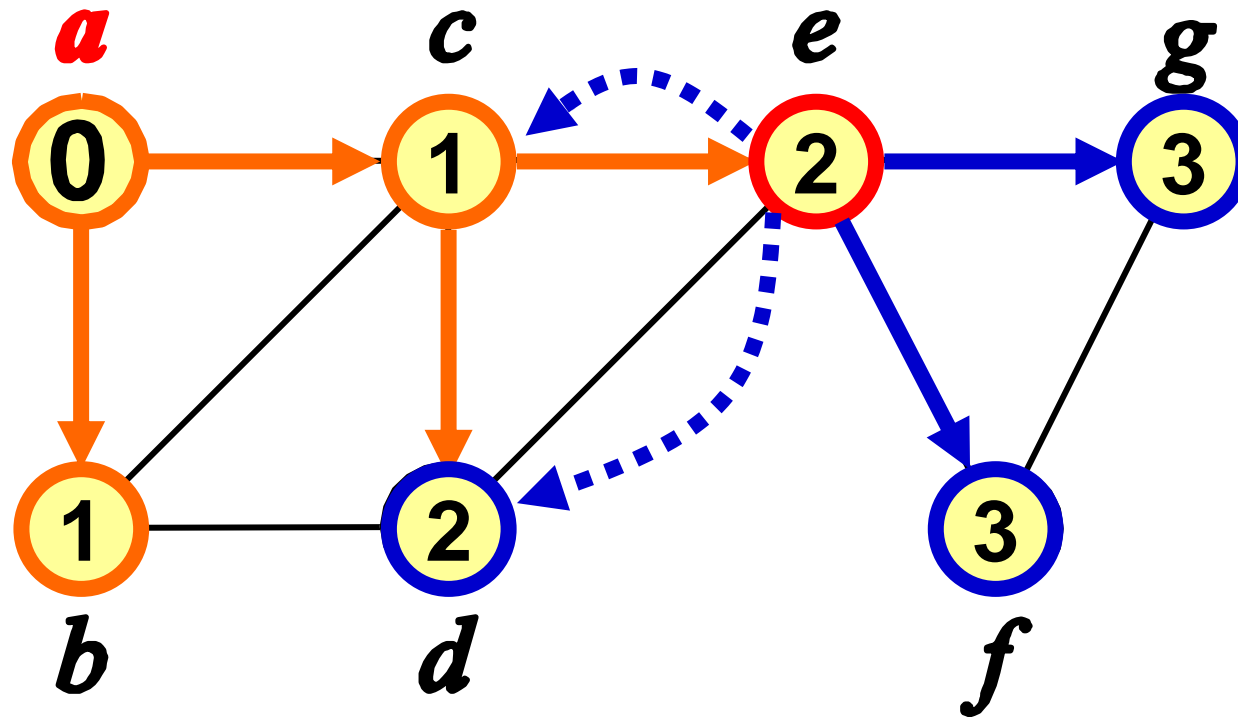
Beispiel für BFS



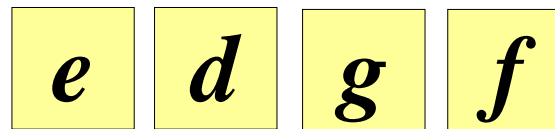
Q



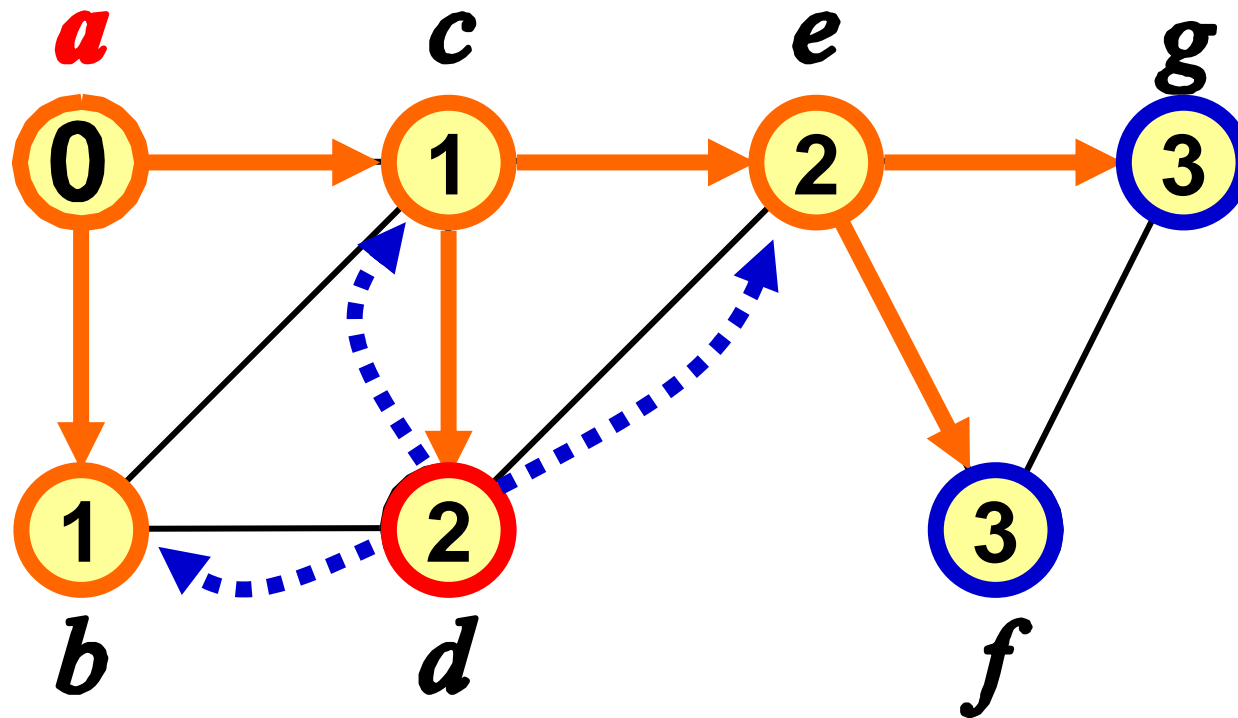
Beispiel für BFS



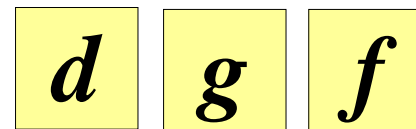
Q



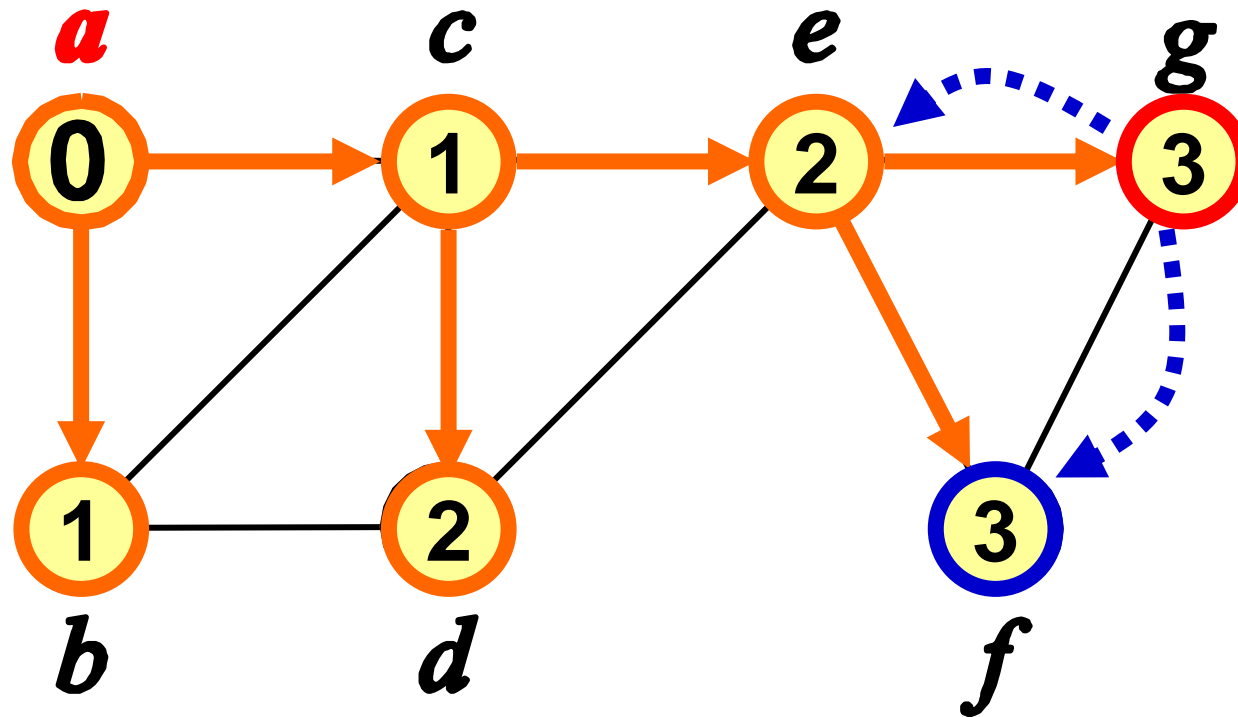
Beispiel für BFS



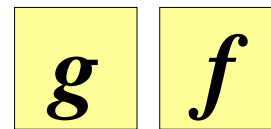
Q



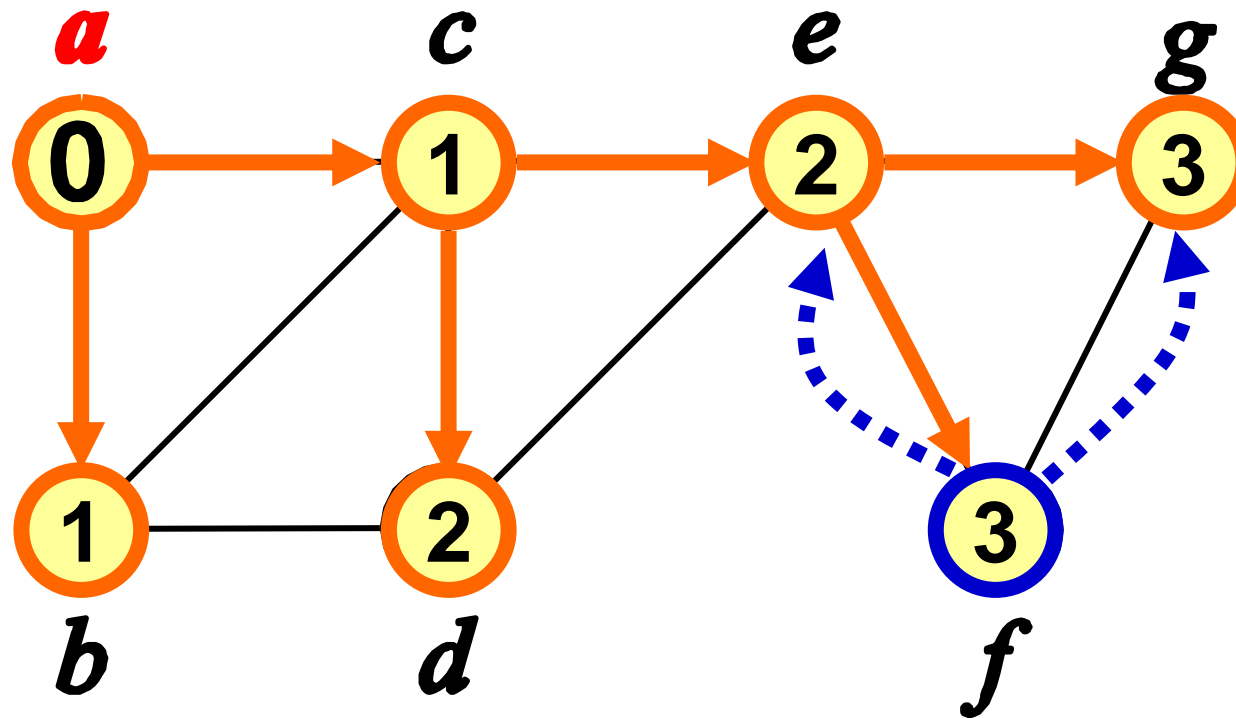
Beispiel für BFS



Q



Beispiel für BFS

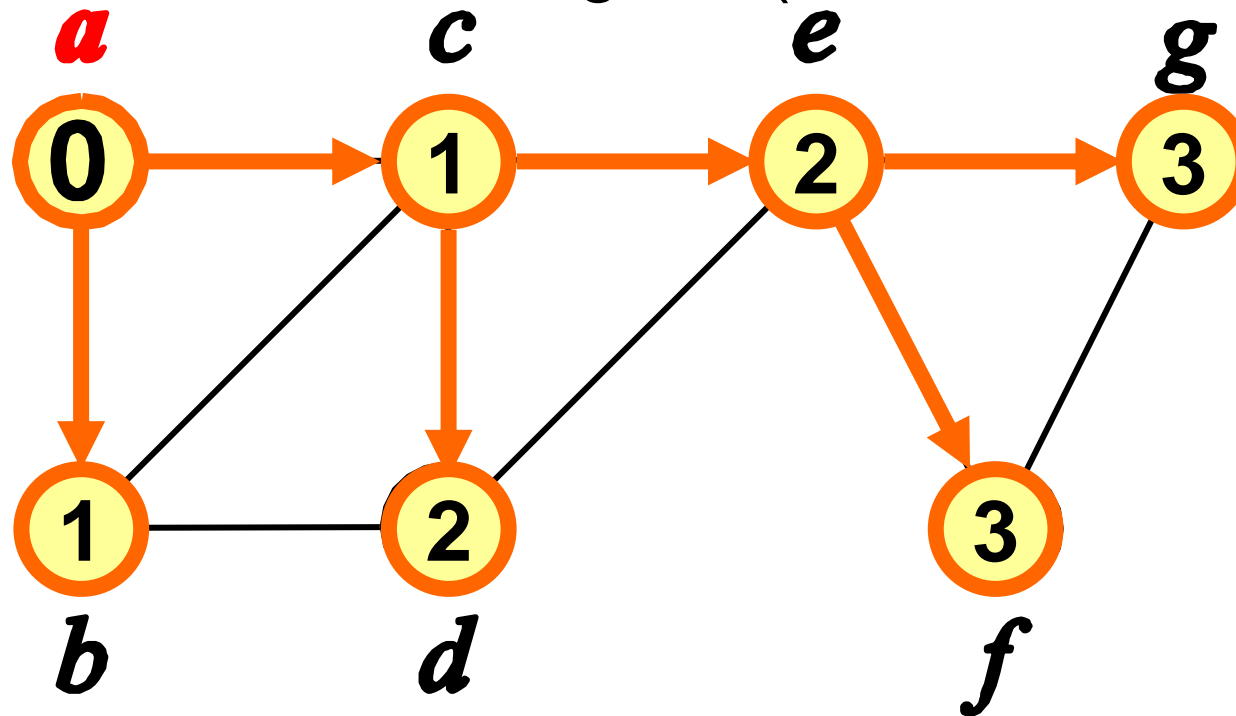


Q

f

Breitensuche entspricht einer level-order Traversierung in diesem BFS-Baum.

G zusammenhängend (sonst DFS-Wald)



Die orangenen Kanten (diejenigen Kanten (u,v) , die zum ersten Mal v besuchen) bilden einen Baum: den BFS-Baum

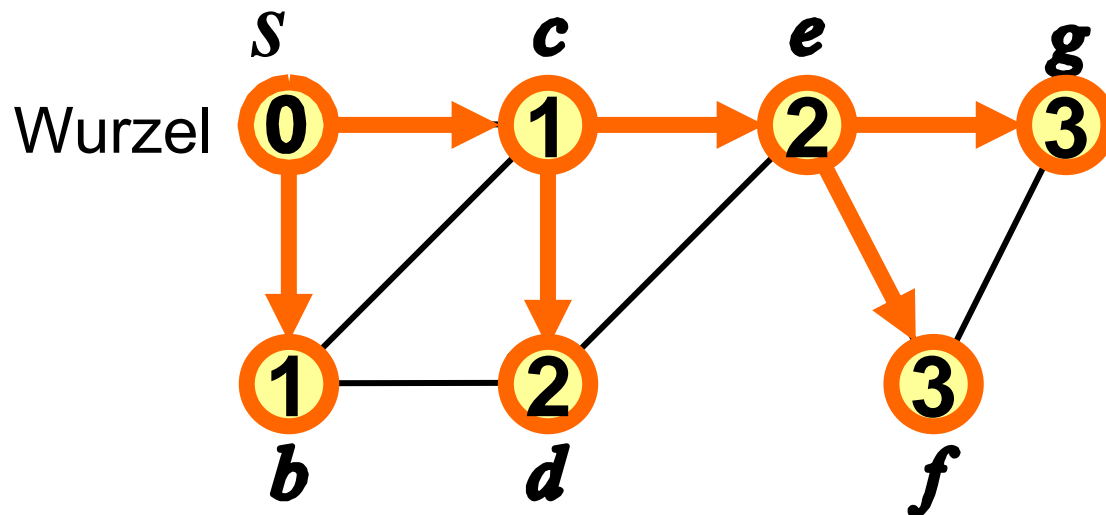
Algorithmus BFS(s)

Sei $G=(V,E)$ ungerichteter Graph, s,u,v : Knoten

```
(1) for all  $v \in V \setminus \{s\}$  do { marked[v]:=0; dist[v]:=∞ }
(2) Q.put(s); marked[s]:=1; dist[s]:=π[s]:=0;
(3) while not Q.ISEMPTY() do {
(4)   u:=Q.GET()
(5)   for all  $v \in N(u)$  do {
(6)     if not marked[v] then {
(7)       Q.Put(v)
(8)       marked[v]:=1:
(9)       dist[v]:=dist[u]+1; π[v]:=u;
(10)  } } }
```

BFS-Baum

- Im Feld $\pi[v]$ werden die Vorgänger von v gespeichert.
- Die Kanten $(\pi(v), v)$ bilden einen Baum mit Wurzel s .
- Die Höhe des BFS-Baumes mit Startknoten s ist eindeutig bestimmt.
- Die Tiefe eines Blattes v entspricht dem graphentheoretischen Abstand von v zu s



Analyse von BFS

- Initialisierung: $\Theta(|V|)$
- Die **while**-Schleife wird für jeden von s aus erreichbaren Knoten genau einmal durchlaufen, da jeder Knoten höchstens einmal in die Queue kommt.
- Die **for all**-Schleife durchläuft für jeden Knoten die Liste seiner Nachbarn $N(v)$, das ist also jeweils $\Theta(d(v))$ Aufwand
- Gesamtaufwand: $\Theta(|V|) + \sum_{v \in V} \Theta(d(v)) = \Theta(|V| + |E|)$,
da im Worst Case alle Knoten von s aus erreichbar sein können

Breitensuche (BFS)

BFS löst das

Unweighted Single-Source Shortest Path
(USSSP):

- Gegeben: ungerichteter Graph $G=(V,E)$
- Gesucht: kürzester Weg von s zu jedem Knoten in G .

Breitensuche (BFS)

Theorem: Sei $G=(V,E)$ ein ungerichteter Graph und $s \in V$. Dann löst der Algorithmus $\text{BFS}(s)$ das Unweighted Single-Source Shortest Path (USSSP) für Startknoten s in Zeit $O(|V|+|E|)$.

Kap. 6.3.2 Tiefensuche

Tiefensuche (DFS)

engl.: Depth-first-search, DFS

Idee: besuche die Knoten rekursiv: wenn v zur ersten Mal gesehen wird, markieren wir ihn als gesehen und erforsche den Graphen von v aus weiter!

- Im Gegensatz zu Breitensuche, wird hier der Graph erst einmal in seiner Tiefe durchdrungen.
- Bei Breitensuche werden erst alle gesehenen Knoten bearbeitet, bevor die neuen bearbeitet werden (Queue). Hier wir immer erst das Neue erledigt.

Tiefensuche DFS

Methode: DFS-VISIT(v):

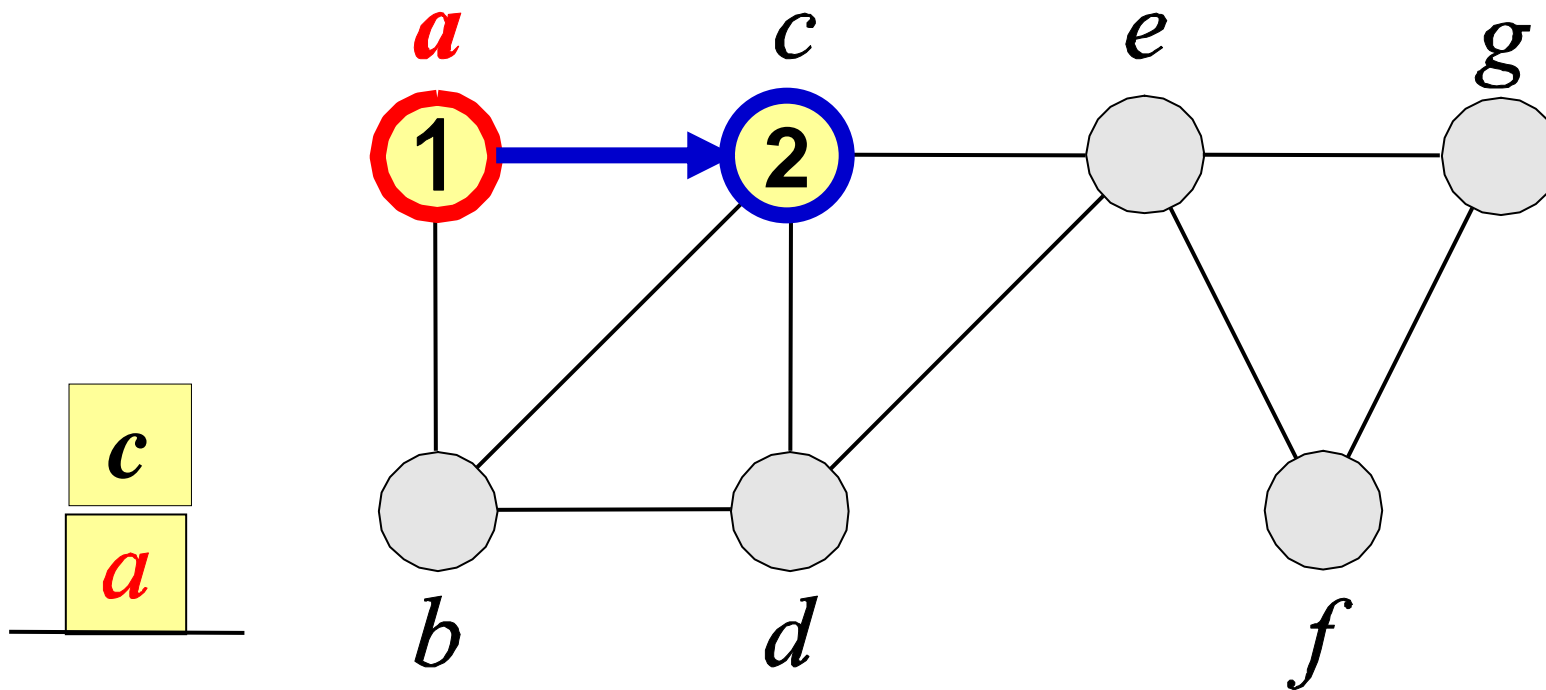
- Durchlaufe alle Nachbarn von v
- Für jeden neu entdeckten (nicht markierten) Knoten w :
 - Markiere w ; (evtl.: vergib eindeutige DFS-Nummer)
 - rufe DFS-VISIT(w) auf.
- Der folgende Algorithmus DFS(G) durchwandert alle Knoten des Graphen.
- Im Feld $\pi[w]$ wird jeweils der Vorgänger von w (also v) gespeichert.

Algorithmus DFS(G)

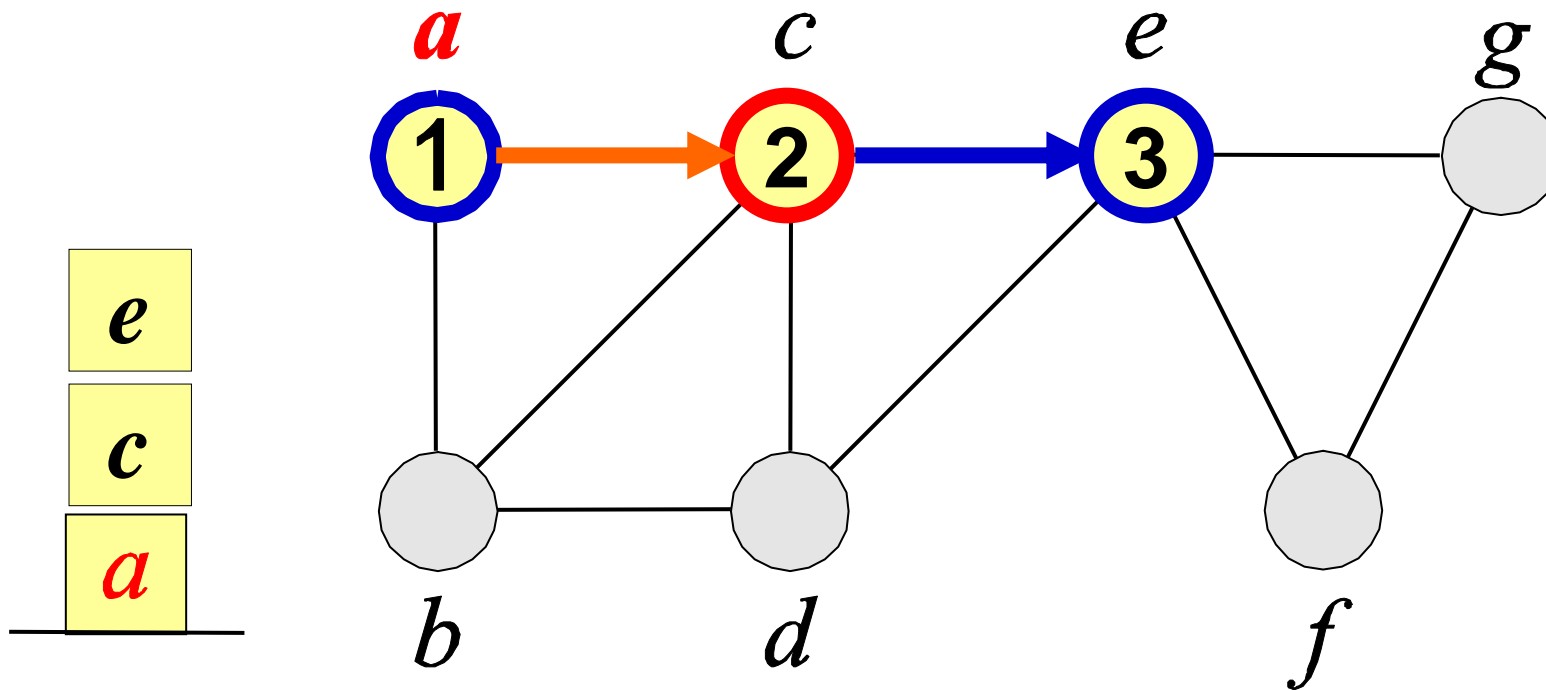
Sei $G=(V,E)$ ungerichteter Graph, v,w : Knoten

```
(1) for all  $v \in V$  do { marked[ $v$ ]:=0;  $\pi$ [ $v$ ]:=nil}
(2) for all  $v \in V$  do {
(3)   if not marked[ $v$ ] then DFS-VISIT( $v$ )
(4) }
(5) procedur DFS-VISIT(Node  $v$ )
(6)   marked[ $v$ ]:=1; // evtl.: vergib DFS-Nummer
(7)   for all  $w \in N(v)$  do
(8)     if not marked[ $w$ ] then
(9)        $\pi$ [ $w$ ]:= $v$ 
(10)      DFS-VISIT( $w$ )
(11) } }
```

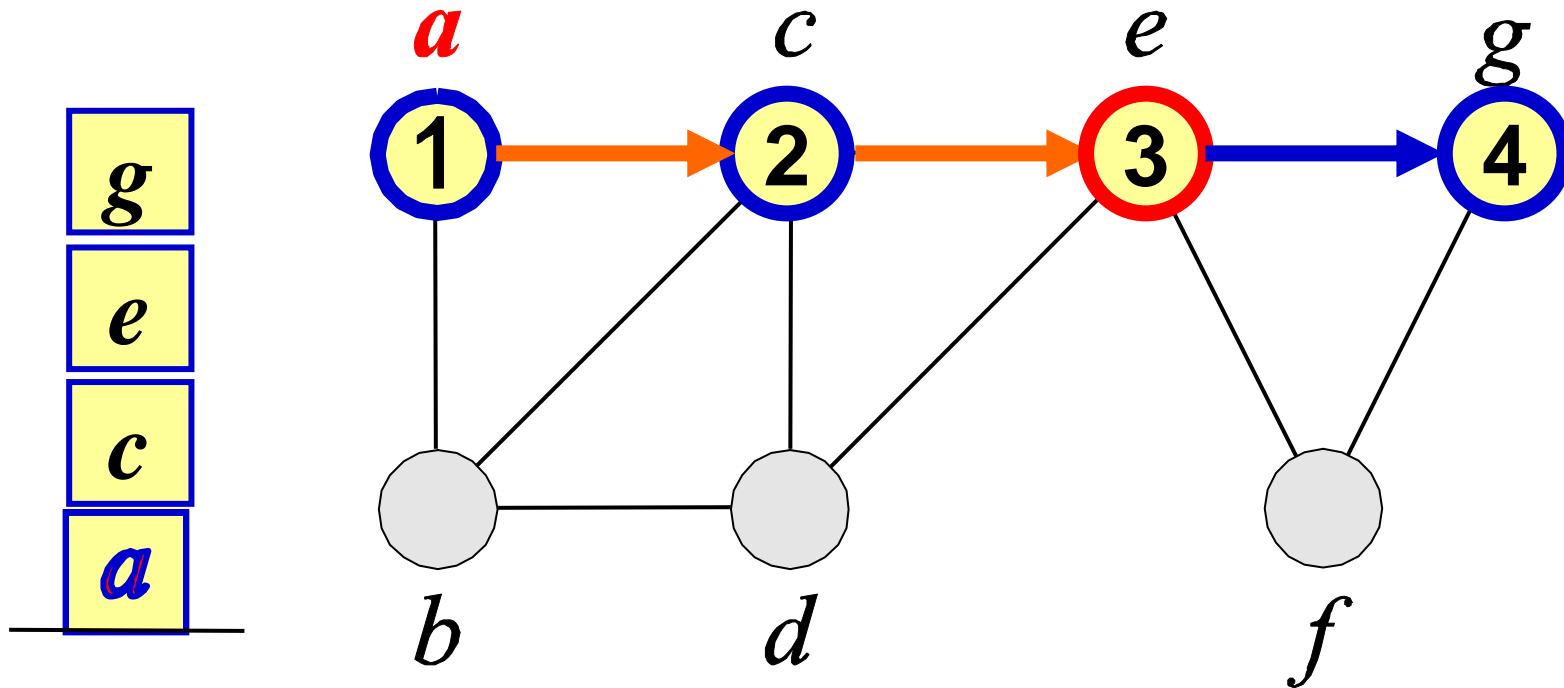
Beispiel für DFS



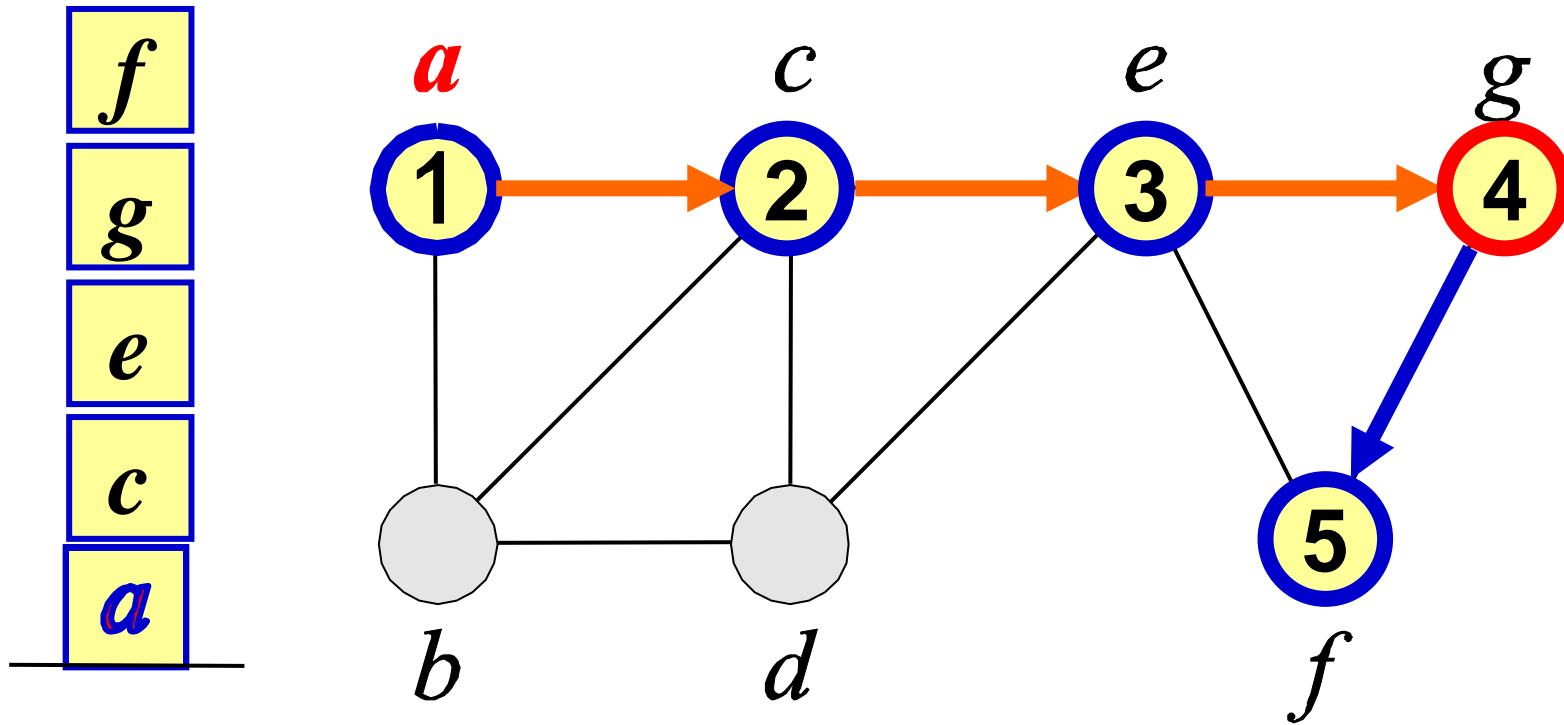
Beispiel für DFS



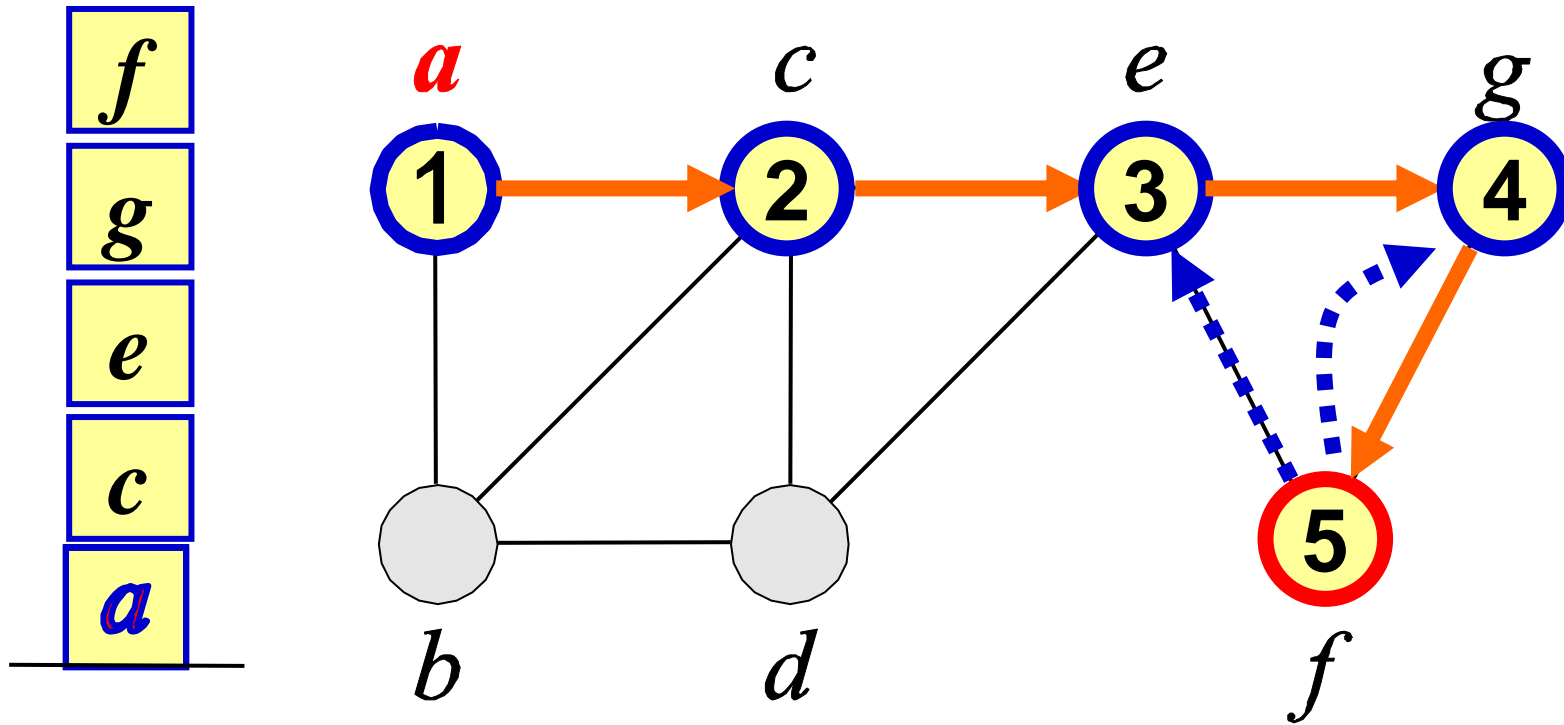
Beispiel für DFS



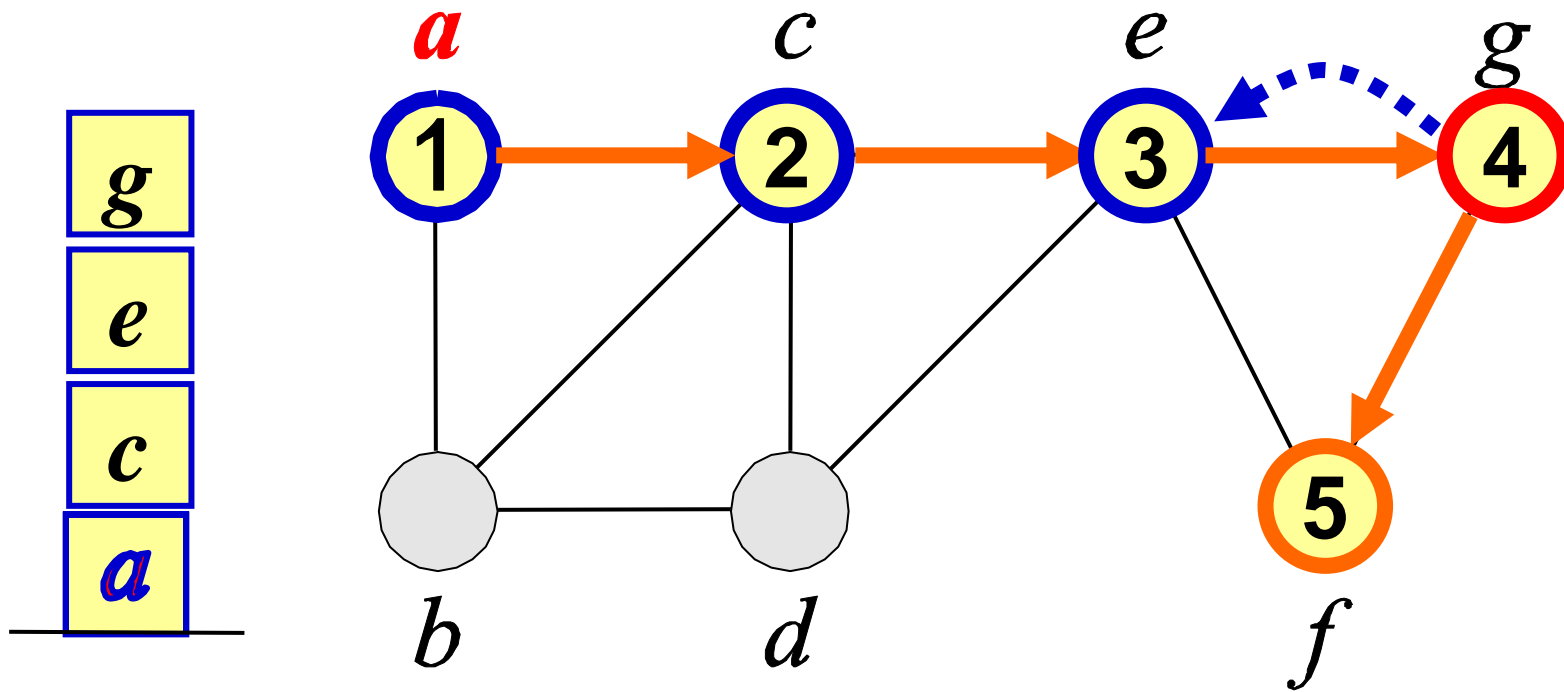
Beispiel für DFS



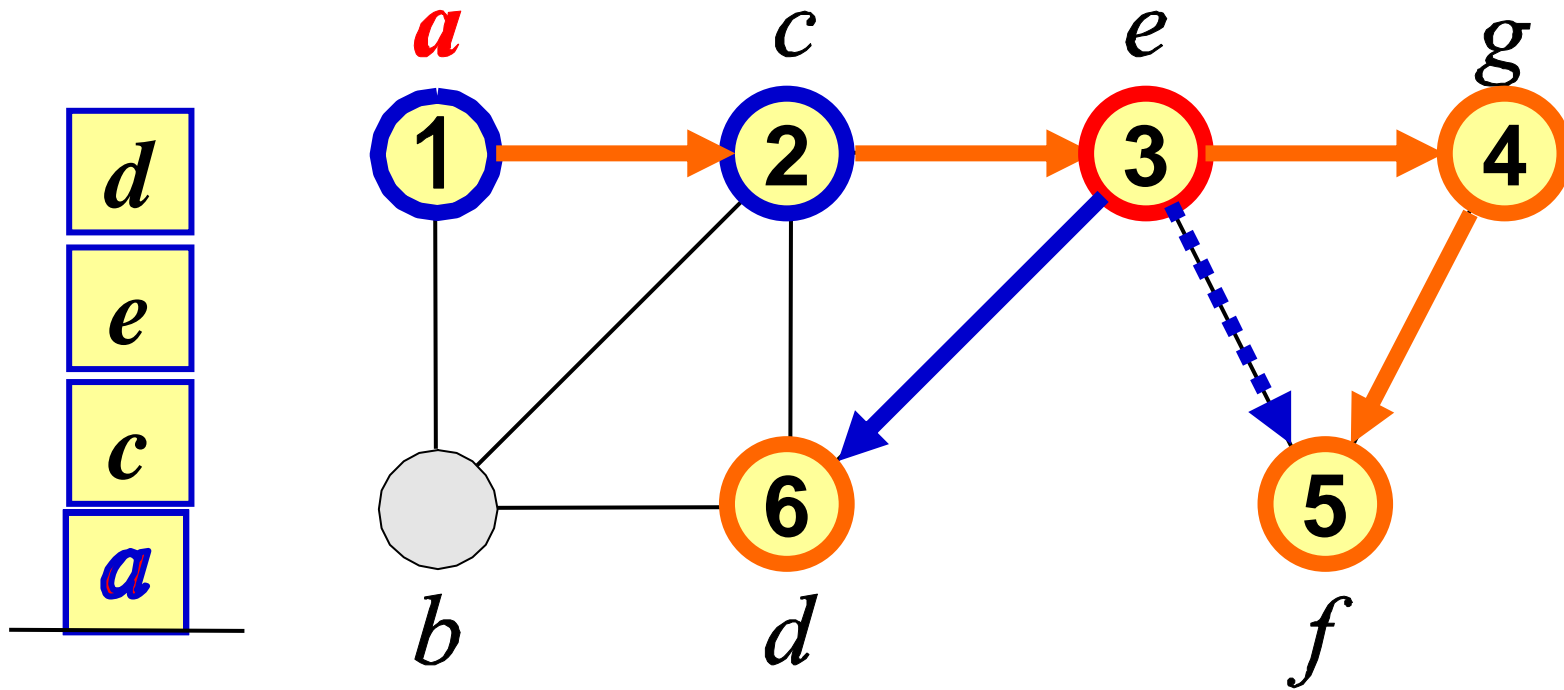
Beispiel für DFS



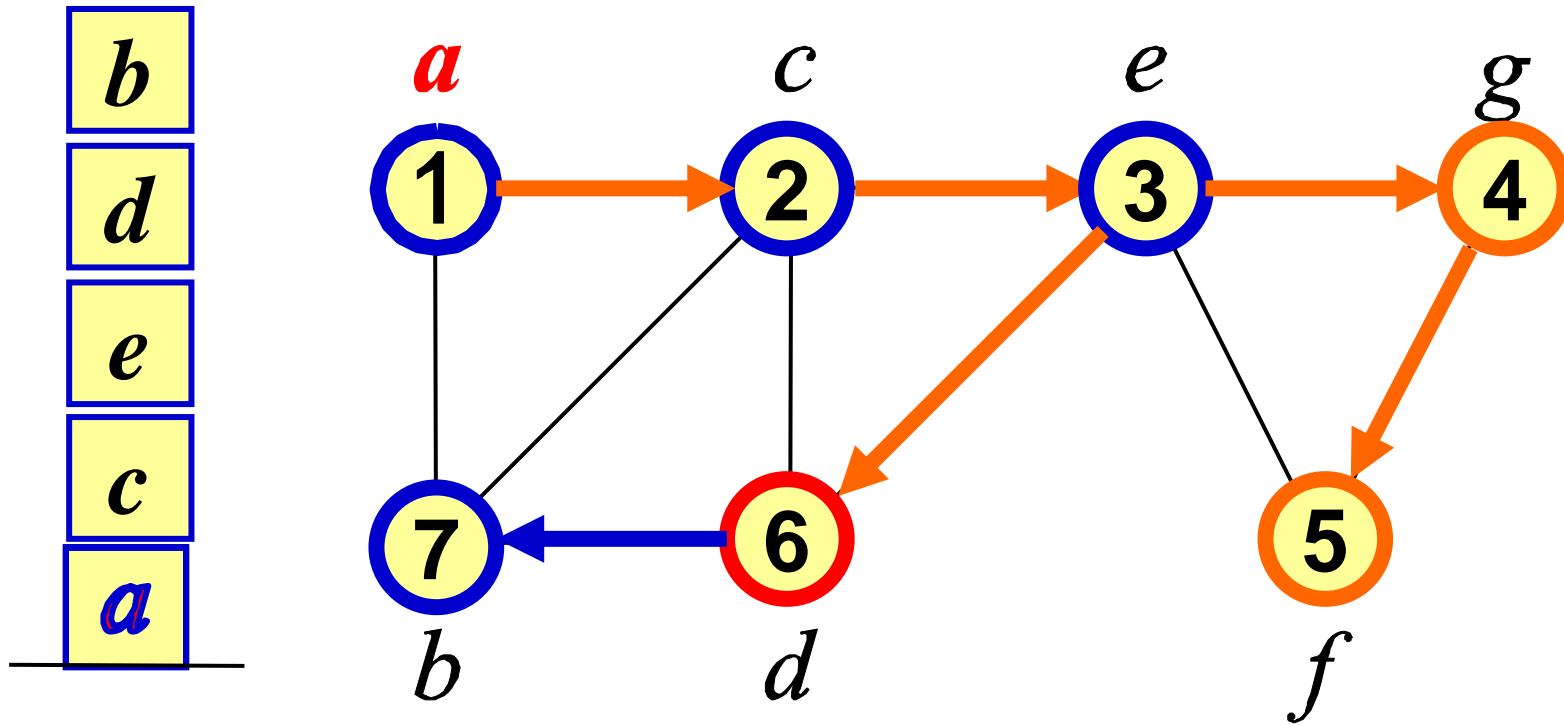
Beispiel für DFS



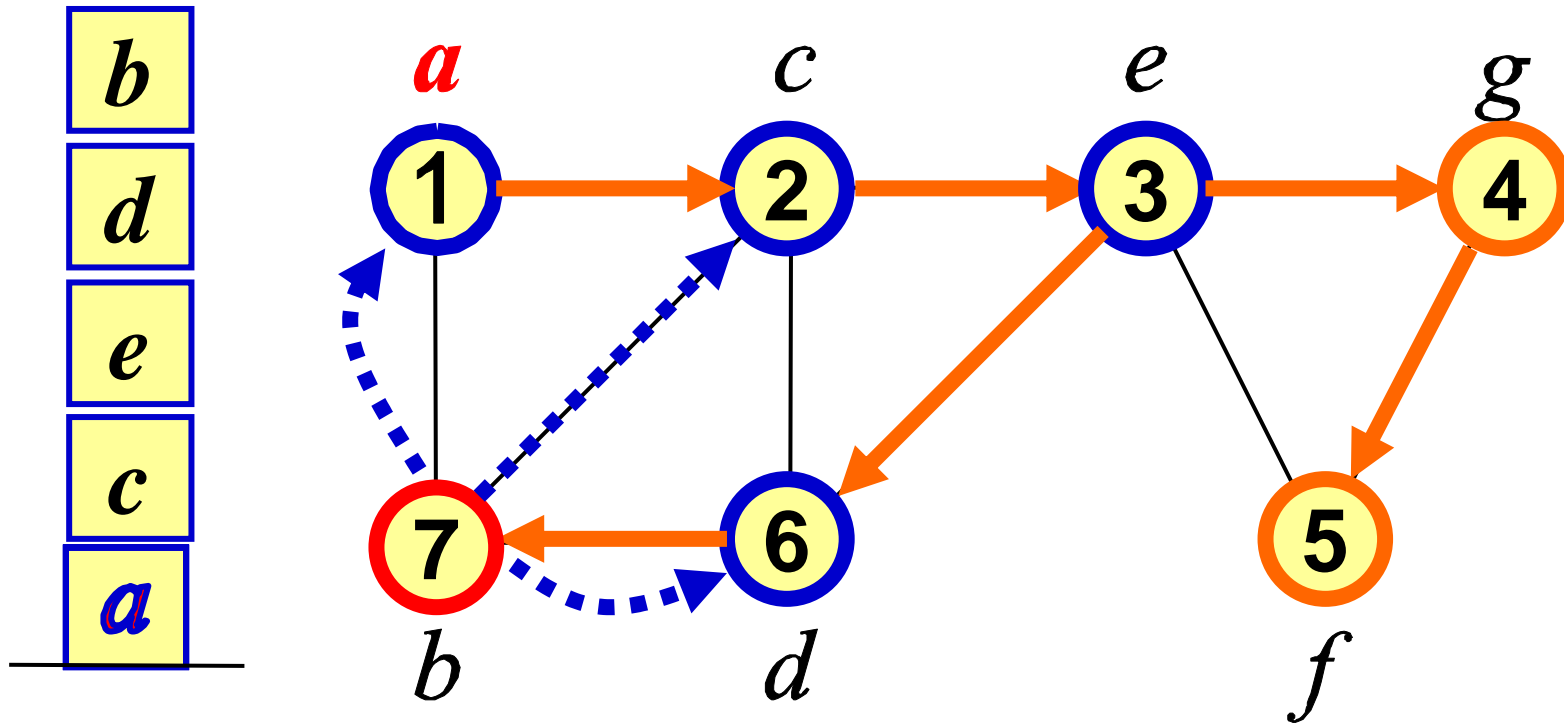
Beispiel für DFS



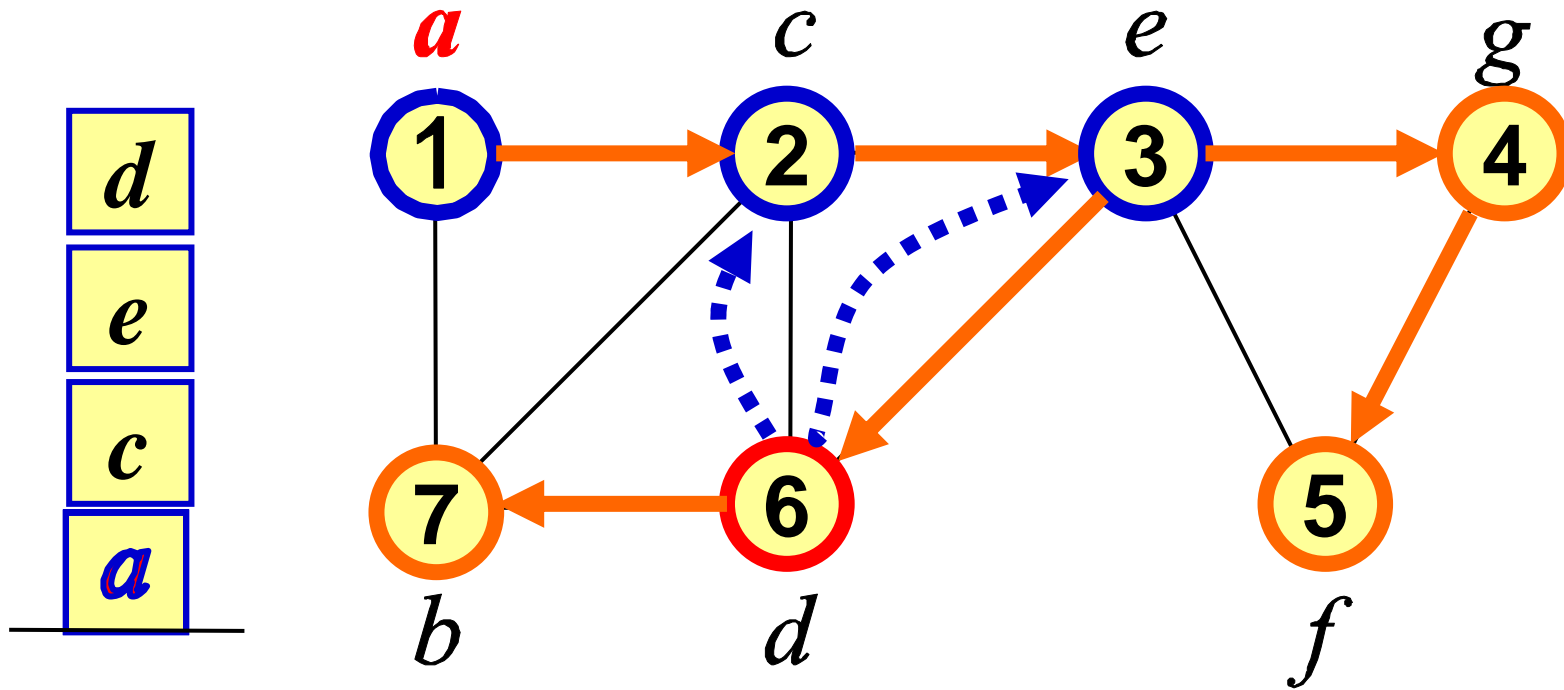
Beispiel für DFS



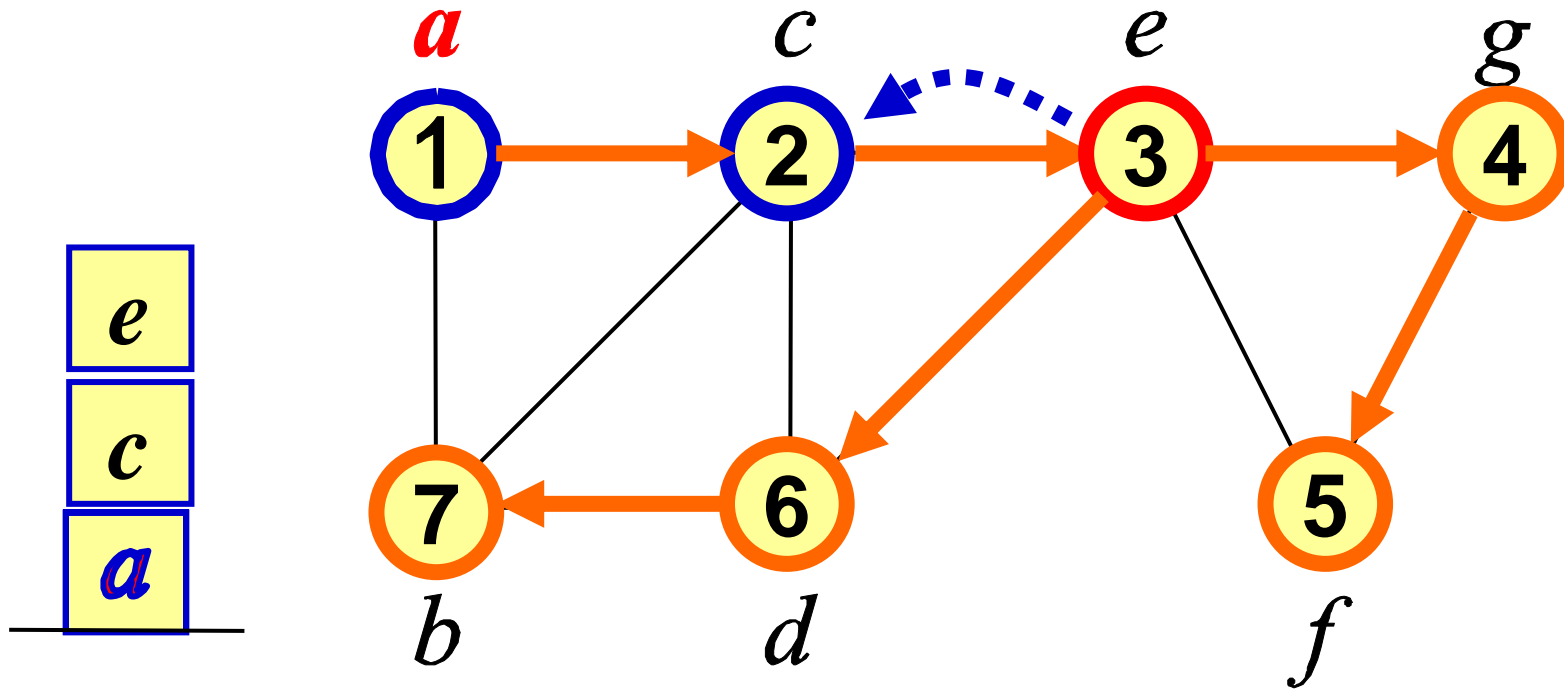
Beispiel für DFS



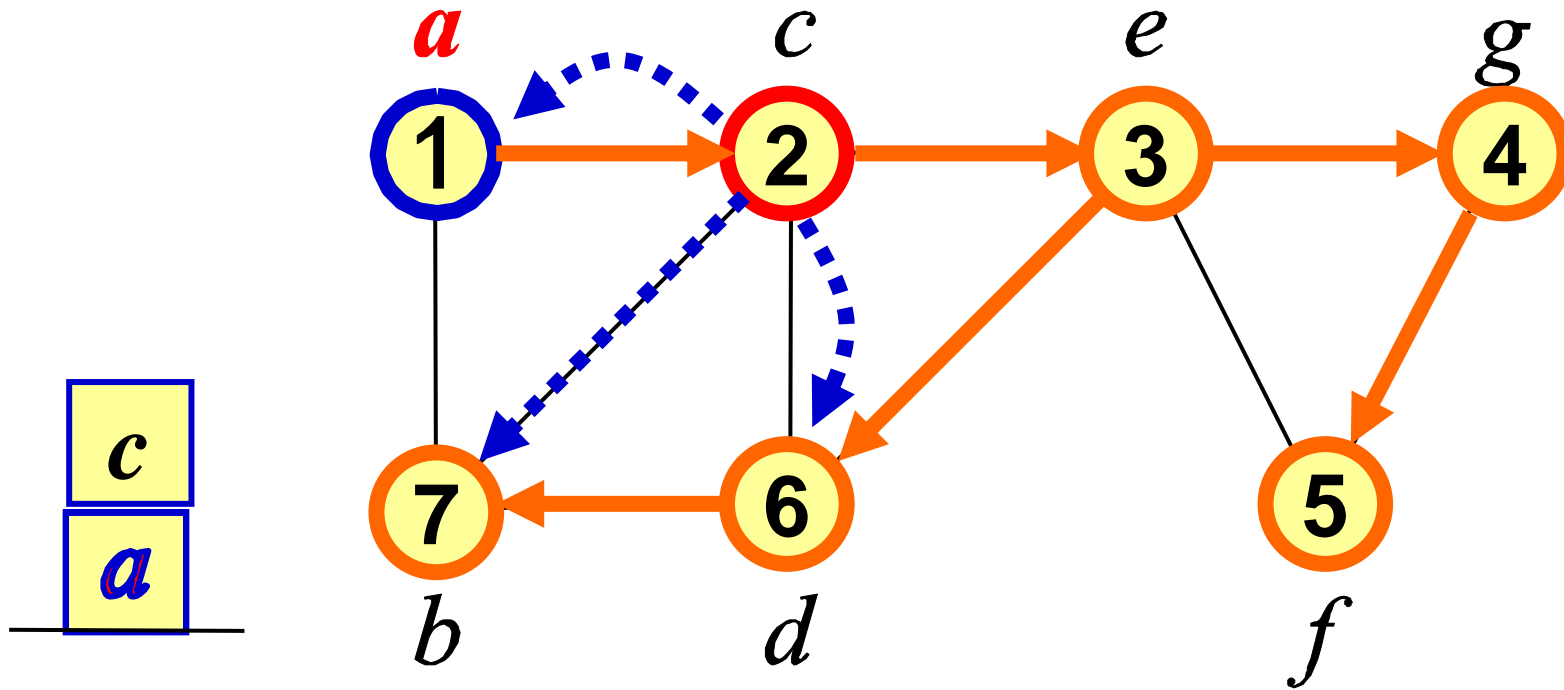
Beispiel für DFS



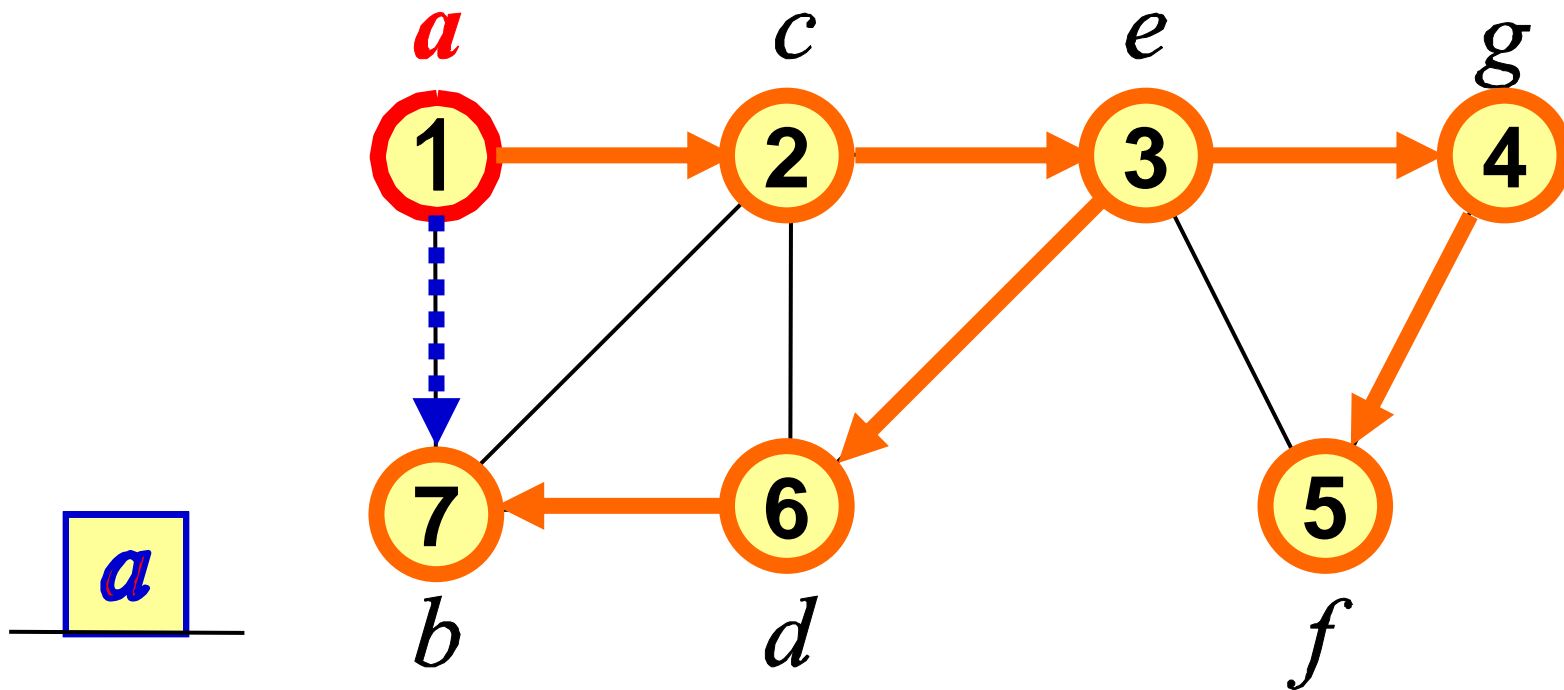
Beispiel für DFS



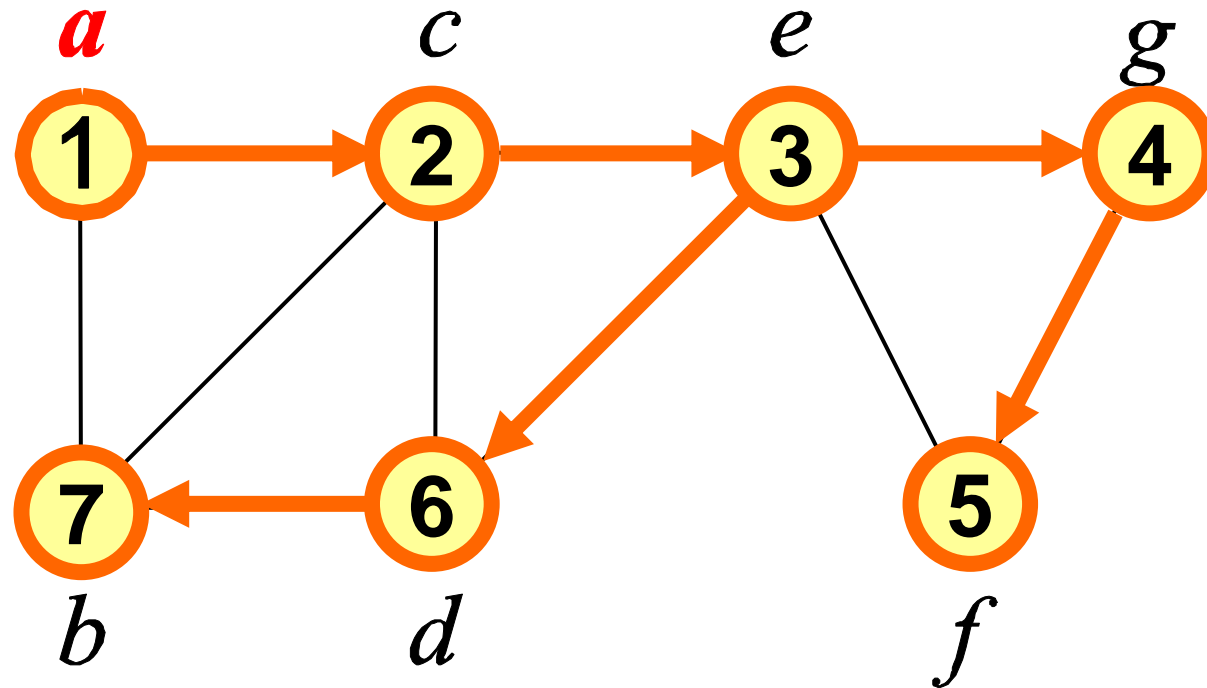
Beispiel für DFS



Beispiel für DFS



Beispiel für DFS



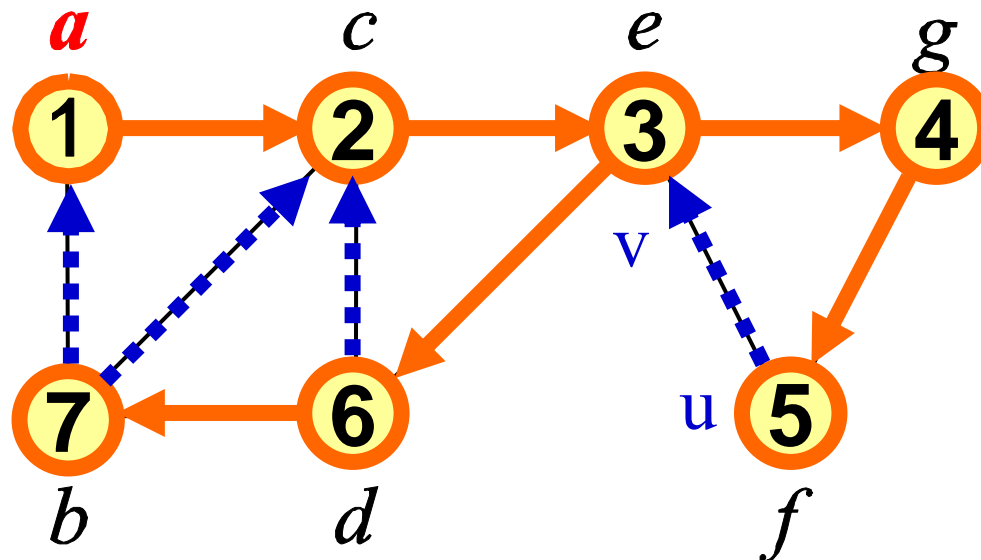
Die orangen Kanten (diejenigen Kanten (u,v) , die zum ersten Mal v besuchen) bilden einen Baum: den DFS-Baum (G zusammenhängend, sonst Wald)

DFS-Wald

DFS-Wald besteht aus

- Baumkanten (tree edges, T-Kanten): $(\pi(v), v)$
- Rückwärtskanten (back edges, B-Kanten): s.u.

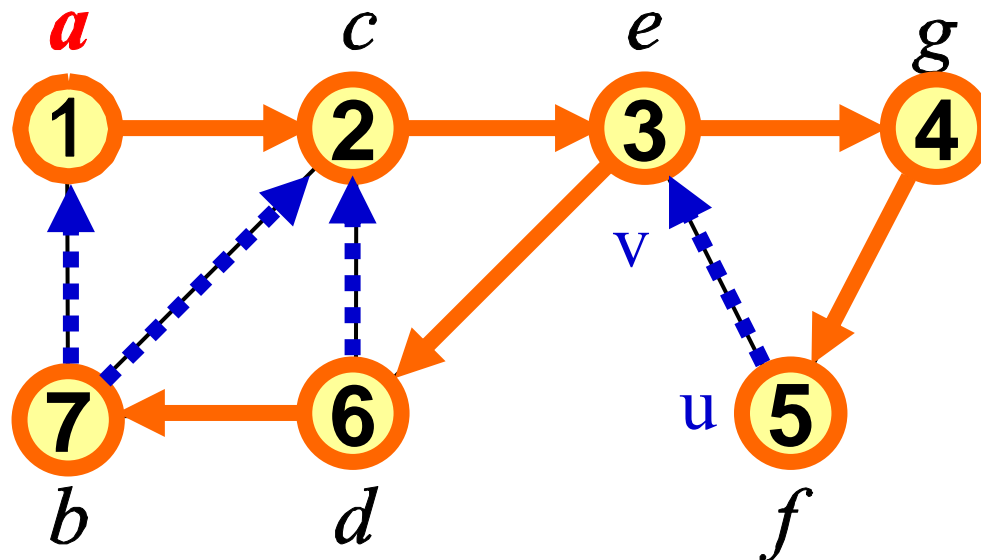
- DFS teilt die Kanten in T-Kanten und B-Kanten
- Für jede B-Kante (u, v) gilt: Es gibt genau einen Weg von v nach u mit T-Kanten im DFS-Wald



DFS-Wald

- Jeder Knoten erhält eine eindeutige DFS-Nummer.
- Für die Rückwärtskanten (u,v) gilt:
 $DFSNUM(u) > DFSNUM(v)$.

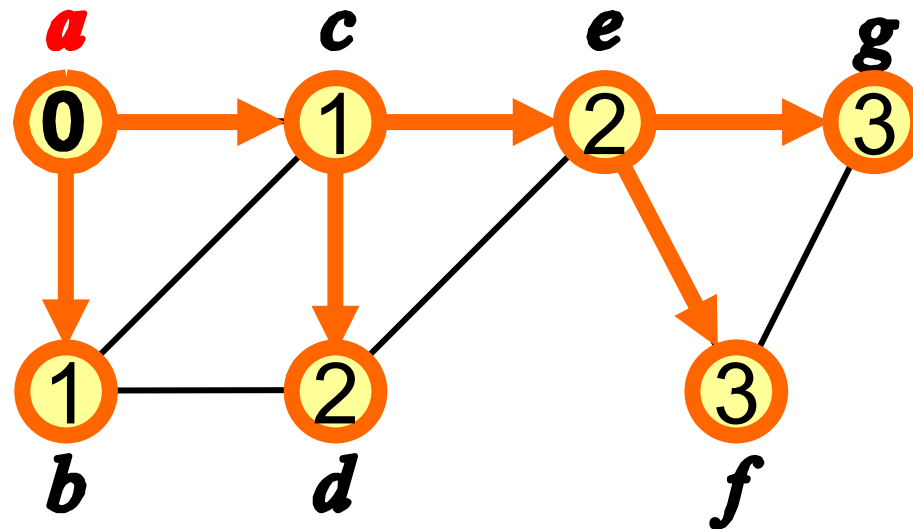
- DFS teilt die Kanten in T-Kanten und B-Kanten
- Für jede B-Kante (u,v) gilt: Es gibt genau einen Weg von v nach u mit T-Kanten im DFS-Wald



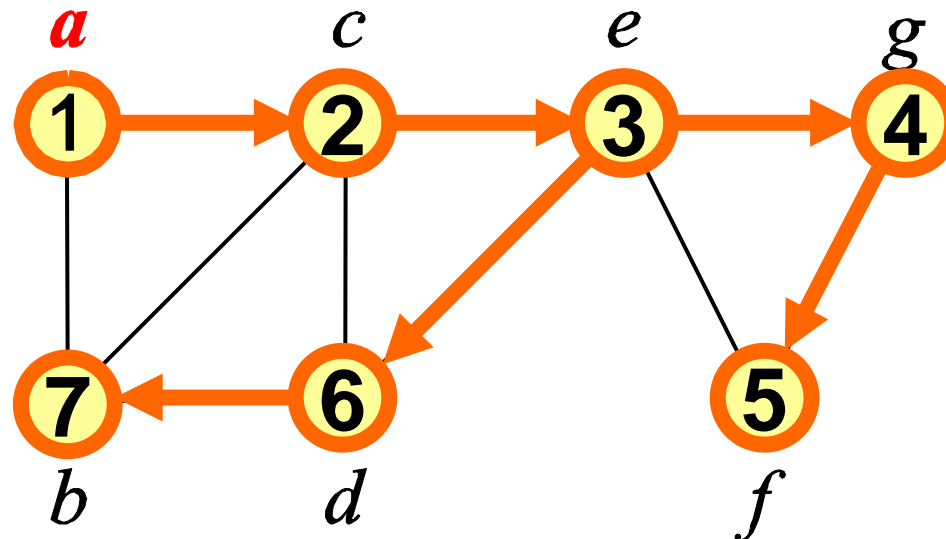
BFS-Baum vs. DFS-Baum

BFS-Baum

Höhe eindeutig



DFS-Baum



Analyse von DFS

- Initialisierung: $\Theta(|V|)$
- Die **for-all**-Schleife ohne Aufrufe von DFS-VISIT(): $O(|V|)$.
- DFS-VISIT() wird für jeden Knoten genau einmal aufgerufen.
- Jeder Aufruf von DFS-VISIT(v) (ohne Kosten der rekursiven Aufrufe) benötigt $\Theta(d(v))$ Zeit.
- Gesamtaufwand: $\Theta(|V|) + \sum_{v \in V} \Theta(d(v)) = \Theta(|V| + |E|)$

Kap. 6.4 Elementare Graphenalgorithmen

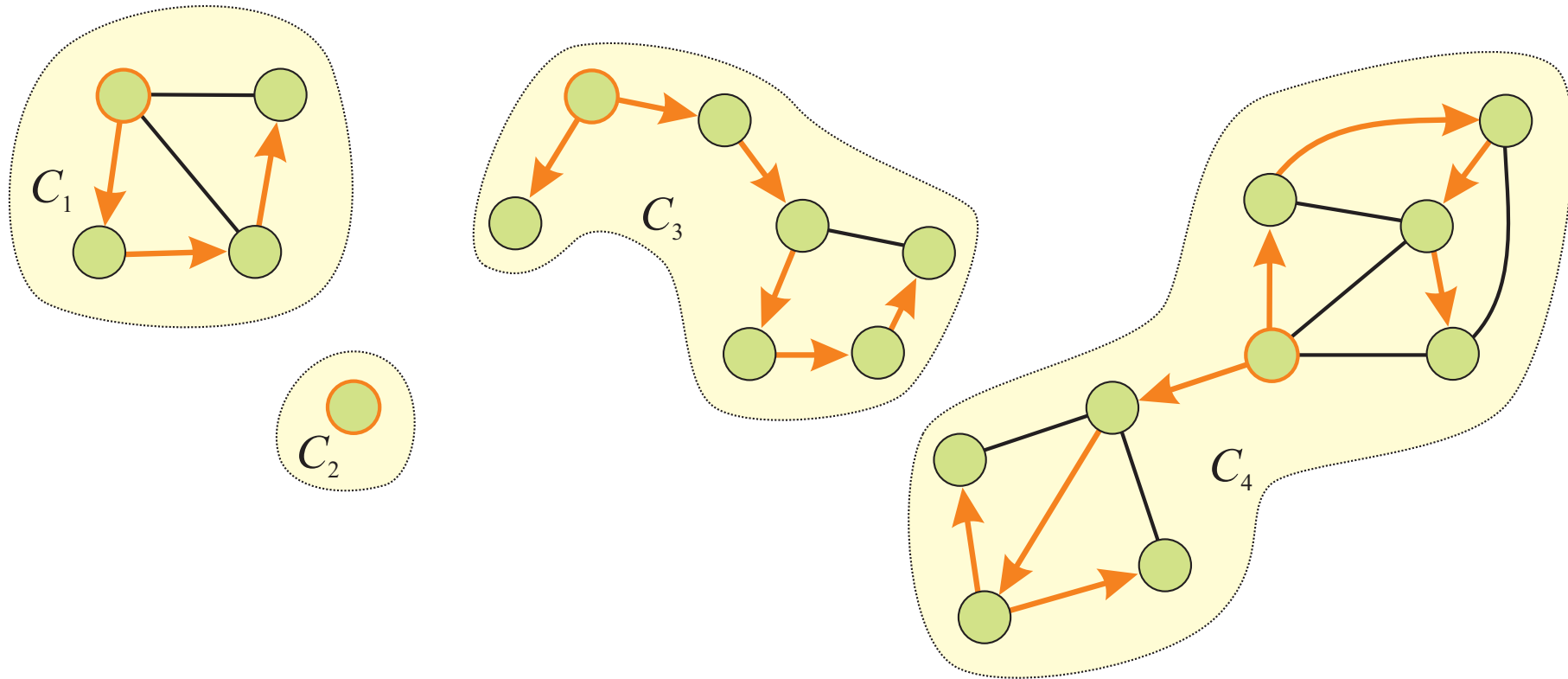
Kap. 6.4.1 Komponenten eines Graphen

Zusammenhangskomponenten

Definitionen (Zusammenhang)

- G heißt **zusammenhängend (connected)**, wenn $|V| \geq 1$ und für jedes Paar u, v von Knoten ein Weg von u nach v in G existiert.
- Ein Graph $G' = (V', E')$ mit $V' \subseteq V$ und $E' \subseteq E$ ist ein **Untergraph (Teilgraph, subgraph)** von G . Wir schreiben $G' \subseteq G$.
- Eine **(Zusammenhangs-) Komponente (component)** von G ist ein maximaler zusammenhängender Untergraph U von G , d.h.
es gibt keinen anderen zusammenhängenden Untergraphen U' mit $U \subseteq U'$.

Graph mit 4 Komponenten



→ kennzeichnet einen möglichen DFS-Wald

Bestimmen der Zusammenhangskomponenten

Idee:

- Beginne an einem Knoten s und bestimme alle von s aus erreichbaren Knoten \rightarrow Komponente 1 (Markierung mit 1)
- Beginne nun bei einem noch unmarkierten Knoten v (`marked==0`) und bestimme alle von v aus erreichbaren Knoten; markiere dabei mit 2 \rightarrow Komponente 2
- ...u.s.w. bis alle Knoten markiert sind

Algorithmus COMPONENTS(G)

Sei $G=(V,E)$ ungerichteter Graph, v,w : Knoten

- (1) **for all** $v \in V$ **do** { marked[v]:=0 }
- (2) numcomps:=0
- (3) **for all** $v \in V$ **do** {
- (4) **if not** marked[v] **then** {
- (5) numComps:=numComps+1
- (6) DFS-COMP(v)
- (7) } }
- (8) **Procedure** DFS-COMP(Node v)
- (9) marked[v]:=numComps
- (10) **for all** $w \in N(v)$ **do**
- (11) **if not** marked[w] **then**
- (12) DFS-COMP(w)

Bestimmen der Zusammenhangskomponenten

Theorem: Der Algorithmus COMPONENTS(G) bestimmt die Zusammenhangskomponenten eines Graphen $G=(V,E)$ in Zeit $\Theta(|V|+|E|)$.

Kap. 6.4.2 Kreise in Graphen

- Aufgabe: Gegeben ist ein ungerichteter Graph $G=(V,E)$. Enthält G einen Kreis?
- Idee: mittels DFS

Kreise und DFS

- **Lemma:** Sei G ein ungerichteter Graph und F ein beliebiger DFS-Wald für G . Dann ist G genau dann kreisfrei, wenn G keine Rückwärtskante (B-Kante) bezüglich F enthält.

Beweis:

- 1. Fall: F enthält keine B-Kante: dann besteht F nur aus T-Kanten $\rightarrow F$ enthält alle Kanten aus $G \rightarrow G$ kreisfrei
- 2. Fall: Sei (u,v) B-Kante bzgl. $F \rightarrow$ es existiert Weg v, \dots, u mit T-Kanten in $F \rightarrow$ Weg mit Kante (u,v) bildet Kreis in G .

Kreise und DFS

- **Methode:** Erweitere DFS um Speicherung der B-Kanten: sobald wir von v aus auf einen markierten Knoten u treffen, haben wir eine B-Kante gefunden, falls u nicht der direkte Vorgänger (Elter) von v ist.

Algorithmus ISACYCLIC(G) speichert dabei alle B-Kanten. Wenn man nur einen Kreis finden möchte, dann kann man abbrechen, sobald man eine B-Kante gefunden hat.

Algorithmus ISACYCLIC(G)

Sei $G=(V,E)$ ungerichteter Graph

```
(1) function ISACYCLIC(Graph  $G=(V,E)$ ):bool {  
(2)    $B:=\emptyset$ ;  $NUM:=0$  //NUM: höchste bish. DFSNUM  
(3)   for all  $v \in V$  do {  $\text{marked}[v]:=0$ ;  $\pi(v):=\text{nil}$  }  
(4)   for all  $v \in V$  do {  
(5)     if not  $\text{marked}[v]$  then {  
(6)       DFS-ACYCLIC( $v$ )  
(7)     }  
(8)   }  
(9)   if  $B \neq \emptyset$  then return 0 else return 1  
(10) }
```

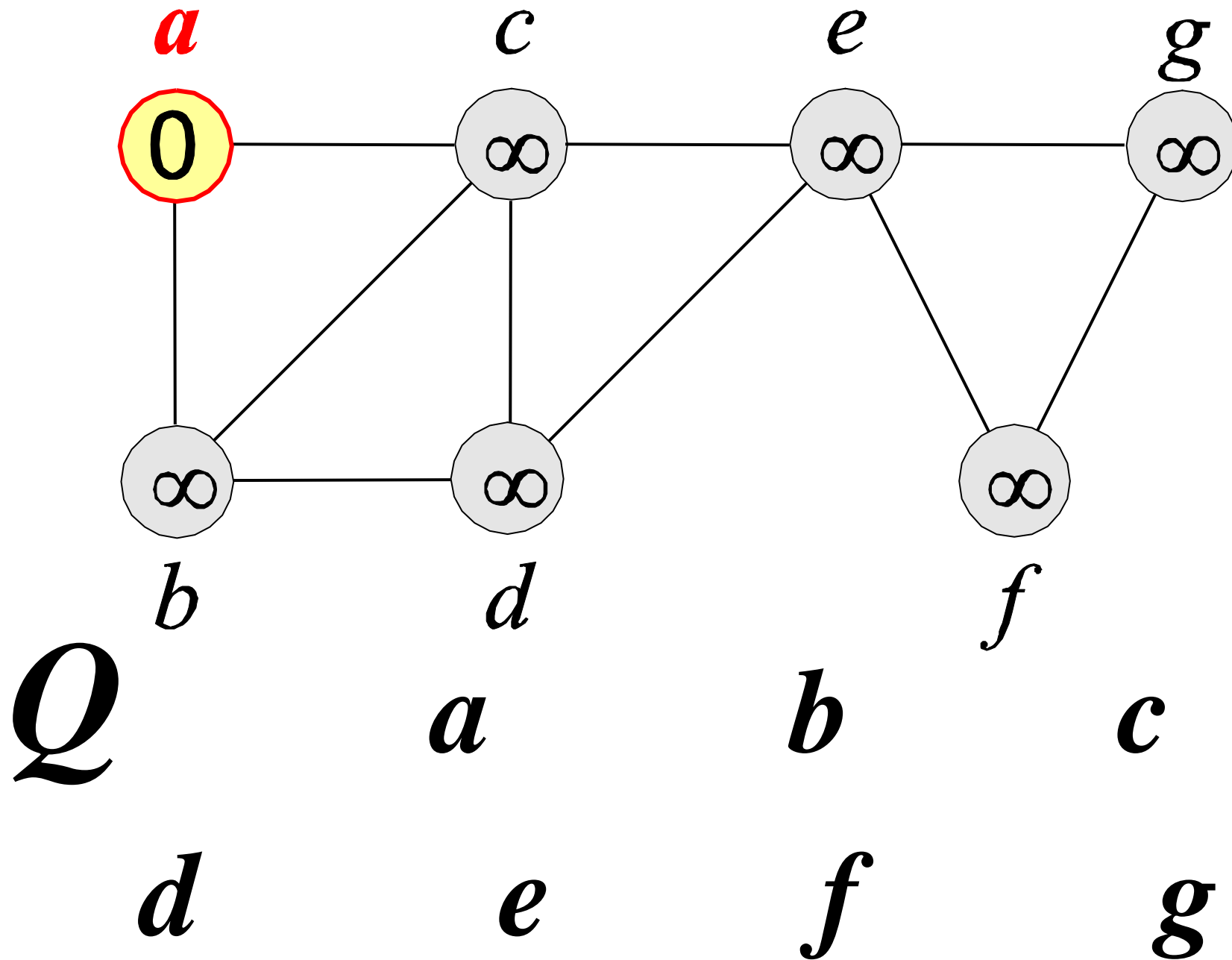
Algorithmus ISACYCLIC(G) ff.

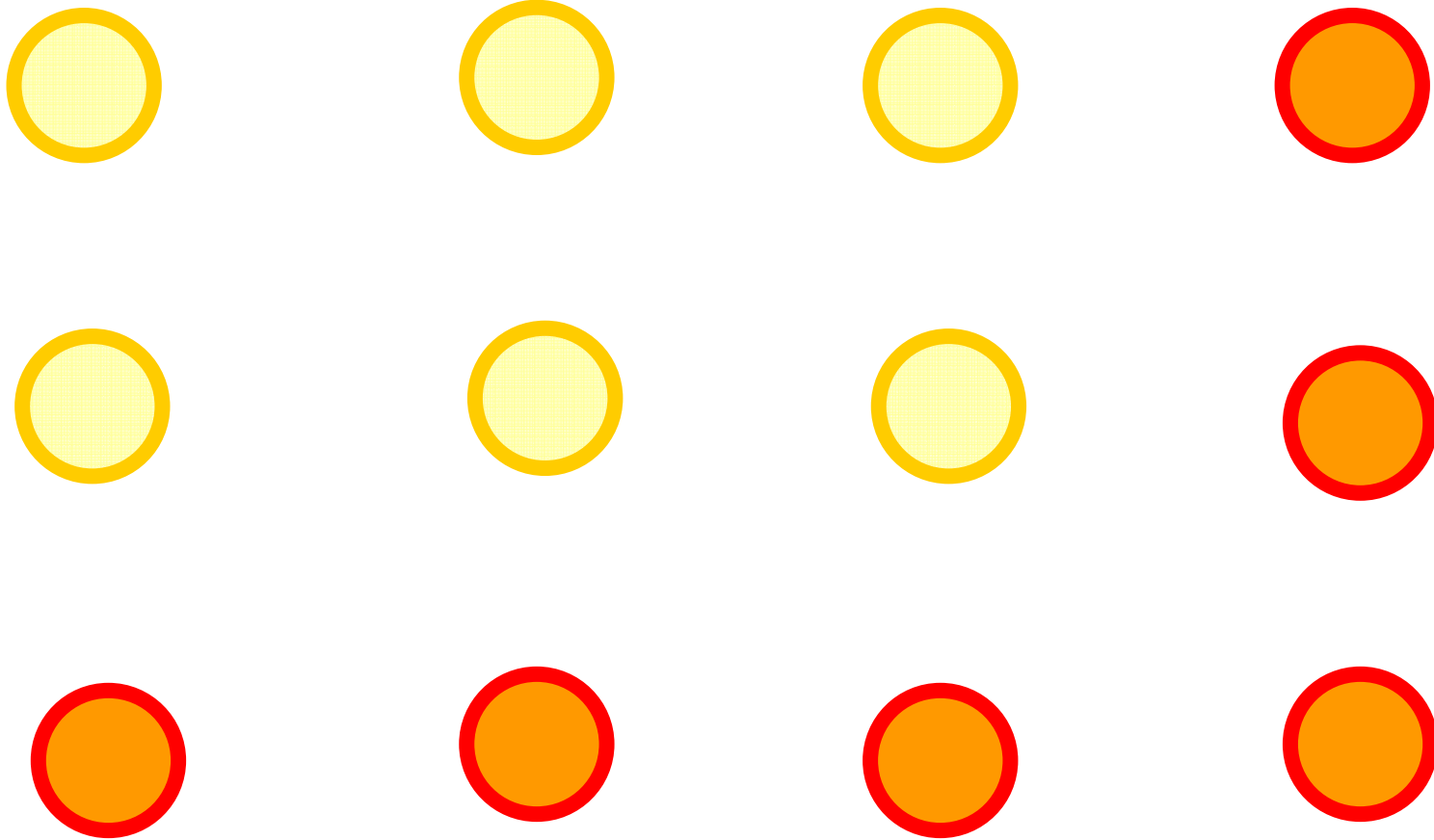
```
(1) Procedure DFS-ACYCLIC(Node  $v$ ) {  
(2)   marked[ $v$ ]:=1;  
(3)   NUM:=NUM+1; DFSNUM[ $v$ ]:=NUM;  
(4)   for all  $w \in N(v)$  do  
(5)     if not marked[ $w$ ] then  
(6)        $\pi[w]$ := $v$   
(7)       DFS-ACYCLIC( $w$ )  
(8)     else if  $\pi[v] \neq w$  AND  
(9)       ( DFSNUM( $v$ )>DFSNUM( $w$ )) then  
(10)       $B$ :=BU( $v, w$ )  
(11) }  
(12) }
```

Test auf Kreisfreiheit

Theorem: Die Funktion ISACYCLIC(G) testet einen ungerichteten Graphen $G=(V,E)$ auf Kreisfreiheit in Zeit $\Theta(|V|+|E|)$.

Beispiel zum Nachspielen: Graph





Beispiel zum Nachspielen: Knotenmarkierungen

Beispiel zum Nachspielen: Kantenmarkierungen

