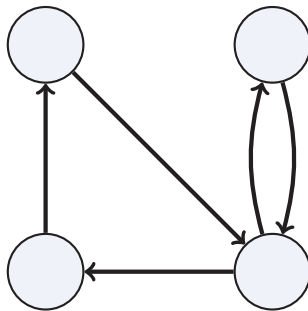


## 3 Starke Zusammenhangskomponenten (SZK) (SCC)

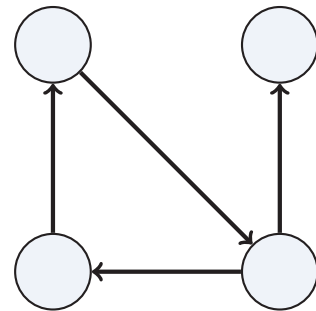
### 3.1 Definition

- a) Ein gerichteter Graph ist stark zusammenhängend  $\Leftrightarrow \forall v, w \in V : v \xrightarrow{*} w$   
(es existiert ein Pfad von v nach w)

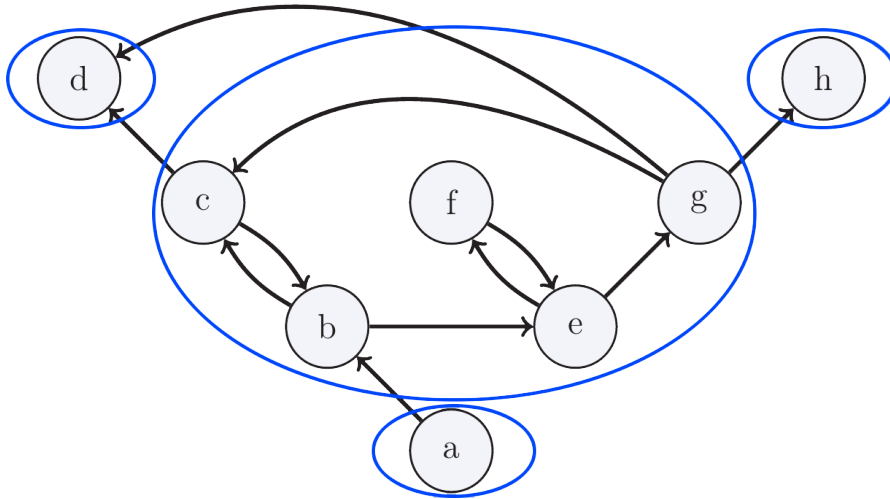
stark zusammenhängend



nicht stark zusammenhängend



- b) Die starken Zusammenhangskomponenten (SZK) von G sind die maximalen stark zusammenhängenden Teilgraphen von G.

**Beispiel 3.1****Idee für Algorithmus**

Führe DFS aus. Sei  $G' = (V', E')$  der Teilgraph aufgespannt von den bereits besuchten Knoten.

Verwalte die SZK von  $G'$  (während DFS ausgeführt wird).

**Im Beispiel 3.1:**

Initialisierung:  $V' = \{a\}$ ,  $E' = \emptyset$

Ablauf:

Sei  $(v, w)$  die nächste in dfs betrachtete Kante.

- 1.Fall  $(v, w) \in T$  (Baumkante) ( $w$  noch nicht besucht)

$$V' = V' \cup \{w\}$$

$$E' = E' \cup \{(v, w)\}$$

$$SZK = SZK \cup \{\{w\}\}$$

Im Beispiel:  $SZK = \{\{a\}, \{b\}\}$

- 2.Fall  $(v, w) \notin T$  dh.  $w$  ist bereits besucht ( $w \in V'$ )

$V'$  unverändert

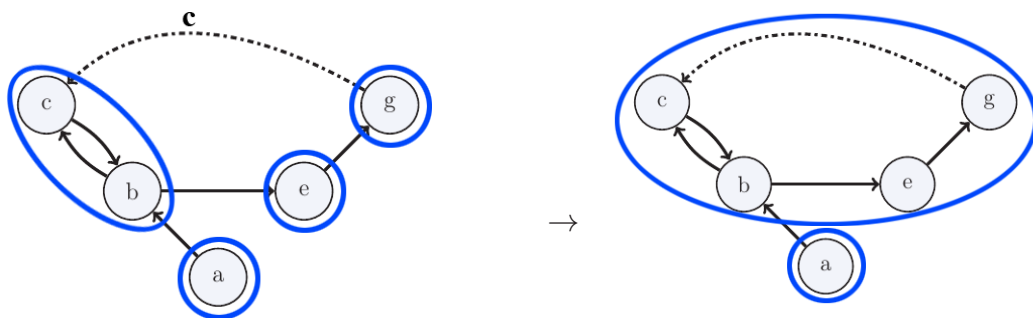
$$E' = E' \cup \{(v, w)\}$$

$(v, w)$  kann mehrere SZK zu einer vereinigen

Im Beispiel: Rückwärtskante  $(c, b) \in B$

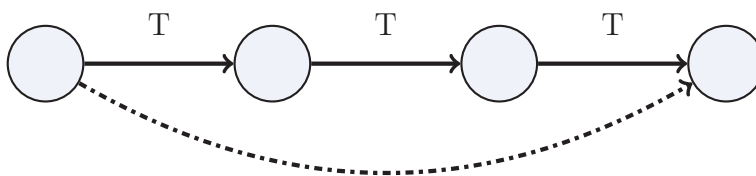


oder Crosskante (Menge C)



### Bemerkung:

Vorwärtskanten generieren keine neuen Pfade in  $G$ . Deswegen wird SZK nicht geändert. Daher können Vorwärtskanten ignoriert werden.



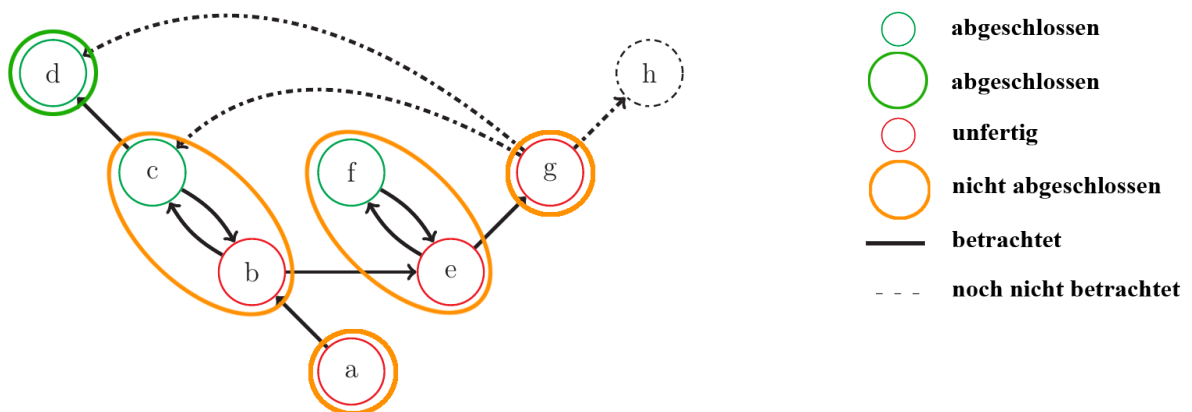
### Definition 3.1

- Eine SZK  $K$  heißt abgeschlossen, falls die Aufrufe von dfs für alle  $v \in K$  abgeschlossen sind.
- Die Wurzel einer SZK  $K$  ist der Knoten mit der kleinsten dfsnum in  $K$ .
- Unfertig = Folge aller Knoten für die dfs aufgerufen wurde, aber deren SZK noch nicht abgeschlossen ist. (Sortiert nach dfsnum)

- d) Wurzeln = Folge der Wurzeln der nicht abgeschlossenen SZK. (Sortiert nach dfsnum)  $\rightarrow$  Teilfolge von „unfertig“

### Im Beispiel 3.1:

Situation, wenn DFS beim Knoten g angelangt ist.



unfertig:  $a \mid b \mid c \mid e \mid f \mid g$   
Wurzeln:  $a \mid b \mid e \mid g$

Der Algorithmus betrachtet danach die Kanten, die aus g heraus gehen.

$(g, d) \in C$

Es passiert nichts, da d in einer abgeschlossenen SZK liegt. ( $\Rightarrow (g, d)$  schließt keinen Kreis.)

$(g, c) \in C$

Die Kante vereinigt die drei SZK mit den Wurzeln b,e und g durch streichen von e und g aus der Wurzelfolge.

unfertig:  $a \mid b \mid c \mid e \mid f \mid g$   
Wurzeln:  $a \mid b$

$(g, h) \in T$

Die Kante fügt den Knoten h an beide Folgen an.

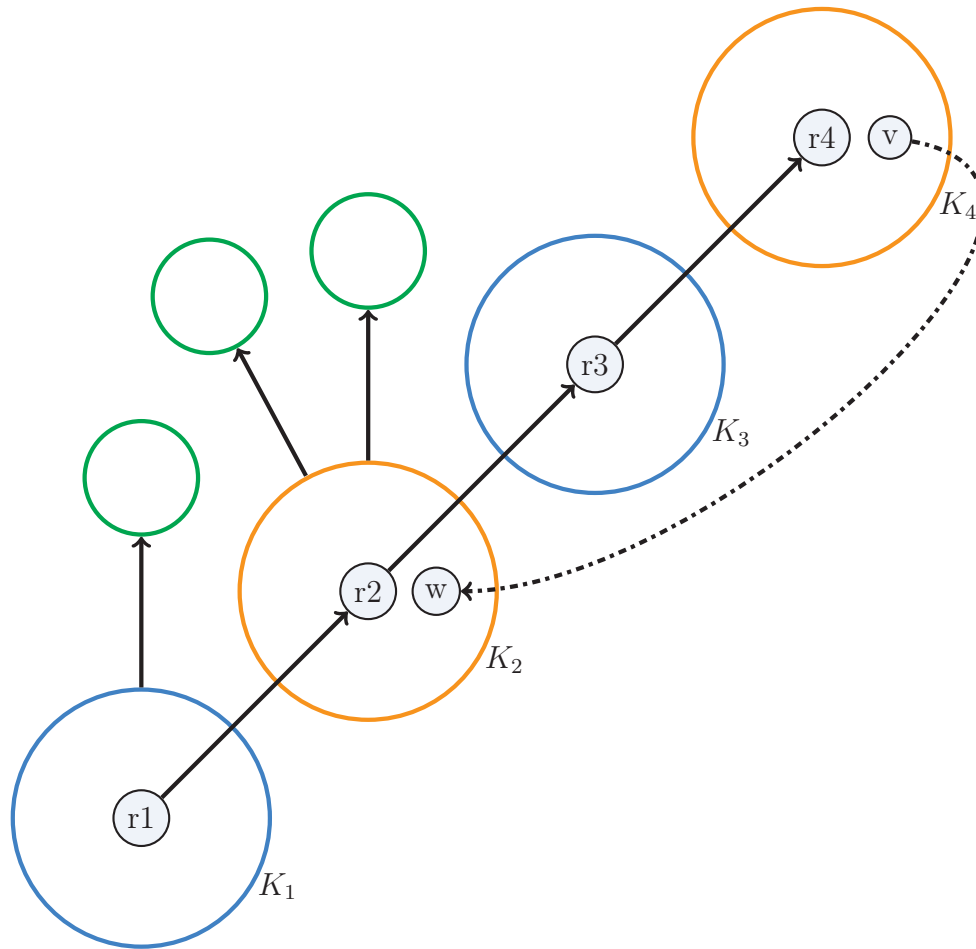
unfertig:  $a \mid b \mid c \mid e \mid f \mid g \mid h$   
Wurzeln:  $a \mid b \mid h$

**Beobachtung:**

Hinzufügen oder Löschen eines Knotens geschieht nur am Ende. Daher nutzt man für die beiden Folgen als Datenstruktur einen Stack.

**Allgemeine Situation für  $(v, w) \notin T$** 

$w$  ist unfertig und liegt daher in einer nicht abgeschlossenen SZK.



In diesem Beispiel ist die Kante  $(v, w)$  eine Cross- oder Rückwärtskante. Sie vereinigt dadurch die SZK  $K_2$ ,  $K_3$  und  $K_4$  zu einer SZK  $K'$ .