Script generated by TTT

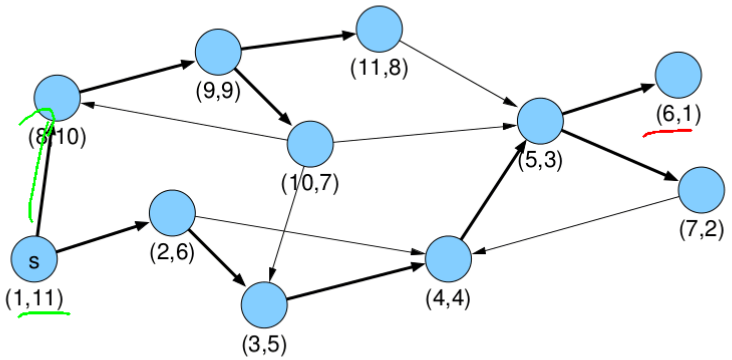
Title: TÃ¤ubig: GAD (02.07.2013)

Date: Tue Jul 02 14:16:52 CEST 2013

Duration: 81:26 min

Pages: 30

Tiefensuche



DFS-Nummerierung

Beobachtung:

- Knoten im DFS-Rekursionsstack (aktiven Knoten) sind bezüglich dfsNum aufsteigend sortiert

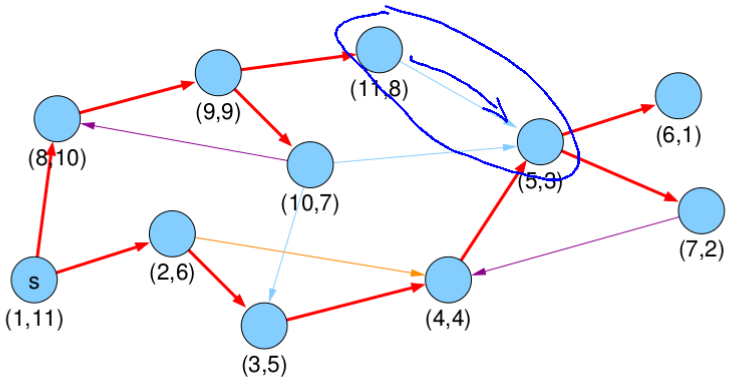
Begründung:

- dfsCount wird nach jeder Zuweisung von dfsNum inkrementiert
- neue aktive Knoten haben also immer die höchste dfsNum

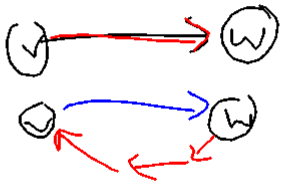
DFS-Nummerierung

Kantentypen:

- Baumkanten: zum Kind
- Vorwärtskanten: zu einem Nachfahren
- Rückwärtskanten: zu einem Vorfahren
- Kreuzkanten: sonstige



DFS-Nummerierung



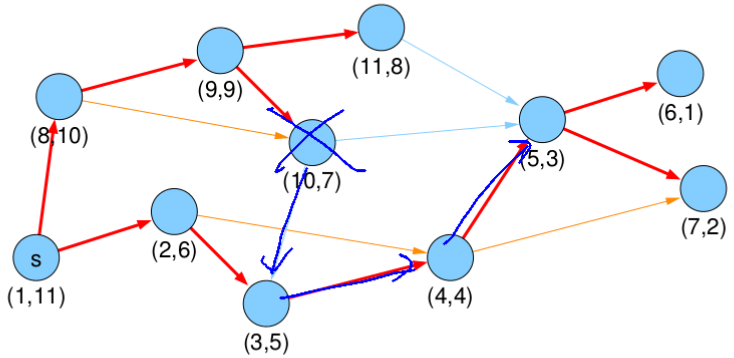
Beobachtung für Kante (v, w):

Kantentyp	dfsNum[v] < dfsNum[w]	finishNum[v] > finishNum[w]
Baum & Vorwärts	ja	ja
Rückwärts	nein	nein (umgekehrt)
Kreuz	nein	ja

DAG-Erkennung per DFS

Anwendung:

- Erkennung von azyklischen gerichteten Graphen (engl. directed acyclic graph / DAG)



- keine gerichteten Kreise

DAG-Erkennung per DFS

Lemma

Folgende Aussagen sind äquivalent:

1. *Graph G ist ein DAG.*
2. *DFS in G enthält keine Rückwärtskante.*
3. $\forall (v, w) \in E$: *finishNum[v] > finishNum[w]*

DAG-Erkennung per DFS

Lemma

Folgende Aussagen sind äquivalent:

1. *Graph G ist ein DAG.*
2. *DFS in G enthält keine Rückwärtskante.*
3. $\forall (v, w) \in E$: *finishNum[v] > finishNum[w]*

Beweis.

- (2)⇒(3): Wenn (2), dann gibt es nur Baum-, Vorwärts- und Kreuzkanten.
  Für alle gilt (3).
- (3)⇒(2): Für Rückwärtskanten gilt sogar die umgekehrte Relation finishNum[v]<finishNum[w].
  Wenn (3), dann kann es also keine Rückwärtskanten geben (2).

...

DAG-Erkennung per DFS

Lemma

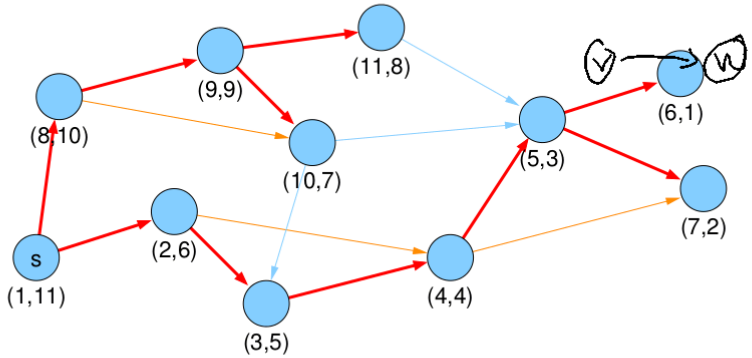
Folgende Aussagen sind äquivalent:

1. *Graph G ist ein DAG.*
2. *DFS in G enthält keine Rückwärtskante.*
3. $\forall (v,w) \in E : finishNum[v] > finishNum[w]$

DAG-Erkennung per DFS

Anwendung:

- Erkennung von azyklischen gerichteten Graphen (engl. directed acyclic graph / DAG)



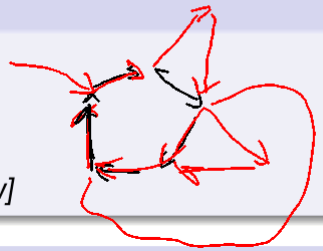
- keine gerichteten Kreise

DAG-Erkennung per DFS

Lemma

Folgende Aussagen sind äquivalent:

1. *Graph G ist ein DAG.*
2. *DFS in G enthält keine Rückwärtskante.*
3. $\forall (v,w) \in E : finishNum[v] > finishNum[w]$



Beweis.

- $\neg(2) \Rightarrow \neg(1)$: Wenn Rückwärtskante (v,w) existiert, gibt es einen gerichteten Kreis ab Knoten w (und G ist kein DAG).
- $\neg(1) \Rightarrow \neg(2)$: Wenn es einen gerichteten Kreis gibt, ist mindestens eine von der DFS besuchte Kante dieses Kreises eine Rückwärtskante (Kante zu einem schon besuchten Knoten, dieser muss Vorfahr sein).

□

Zusammenhang in Graphen

Definition

Ein ungerichteter Graph heißt zusammenhängend, wenn es von jedem Knoten einen Pfad zu jedem anderen Knoten gibt.

Ein maximaler zusammenhängender induzierter Teilgraph wird als Zusammenhangskomponente bezeichnet.

Die Zusammenhangskomponenten eines ungerichteten Graphen können mit DFS oder BFS in $O(n+m)$ bestimmt werden.

Knoten-Zusammenhang

Definition

Ein ungerichteter Graph $G = (V, E)$ heißt k-fach zusammenhängend (oder genauer gesagt k-*knoten*zusammenhängend), falls

- $|V| > k$ und
- für jede echte Knotenteilmenge $X \subset V$ mit $|X| < k$ der Graph $G - X$ zusammenhängend ist.

Knoten-Zusammenhang

Definition

Ein ungerichteter Graph $G = (V, E)$ heißt k-fach zusammenhängend (oder genauer gesagt k-*knoten*zusammenhängend), falls

- $|V| > k$ und
- für jede echte Knotenteilmenge $X \subset V$ mit $|X| < k$ der Graph $G - X$ zusammenhängend ist.

Bemerkung:

- "zusammenhängend" ist im wesentlichen gleichbedeutend mit "1-knotenzusammenhängend"

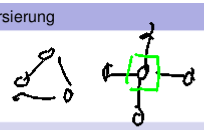
  Ausnahme: Graph mit nur einem Knoten ist zusammenhängend, aber nicht 1-zusammenhängend

Artikulationsknoten und Blöcke

Definition

Ein Knoten v eines Graphen G heißt Artikulationsknoten (engl. *cut-vertex*), wenn sich die Anzahl der Zusammenhangskomponenten von G durch das Entfernen von v erhöht.

Artikulationsknoten und Blöcke



Definition

Ein Knoten v eines Graphen G heißt Artikulationsknoten (engl. *cut-vertex*), wenn sich die Anzahl der Zusammenhangskomponenten von G durch das Entfernen von v erhöht.

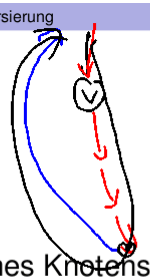
Definition

Die Zweifachzusammenhangskomponenten eines Graphen sind die maximalen Teilgraphen, die 2-fach zusammenhängend sind.

Ein Block ist ein maximaler zusammenhängender Teilgraph, der keinen Artikulationsknoten enthält.

D.h. die Menge der Blöcke besteht aus den Zweifachzusammenhangskomponenten, den Brücken (engl. *cut edges*), sowie den isolierten Knoten.

Blöcke und DFS



Modifizierte DFS nach R. E. Tarjan:

- `num[v]`: DFS-Nummer von v
- `low[v]`: minimale Nummer `num[w]` eines Knotens w, der von v aus über beliebig viele (≥ 0) Baumkanten (abwärts), evt. gefolgt von einer einzigen Rückwärtskante (aufwärts) erreicht werden kann

- `low[v]`: Minimum von
  - `num[v]`
  - `low[w]`, wobei w ein Kind von v im DFS-Baum ist (Baumkante)
  - `num[w]`, wobei $\{v, w\}$ eine Rückwärtskante ist

Artikulationsknoten und DFS

Lemma

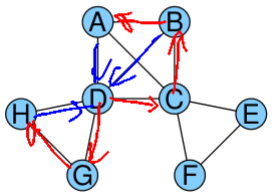
Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph und T ein DFS-Baum in G.

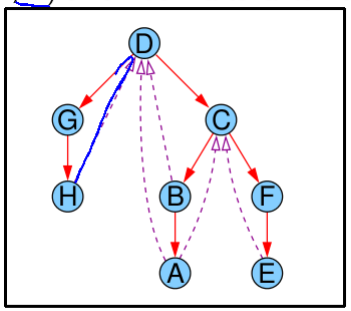
Ein Knoten $a \in V$ ist genau dann ein Artikulationsknoten, wenn

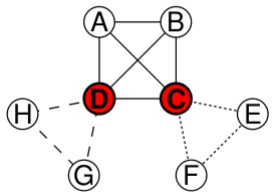
- *a die Wurzel von T ist und mindestens 2 Kinder hat, oder*
- *a nicht die Wurzel von T ist und es ein Kind b von a mit `low[b]`$\geq$`num[a]` gibt.*

Artikulationsknoten und Blöcke per DFS

- bei Aufruf der DFS für Knoten v wird `num[v]` bestimmt und `low[v]` mit `num[v]` initialisiert
- nach Besuch eines Nachbarknotens w:
  Update von `low[v]` durch Vergleich mit
  - `low[w]` nach Rückkehr vom rekursiven Aufruf, falls (v, w) eine Baumkante war
  - `num[w]`, falls (v, w) eine Rückwärtskante war

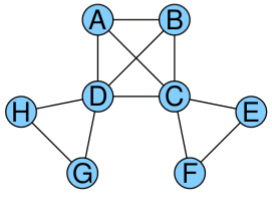


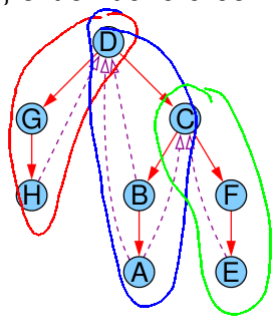


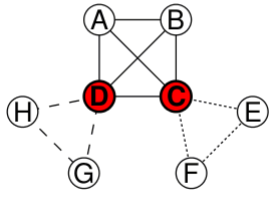


Artikulationsknoten und Blöcke per DFS

- Kanten werden auf einem anfangs leeren Stack gesammelt
- Rückwärtskanten kommen direkt auf den Stack (ohne rek. Aufruf)
- Baumkanten kommen vor dem rekursiven Aufruf auf den Stack
- nach Rückkehr von einem rekursiven Aufruf werden im Fall `low[w]`$\geq$`num[v]` die obersten Kanten vom Stack bis einschließlich der Baumkante $\{v, w\}$ entfernt und bilden den nächsten Block







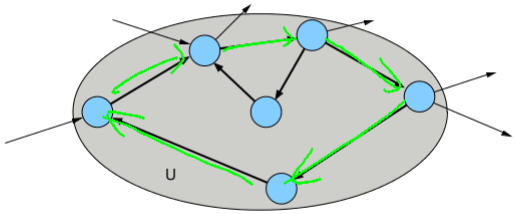
Starke Zusammenhangskomponenten

Definition

Sei $G = (V, E)$ ein gerichteter Graph.

Knotenteilmenge $U \subseteq V$ heißt stark zusammenhängend genau dann, wenn für alle $u, v \in U$ ein gerichteter Pfad von u nach v in G existiert.

Für Knotenteilmenge $U \subseteq V$ heißt der induzierte Teilgraph $G[U]$ starke Zusammenhangskomponente von G, wenn U stark zusammenhängend und (inklusions-)maximal ist.



Starke Zusammenhangskomponenten

Beobachtungen:

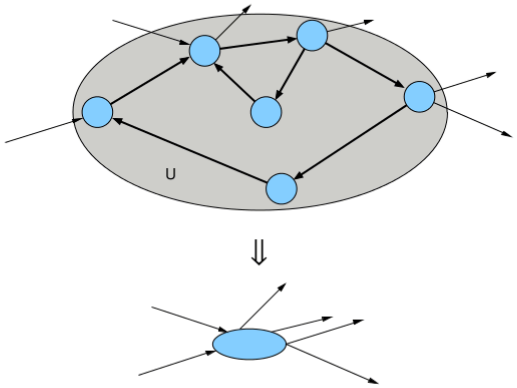
- Knoten $x, y \in V$ sind stark zusammenhängend, falls beide Knoten auf einem gemeinsamen gerichteten Kreis liegen (oder $x = y$).
- Die starken Zusammenhangskomponenten bilden eine Partition der Knotenmenge.

  (im Gegensatz zu 2-Zhk. bei ungerichteten Graphen, wo nur die Kantenmenge partitioniert wird, sich aber zwei verschiedene 2-Zhk. in einem Knoten überlappen können)

Starke Zusammenhangskomponenten

Beobachtungen:

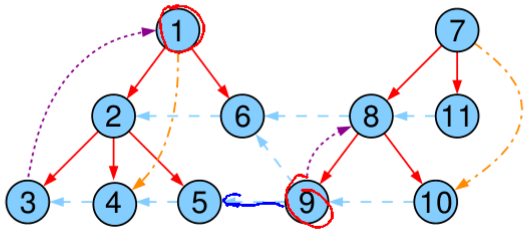
- Schrumpft man alle starken Zusammenhangskomponenten zu einzelnen (Super-)Knoten, ergibt sich ein DAG.



Starke Zhk. und DFS / Variante 1

Modifizierte DFS nach R. E. Tarjan

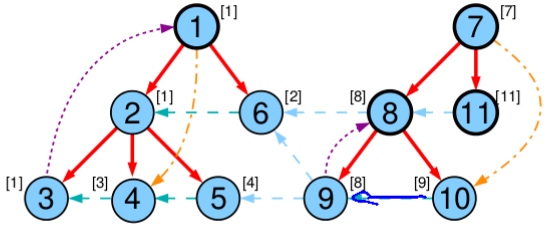
- `num[v]`: DFS-Nummer von v
- In einer starken Zhk. heißt der Knoten mit kleinster DFS-Nummer Wurzel der starken Zhk.
- `low[v]`: minimales `num[w]` eines Knotens w, der von v aus über beliebig viele (≥ 0) Baumkanten abwärts, evt. gefolgt von einer einzigen Rückwärtskante oder einer Querkante zu einer ZHK, deren Wurzel echter Vorfahre von v ist, erreicht werden kann



Starke Zhk. und DFS / Variante 1

Modifizierte DFS nach R. E. Tarjan

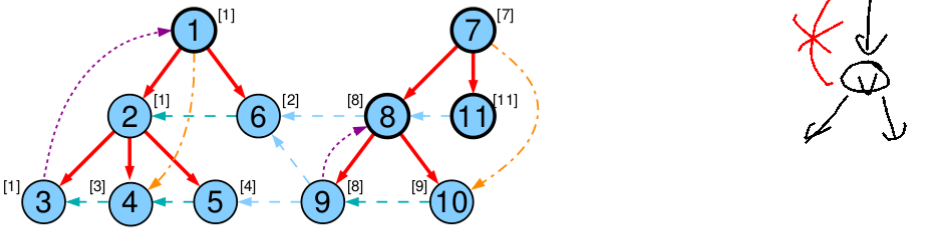
- `low[v]`: Minimum von
  - `num[v]`
  - `low[w]`, wobei w ein Kind von v im DFS-Baum ist (Baumkante)
  - `num[w]`, wobei $\{v, w\}$ eine Rückwärtskante ist
  - `num[w]`, wobei $\{v, w\}$ eine Querkante ist und die Wurzel der starken Zusammenhangskomponente von w ist Vorfahre von v

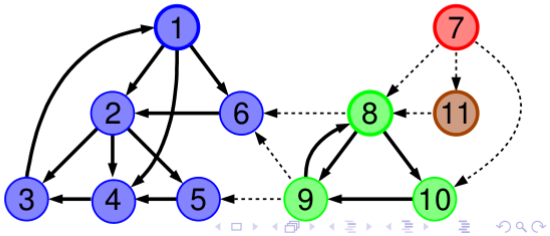


Starke Zhk. und DFS / Variante 1

Modifizierte DFS nach R. E. Tarjan

- Knoten v ist genau dann Wurzel einer starken Zusammenhangskomponente, wenn `num[v]=low[v]`





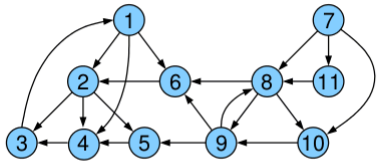
Starke Zhk. und DFS / Variante 2

Idee:

- beginne mit Graph ohne Kanten, jeder Knoten ist eigene SCC
- füge nach und nach einzelne Kanten ein

$\Rightarrow$ aktueller (current) Graph $G_c = (V, E_c)$

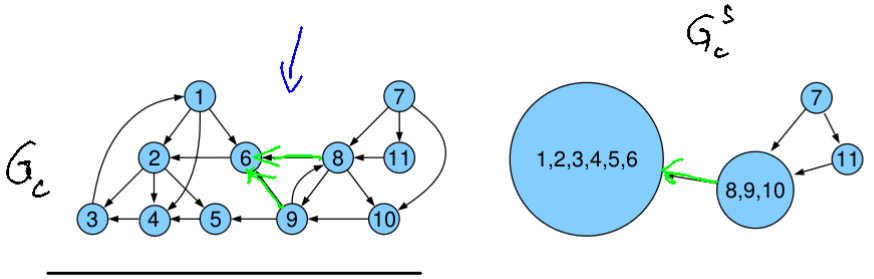
- Update der starken Zusammenhangskomponenten (SCCs)



Starke Zhk. und DFS / Variante 2

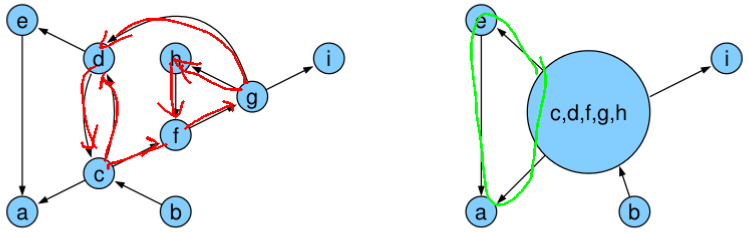
Idee:

- betrachte geschrumpften (shrunken) Graph G_c^s:
  Knoten entsprechen SCCs von G_c, Kante (C, D) genau dann, wenn es Knoten $u \in C$ und $v \in D$ mit $(u, v) \in E_c$ gibt
- geschrumpfter Graph G_c^s ist ein DAG
- Ziel: Aktualisierung des geschrumpften Graphen beim Einfügen



Starke Zhk. und DFS / Variante 2

Geschrumpfter Graph
(Beispiel aus Mehlhorn / Sanders)



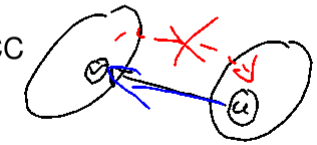
Starke Zhk. und DFS / Variante 2

Update des geschrumpften Graphen nach Einfügen einer Kante:

3 Möglichkeiten:

- beide Endpunkte gehören zu derselben SCC

⇒ geschrumpfter Graph unverändert



- Kante verbindet Knoten aus zwei verschiedenen SCCs, aber schließt keinen Kreis

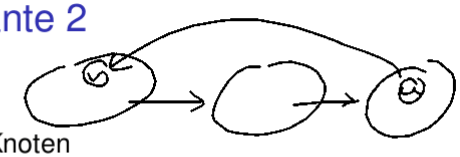
⇒ SCCs im geschrumpften Graph unverändert, aber eine Kante wird im geschrumpften Graph eingefügt (falls nicht schon vorhanden)

- Kante verbindet Knoten aus zwei verschiedenen SCCs und schließt einen oder mehrere Kreise

⇒ alle SCCs, die auf einem der Kreise liegen, werden zu einer einzigen SCC verschmolzen

Starke Zhk. und DFS / Variante 2

Prinzip:



- Tiefensuche
  V_c schon markierte (entdeckte) Knoten
  E_c schon gefundene Kanten
- 3 Arten von SCC: unentdeckt, offen, geschlossen
- unentdeckte Knoten haben Ein- / Ausgangsgrad Null in G_c

⇒ zunächst bildet jeder Knoten eine eigene unentdeckte SCC, andere SCCs enthalten nur markierte Knoten

- SCCs mit mindestens einem aktiven Knoten (ohne finishNum) heißen offen
- SCC heißt geschlossen, falls sie nur fertige Knoten (mit finishNum) enthält
- Knoten in offenen / geschlossenen SCCs heißen offen / geschlossen