

Grundlagen: Algorithmen und Datenstrukturen

Prof. Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Sommersemester 2010

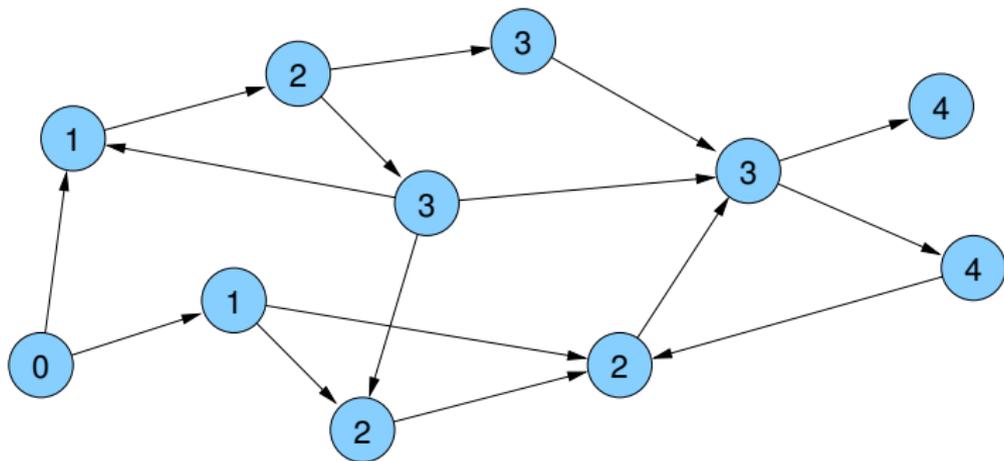


Übersicht

- 1 Graphen
 - Graphtraversierung

Breitensuche

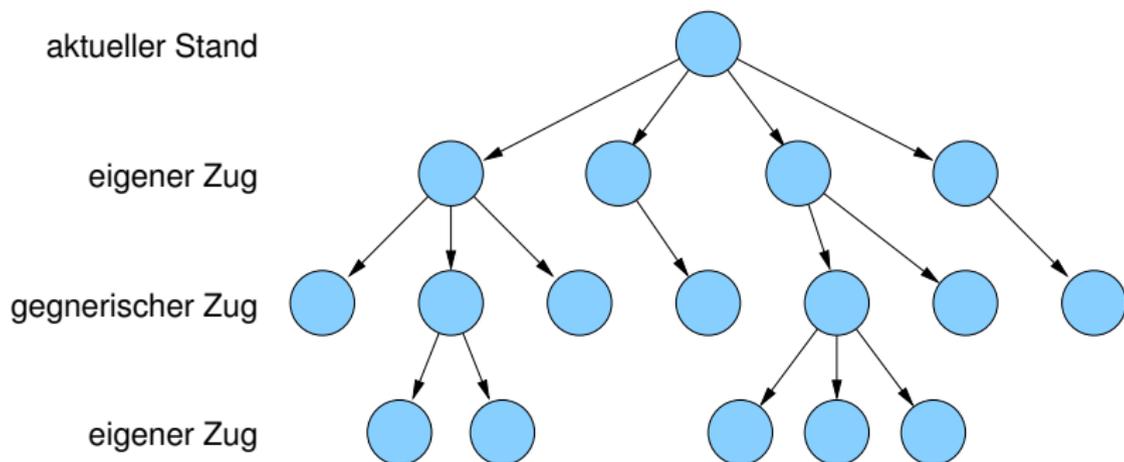
Anwendung: Single Source Shortest Path (SSSP) Problem
in **ungewichteten** Graphen



Breitensuche

Anwendung: Bestimmung des nächsten Zuges bei Spielen

Exploration des Spielbaums



Problem: halte Aufwand zur Suche eines guten Zuges in Grenzen

Breitensuche

Anwendung: Bestimmung des nächsten Zugs bei Spielen

- **Standard-BFS:** verwendet **FIFO-Queue**
ebenenweise Erkundung
aber: zu teuer!
- **Best-First Search:** verwendet **Priority Queue**
(z.B. realisiert durch binären Heap)
Priorität eines Knotens wird durch eine Güte-Heuristik des repräsentierten Spielzustands gegeben

Tiefensuche

Übergeordnete Methode (falls nicht alle Knoten erreicht werden)

alle Knoten nicht markiert;

init();

foreach ($s \in V$)

if (s nicht markiert) {

markiere s;

root(s);

DFS(s,s);

}

Tiefensuche

```
void DFS(Node u, Node v) {  
    foreach ((v, w) ∈ E)  
        if (is_marked(w))  
            traverseNonTreeEdge(v,w);  
        else {  
            traverseTreeEdge(v,w);  
            mark(w);  
            DFS(v,w);  
        }  
    backtrack(u,v);  
}
```

Tiefensuche

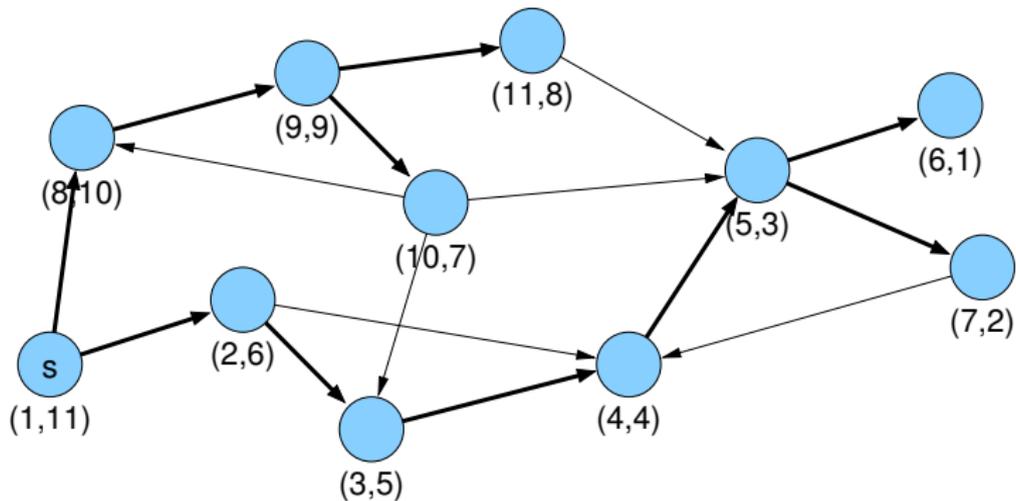
Variablen:

- `int[] dfsNum;` // Explorationsreihenfolge
- `int[] finishNum;` // Fertigstellungsreihenfolge
- `int dfsCount, finishCount;` // Zähler

Methoden:

- `void init() { dfsCount = 1; finishCount = 1; }`
- `void root(Node s) { dfsNum[s] = dfsCount; dfsCount++; }`
- `void traverseTreeEdge(Node v, Node w)`
`{ dfsNum[w] = dfsCount; dfsCount++; }`
- `void traverseNonTreeEdge(Node v, Node w) { }`
- `void backtrack(Node u, Node v)`
`{ finishNum[v] = finishCount; finishCount++; }`

Tiefensuche



DFS-Nummerierung

Lemma

Die Knoten im DFS-Rekursionsstack (die aktiven Knoten) sind bezüglich dfsNum sortiert.

Beweis.

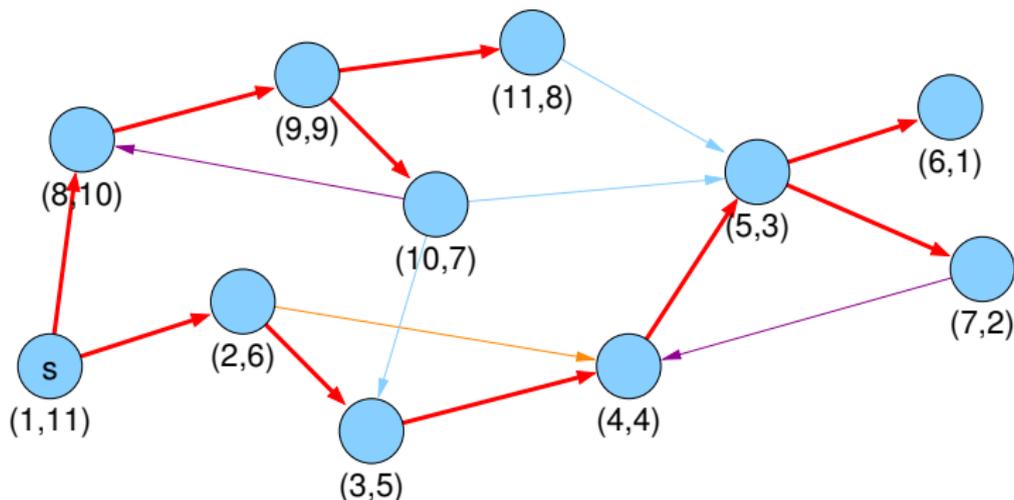
- dfsCount wird nach jeder Zuweisung von dfsNum inkrementiert
- neue aktive Knoten haben also immer die höchste dfsNum



DFS-Nummerierung

Kantentypen:

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



DFS-Nummerierung

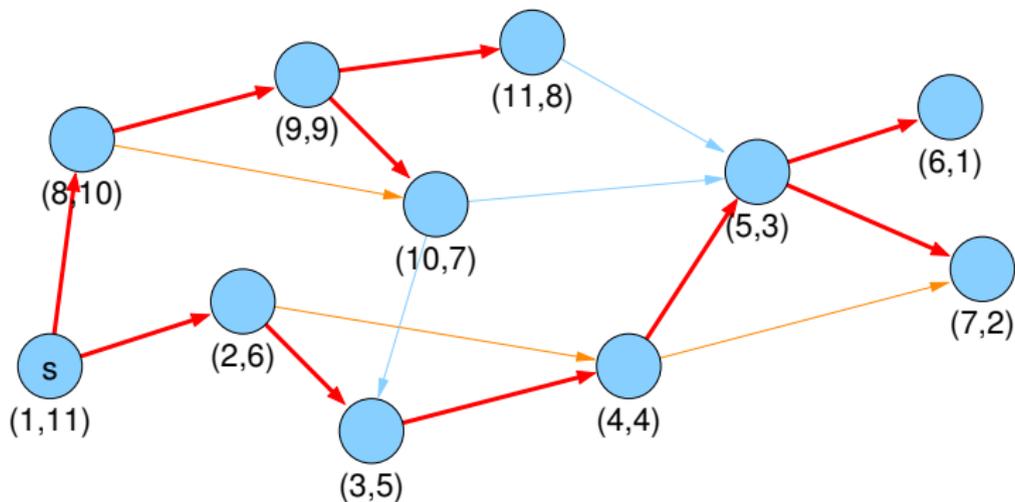
Beobachtung für Kante (v, w) :

Kantentyp	$\text{dfsNum}[v] < \text{dfsNum}[w]$	$\text{finishNum}[v] > \text{finishNum}[w]$
Baum & Vorwärts	ja	ja
Rückwärts	nein	nein (umgekehrt)
Kreuz	nein	ja

DFS-Nummerierung

Anwendung:

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



- keine gerichteten Kreise

DFS-Nummerierung

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

...

DFS-Nummerierung

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 -knotenzusammenhängend**), falls

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

Bemerkung:

- einzelne isolierte Knoten sind zwar 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.

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 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

Blöcke und DFS

Lemma

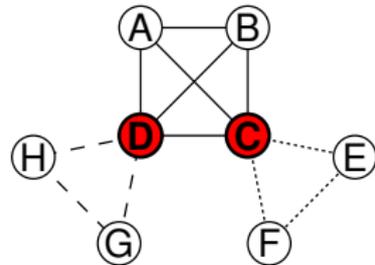
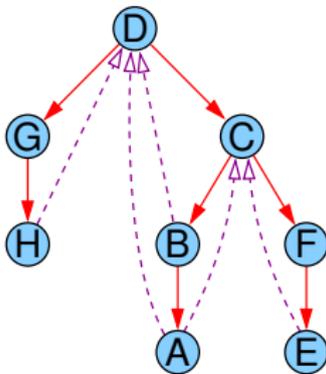
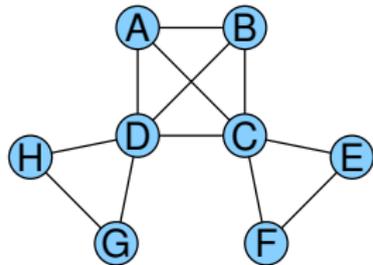
Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph und T ein DFS-Baum in G .

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.

Blöcke und DFS

Die Kanten werden auf einem Stack gesammelt und nach der Erkennung eines Artikulationsknotens wird der gesamte Block abgeplückt.



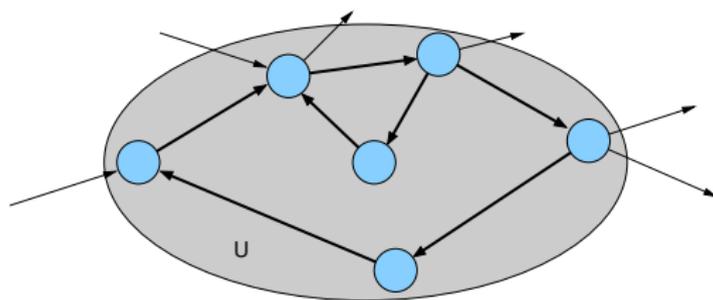
Starke Zusammenhangskomponenten

Definition

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

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

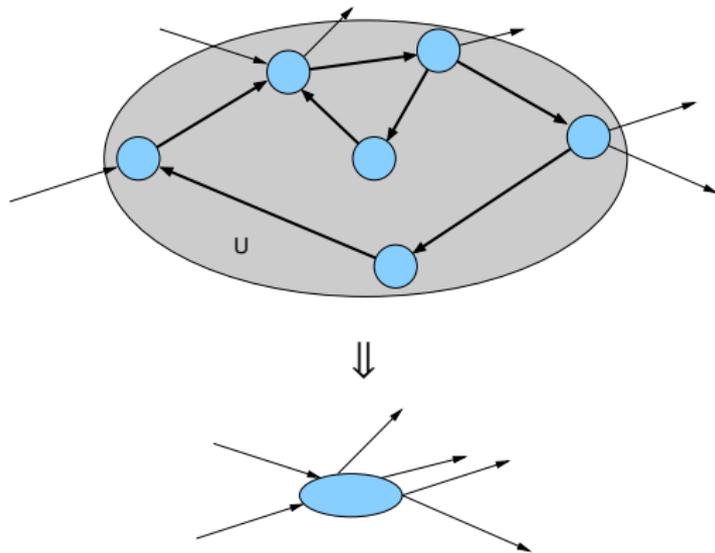
$U \subseteq V$ heißt **starke Zusammenhangskomponente** (ZHK) von G , wenn U stark zusammenhängend und (inklusions-)maximal ist



Starke Zusammenhangskomponenten

Beobachtung:

Schrumpft man alle starken Zusammenhangskomponenten zu einzelnen Knoten, ergibt sich ein DAG.



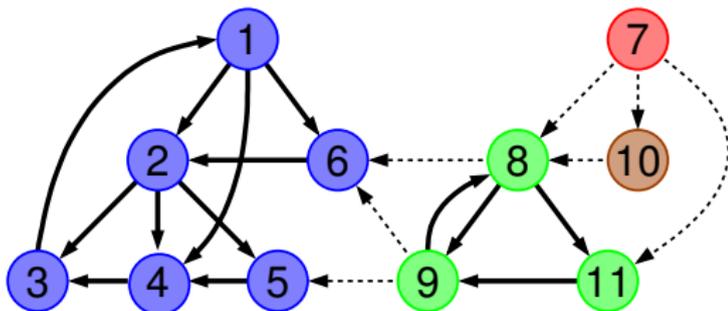
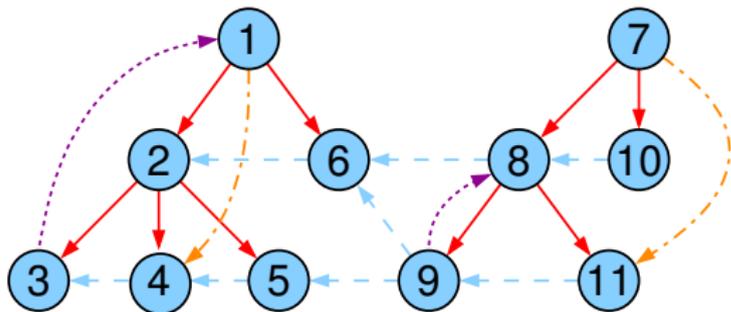
Starke Zhk. und DFS / Variante 1

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 oder einer Querkante zu einer ZHK, deren Wurzel echter Vorfahre von v ist, 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
 - ▶ **num**[w], wobei $\{v, w\}$ eine **Querkante** ist und die Wurzel der starken Zusammenhangskomponente von w ist Vorfahre von v

Starke Zusammenhangskomponenten

- Knoten v ist genau dann Wurzel einer starken Zusammenhangskomponente, wenn $\text{num}[v] = \text{low}[v]$



Starke Zhk. und DFS

Prinzip:

- Unterscheidung in **fertige** / **unfertige** Knoten
- Zusammenhangskomponente heißt **geschlossen**, falls sie nur fertige Knoten enthält, ansonsten heißt sie **offen**
- **Repräsentant** einer ZHK ist der Knoten mit der kleinsten dfsNum

Starke Zhk. und DFS

Beobachtungen (Invarianten):

- Pfade aus **geschlossenen** Komponenten führen immer zu **fertigen** Knoten
- Der Pfad vom Startknoten zum aktuellen Knoten enthält die **Repräsentanten** aller **offenen** ZHKs.
- Betrachtet man die Knoten in offenen ZHKs sortiert nach DFS-Nummern, so **partitionieren** die Repräsentanten diese Folge in die offenen ZHKs.