Informatik II: Algorithmen und Datenstrukturen SS 2013

Vorlesung 10a, Dienstag, 25. Juni 2013

(Graphen, Breitensuche, Tiefensuche, Zusammenhangskomponenten)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Ihre Erfahrungen mit dem Ü9 (Prioritätswürgeschlangen)
- Graphenalgorithmen, Teil 1
  - Graphenmodelle: Adjazenzmatrix, Adjazenzlisten
  - Breitensuche / Breadth First Search (BFS)
  - Tiefensuche / Depth First Search (DFS)
  - Zusammenhangskomponenten / Connected Components (CC)
  - Übungsblatt 10, Aufgabe 1: Zeigen, dass BFS kürzeste Wege berechnet + kann DFS das auch ?

Erfahrungen mit dem Ü9 (Würgeschlangen)

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 25. Juni 16:00
  - Vorlesungen sind in den letzten Wochen zunehmend einfacher geworden
  - Aufgabe 2: wie formal? Das meiste wurde schon in der Vorlesung gemacht; einer meinte: nutzlos
  - Checkstyle Fehler bei so was wie items.get(1).heapIndex = 1
  - Aus Dickköpfigkeit nicht im Forum gefragt … das trainiert die Frustrationstoleranz
  - Keine Zeit, keine Zeit, keine Zeit
  - Sommersonnenwende war am 21. Juni um 7:04 Uhr

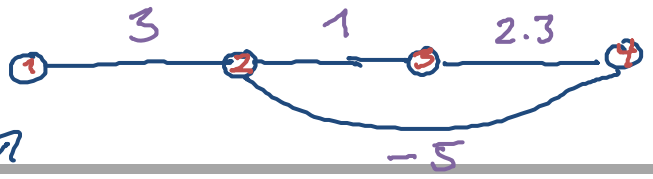
    "eine meiner Lieblingsuhrzeiten, gleich nach 17:46 Uhr"

Graphen 1/6

UNI FREIBURG

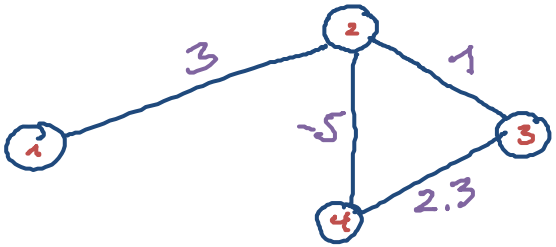
- Definition:
  - Ein Graph G besteht aus einer Menge V von Knoten ...
    - Englisch: vertices (daher V) oder nodes
  - ... und einer Menge E von Kanten
    - English: edges (daher E) oder arcs
  - Eine Kante e verbindet jeweils zwei Knoten u und v
    - ungerichtete Kante: $e = \{u, v\}$ (Menge)
    - gerichtete Kante: $e = (u, v)$ (Tupel)
  - **Gewichteter** Graph
    - Eine reelle Zahl pro Kante, das sogenannte Gewicht der Kante, je nach Anwendung auch Länge oder Kosten der Kante genannt

Graphen 2/6

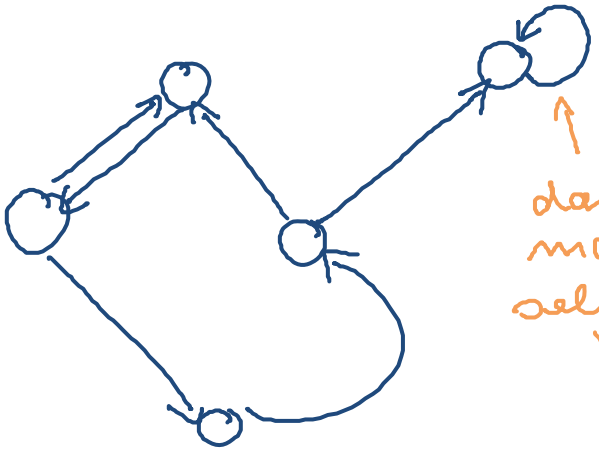


UNI FREIBURG

- Beispiele



ungerichteter Graph mit Kantenkosten mit 4 Knoten und 4 Kanten



das nennt man eine self-loop

gerichteter Graph ohne Kantenkosten

Graphen 3/6

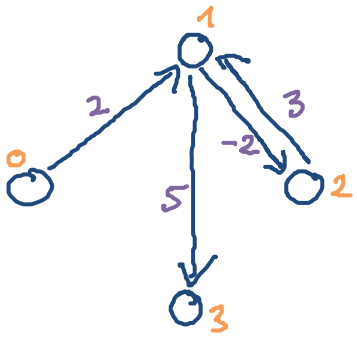
bei unger. Graphen speichert man jede Kante i.d.R. in beide Richtungen ab.

- Wie repräsentiert man Graphen im Rechner
  - Da gibt es zwei klassischen Arten

    Adjazenzmatrix ... Platzverbrauch $\Theta(|V|^2)$

    Adjazenzlisten bzw. **–felder** ... Platzverbrauch $\Theta(|V| + |E|)$

gerichtet. Graph mit 4 Knoten und 4 Kanten mit Kantenkosten



Einträge + x ist gerade |E|

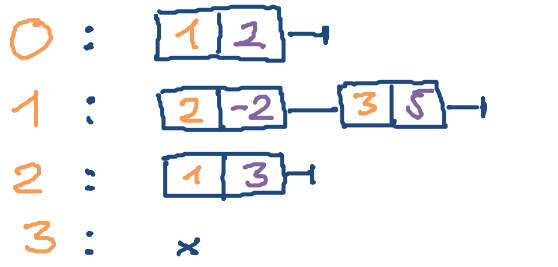
Adjazenz-matrize

	0	1	2	3
0	x	2	x	x
1	x	x	-2	5
2	x	3	x	x
3	x	x	x	x

x = keine Kante

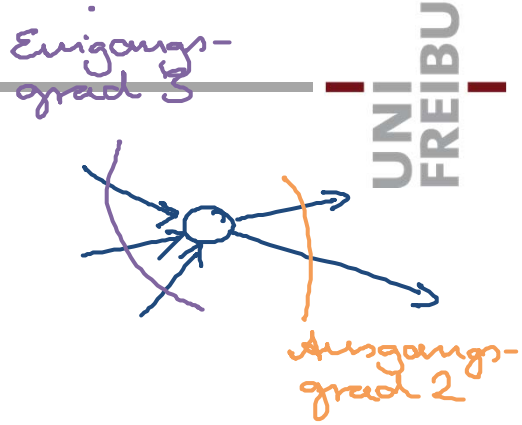
bei ungerichtetem Graph ist diese Matrix SYMMETRISCH

Adjazenzlisten



Anzahl Einträge ist gerade |E|

Graphen 4/6

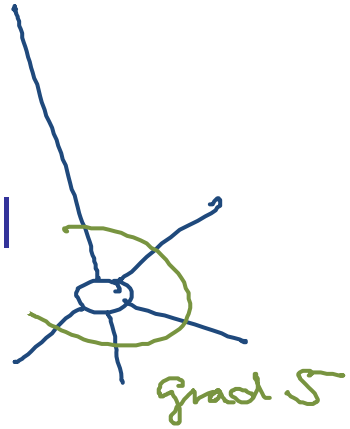


- Grade in einem Graphen $G = (V, E)$
  - Falls gerichtet
    - **Eingangsgrad** von einem Knoten u

      = Anzahl eingehender Kanten = $|(v,u) : (v,u) \in E|$
    - **Ausgangsgrad** von einem Knoten u

      = Anzahl ausgehender Kanten = $|(u,v) : (u,v) \in E|$
  - Falls ungerichtet
    - **Grad** von einem Knoten u

      = Anzahl adjazenter Kanten = $|\{u,v\} : \{u,v\} \in E|$



Graphen 5/6

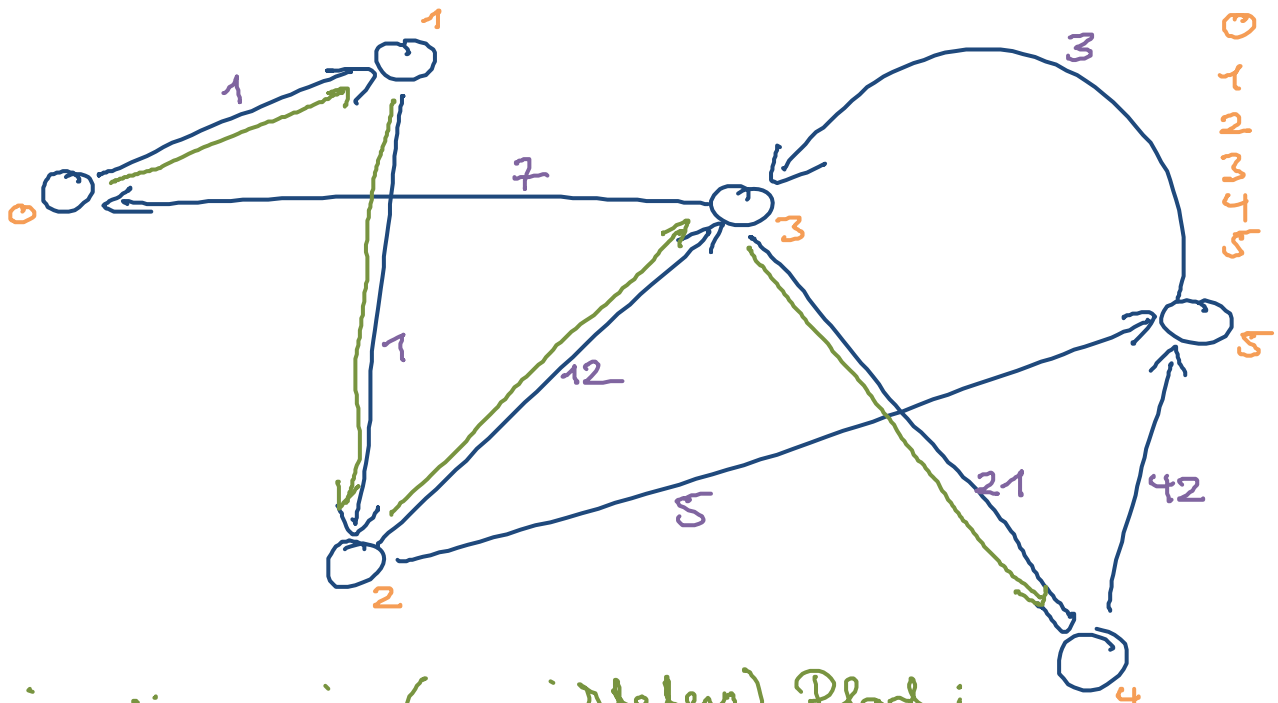
UNI FREIBURG

- Pfade in einem Graphen $G = (V, E)$
  - Ein Pfad in G ist eine Folge $u_1, u_2, u_3, \ldots, u_l \in V$ mit
    - $(u_1,u_2), (u_2,u_3), \ldots, (u_{l-1},u_l) \in E$ [gerichteter Graph]
    - $\{u_1,u_2\}, \{u_2,u_3\}, \ldots, \{u_{l-1},u_l\} \in E$ [ungerichteter Graph]
  - Die **Länge des Pfades** (auch: Kosten des Pfades)
    - ohne Kantengewichte: Anzahl der Kanten
    - mit Kantengewichte: Summe der Gewichte auf dem Pfad
  - Der **kürzeste Pfad** (engl. shortest path) zwischen zwei Knoten u und v ist der Pfad $u, \ldots, v$ mit der kürzesten Länge
  - Der **Durchmesser** eines Graphen ist der längste kürzeste Pfad = $\max_{u,v}${Länge von P : P ist ein kürzester Pfad zwischen u und v}

Graphen 6/6

Durchmesser der ungerichteten Version dieses Graphen (= alle Kanten in beide Richtungen): 29

- Beispiel Pfade und Durchmesser



kürz. Pfade

	0	1	2	3	4	5
0	×	1	2	7	28	7
1		×	1	8	29	6
2			×	8	29	5
3				×	21	3
4					×	24
5						×

symm.

Beispiel für einen (gerichteten) Pfad:

$(0,1)^{1}$ $(1,2)^{1}$ $(2,3)^{12}$ $(3,4)^{21}$

Kosten des Pfades: $1+1+12+21 = 35$

Graphexploration

UNI FREIBURG

- Informale Definition
  - Gegeben ein Graph $G = (V, E)$ und ein Startknoten $s \in V$, besuche "systematisch" alle Knoten von V, die von s aus erreichbar sind
  - Breitensuche = in der Reihenfolge der "Entfernung" von s

    Englisch: breadth first search = BFS
  - Tiefensuche = erstmal "möglichst weit weg" von s

    Englisch: depth first search = DFS
  - Das ist kein "Problem" an sich, taucht aber oft als Teil / Subroutine von anderen Algorithmen auf

    Zum Beispiel zur Berechnung der Zus.hangskomponenten eines Graphen, siehe Folien am Ende

Breitensuche (BFS) 1/2

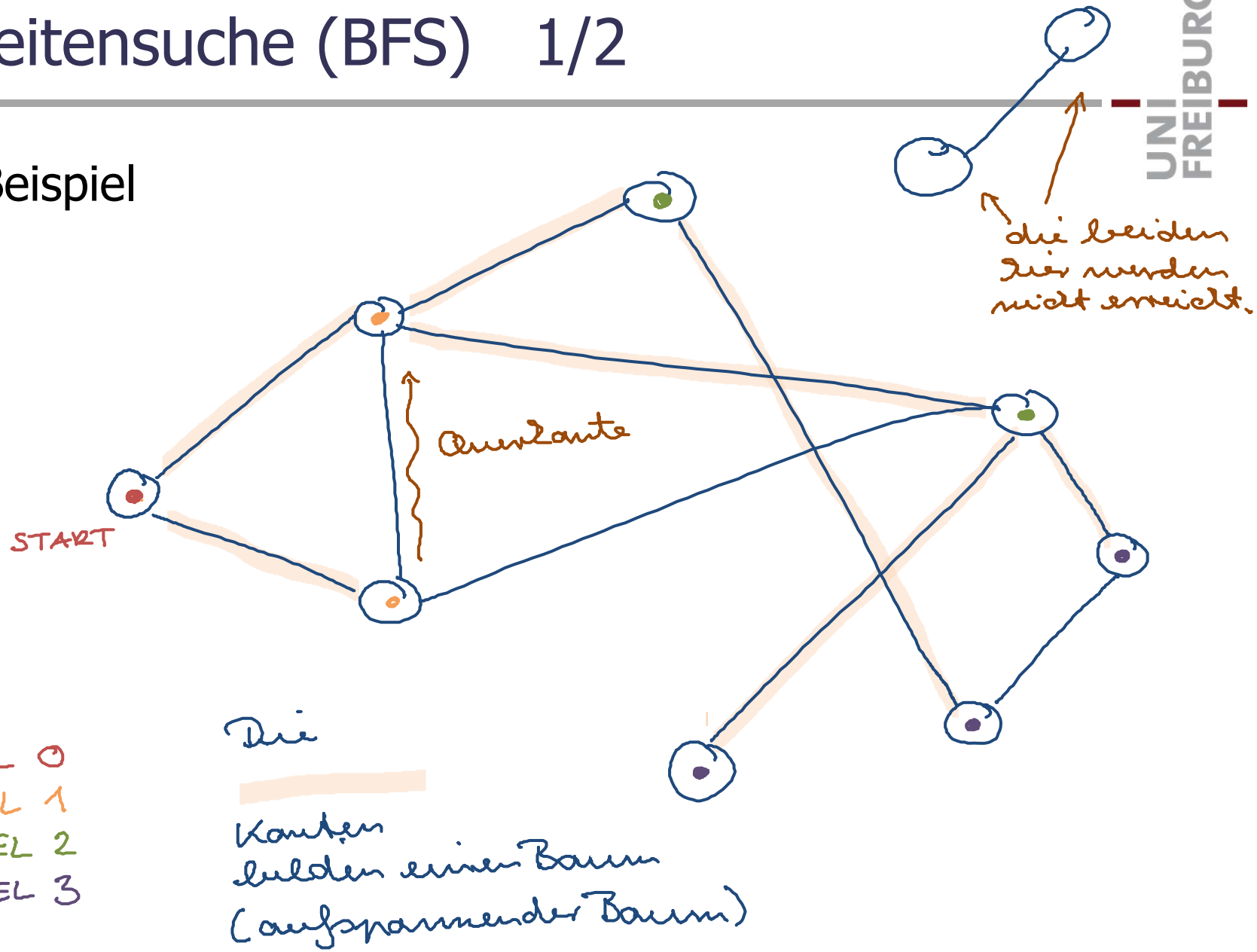
UNI FREIBURG

- Idee
  - Markierung für jeden Knoten, zu Beginn alle unmarkiert
  - Beginne mit einem Startknoten und markiere ihn (Level 0)
  - Finde alle Knoten die zum Startknoten benachbart und noch nicht markiert sind und markiere sie (Level 1)
  - Finde alle Knoten, die zu einem Level 1 Knoten benachbart und noch nicht markiert sind und markiere sie (Level 2)
  - Usw. bis ein Level keine benachbarten Knoten mehr hat, die noch nicht markiert sind

    Das markiert insbesondere alle Knoten, die in derselben Zusammenhangskomponente sind wie der Startknoten

Breitensuche (BFS) 1/2

- Beispiel



Tiefensuche (DFS) 1/2

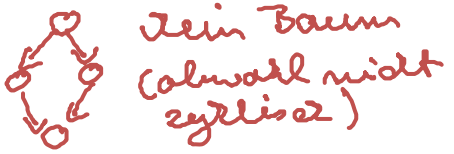
UNI FREIBURG

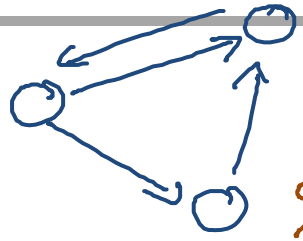
- Idee
  - Markierung für jeden Knoten, zu Beginn alle unmarkiert
  - Beginne mit einem Startknoten und markiere ihn
  - Gehe in irgendeiner Reihenfolge die zum Startknoten benachbarten Knoten durch und tue Folgendes:

    Falls der Knoten noch nicht markiert ist, markiere ihn und starte **rekursiv** eine Tiefensuche von dort aus
  - Das sucht zuerst "in die Tiefe" (vom Startknoten aus)
  - Auch DFS markiert schließlich alle Knoten, die in derselben Zusammenhangskomponenten liegen wie der Startknoten
  - Auf azyklischen Graphen liefert DFS **topologische Sortierung**

    Das ist eine Nummerierung der Knoten, so dass jede Kante von einem Knoten mit kleinerer Nummer zu einem mit größerer Nummer geht

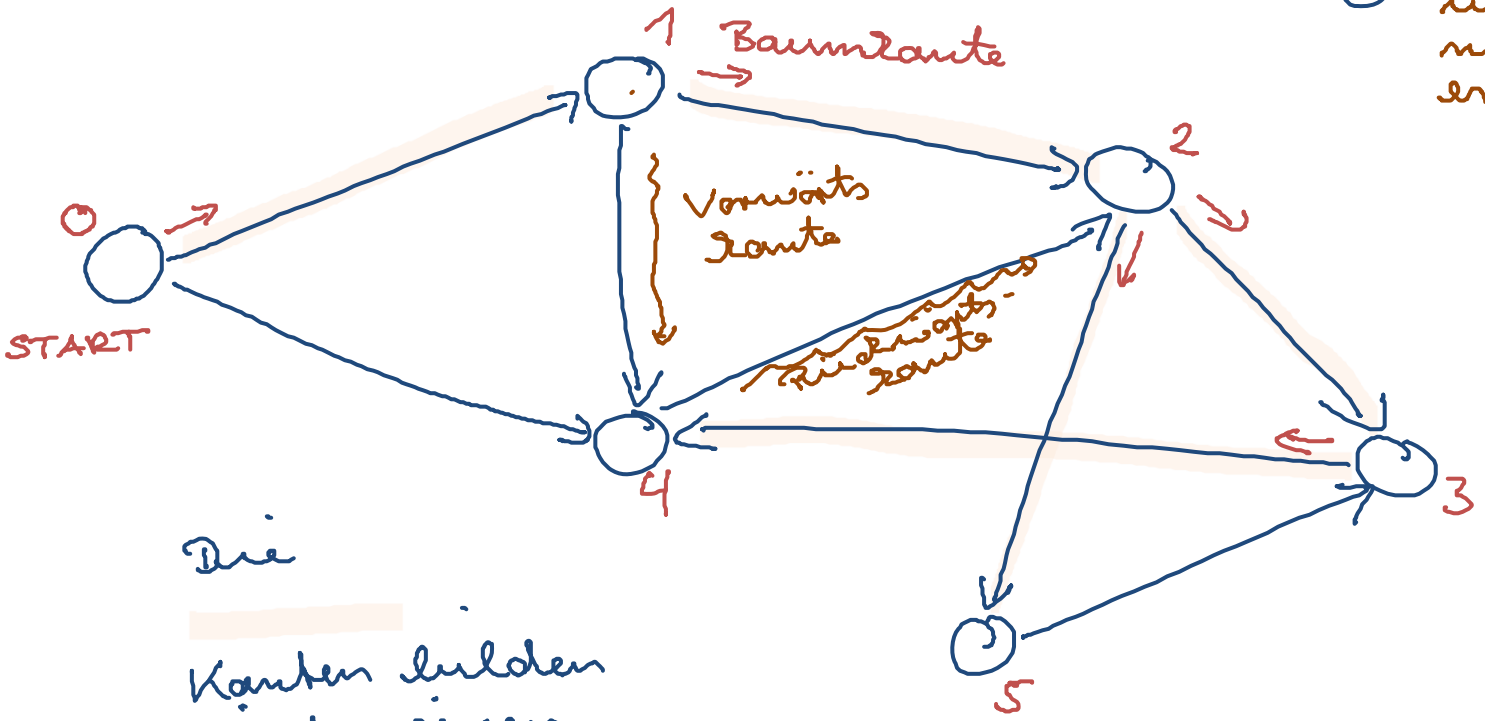
Tiefensuche (DFS) 2/2





die
werden
nicht
erreicht

- Beispiel



Komplexität von BFS und DFS

UNI FREIBURG

- Für beide Verfahren gilt:
  - Konstante Arbeit für jeden Knoten und jede Kante
  - Die Laufzeit ist also genau $\Theta(|V'| + |E'|)$

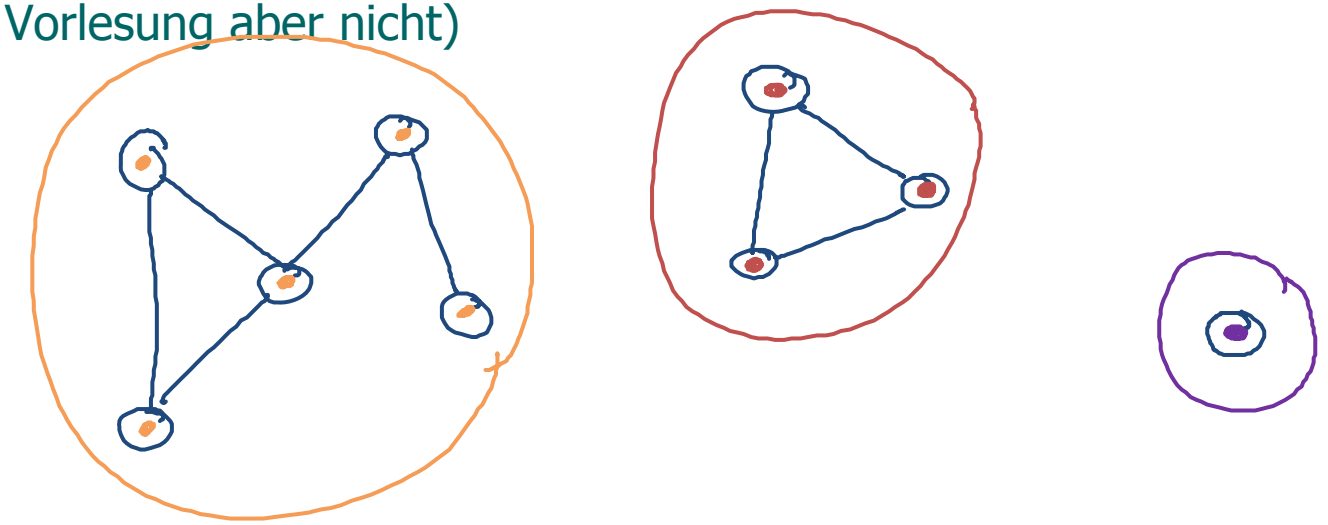
    wobei V' und E' gerade die Menge aller Knoten und Kanten in der Zusammenhangskomponente sind, in der der Startknoten liegt
  - Das kann man also (bis auf einen konstanten Faktor) nicht besser machen

Zusammenhangskomponenten 1/2

UNI FREIBURG

- Für einen **ungerichteten** Graphen $G = (V,E)$
  - Die Zusammenhangskomponenten bilden eine Partition von V, also $V = V_1 \cup \ldots \cup V_k$
  - Zwei Knoten u und v sind in derselben Zusammenhangskomponente, wenn es einen Pfad zwischen u und v gibt

(Für **gerichtete** Graphen ist die Definition komplizierter, man spricht dann von **starken** Zusammenhangskomponenten, das machen wir in dieser Vorlesung aber nicht)



Zusammenhangskomponenten 2/2

UNI FREIBURG

- Berechnung durch DFS oder BFS
  - Man speichert eine Markierung für jeden Knoten, wobei am Anfang alle Knoten unmarkiert sind
  - Dann wiederholt man das Folgende, so lange es noch unmarkierte Knoten gibt:

    Nimm einen beliebigen unmarkierten Knoten x (am Anfang: irgendeinen) und markiere ihn

    Starte DFS oder BFS von x und finde alle von x aus erreichbaren Knoten und markiere sie

    Die Reihenfolge, in der die Knoten besucht werden ist hier egal, deswegen auch hier egal of DFS oder BFS

    Diese Knoten bilden genau eine Zus.hangskomponente

Literatur / Links

- Graphen
  - In Mehlhorn/Sanders:

    8 Graph Representation
  - In Wikipedia

    http://en.wikipedia.org/wiki/Graph_(mathematics)
- Breitensuche, Tiefensuche, Z-Komponenten
  - In Mehlhorn/Sanders:

    9 Graph Traversal
  - In Wikipedia

    http://en.wikipedia.org/wiki/Breadth-first_search

    http://en.wikipedia.org/wiki/Depth-first_search

    http://en.wikipedia.org/wiki/Connected_component