Algorithmen und Datenstrukturen (für ESE) WS 2011 / 2012

Vorlesung 11, Dienstag, 24. Januar 2012
(Graphen, Breitensuche, Tiefensuche)

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 Ü10 (a,b-Bäume)
- Graphen
  - Neben Feldern, Listen und Bäumen die häufigste Datenstruktur
    - Bäume sind eine spezielle Art von Graph
  - Darstellung im Rechner
  - Ein paar algorithmische Probleme auf Graphen
  - **Breitensuche** und **Tiefensuche** im Detail
  - **Übungsaufgabe (Ü11):**
    - Breitensuche **oder** Tiefensuche implementieren **...** und damit auf einem zufälligen Graphen den Durchmesser berechnen

Ihre Erfahrungen mit dem Ü10 (a,b-Baum)

UNI FREIBURG

- Zusammenfassung von Ihrem Feedback **Stand 24.1. 15:53**
  - Vorlesung war informativ und verständlich, viele Beispiele
  - Nicht ganz so nerviger Beweis, hat fast Spaß gemacht
  - Insbesondere die tschechische Wikipedia-Seite hat geholfen
  - Bei Beweisen nie sicher ob alles richtig ist / kein Compiler
    Beim Programmieren merkt man ob es läuft
  - Welche Fälle alles? Antworten im Forum verwirrend
  - Nicht das Gefühl beim Beweisen was zu lernen
  - Übungsblatt war viel Wirbel um nichts
  - Programmieren macht einfach mehr Spaß
  - Tutorentreffen hat sehr viel gebracht
  - Infos zu Klausuraufgaben wäre toll
  - Großes Lob zur Aufzeichnung. Top!
  - Lob für guten und zeitnahen Support im Forum

Kritik

- Sie können gerne Kritik üben
  - Wir bitten sogar darum und nehmen Sie sehr ernst
  - Aber Kritik üben will gelernt sein
  - Ein sehr wichtiger "soft skill" im Leben
  - Bleiben Sie sachlich und bleiben Sie bei sich

    "Die Erklärung war total Scheiße"

    "Ich habe nicht verstanden, warum ..."

    "Ich glaube da stimmt was nicht: ..."
  - Übermäßig politisch korrekt aber bitte auch nicht

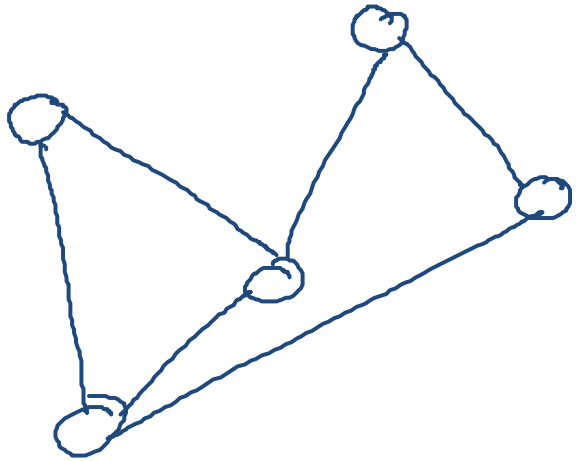
    "Das macht mich jetzt echt ganz betroffen, dass die Erklärung total Scheiße ... äh, dass ich die Erklärung nicht verstanden habe"
  - Wenn Sie sauer sind ... trinken Sie erstmal einen Tee

Graphen

- 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

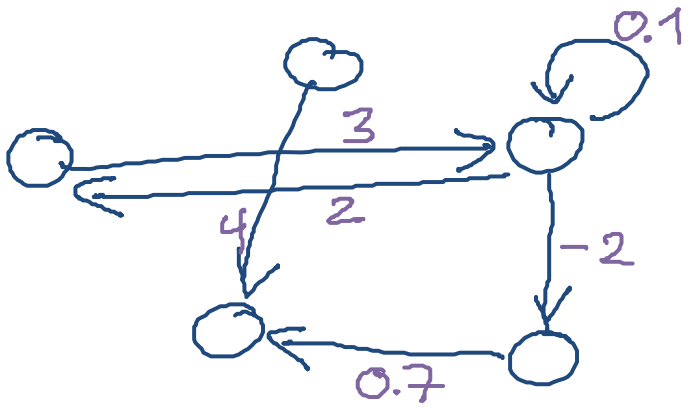
Graphen — Beispiele

ungerichteter Graph:



5 Knoten
6 Kanten

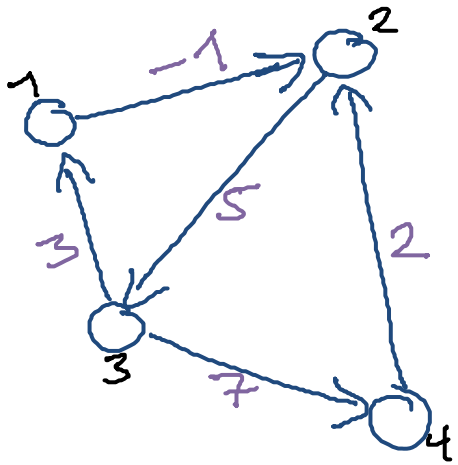
gerichteter Graph:



5 Knoten
6 Kanten
Gerichtet

Graphen — Repräsentation im Rechner

- Zwei klassische Arten
  - Adjazenzmatrix
    - Platzverbrauch ~ $|V|^2$
  - Adjazenzlisten bzw. -felder
    - Platzverbrauch ~ $|V| + |E|$
- Beispiel



Knotennummern
Gewichte

Adjazenz-matrix:

	1	2	3	4
1	x	-1	x	x
2	x	x	5	x
3	3	x	x	7
4	x	2	x	x

Adjazenz-listen:

1 : (2,-1)
2 : (3,5)
3 : (1,3) ; (4,7)
4 : (2,2)

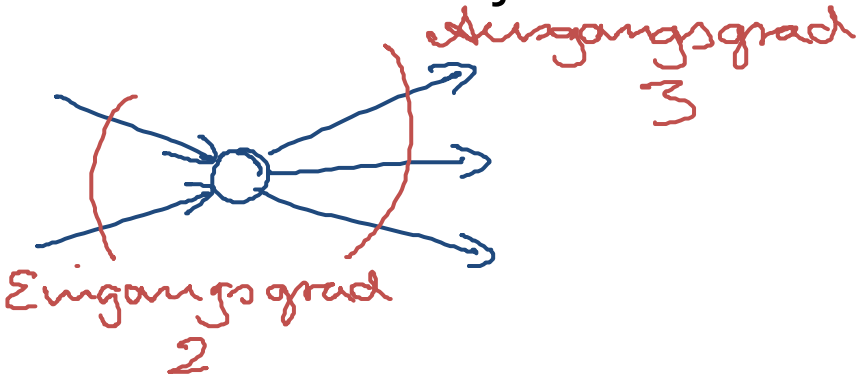
Graphen — Grad

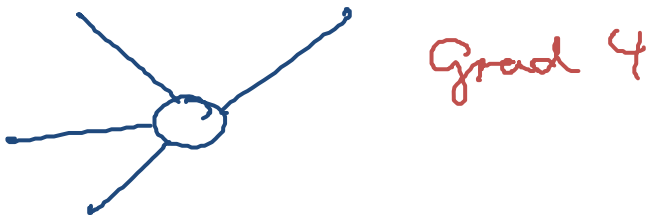
- Für einen Graphen $G = (V, E)$
  - falls gerichtet
    - Eingangsgrad von einem Knoten u

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

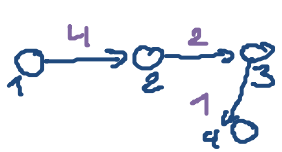
      = Anzahl ausgehender Kanten = $|(u,v) : (u,v) \in E|$
  - falls ungerichtet
    - Grad von einem Knoten u

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



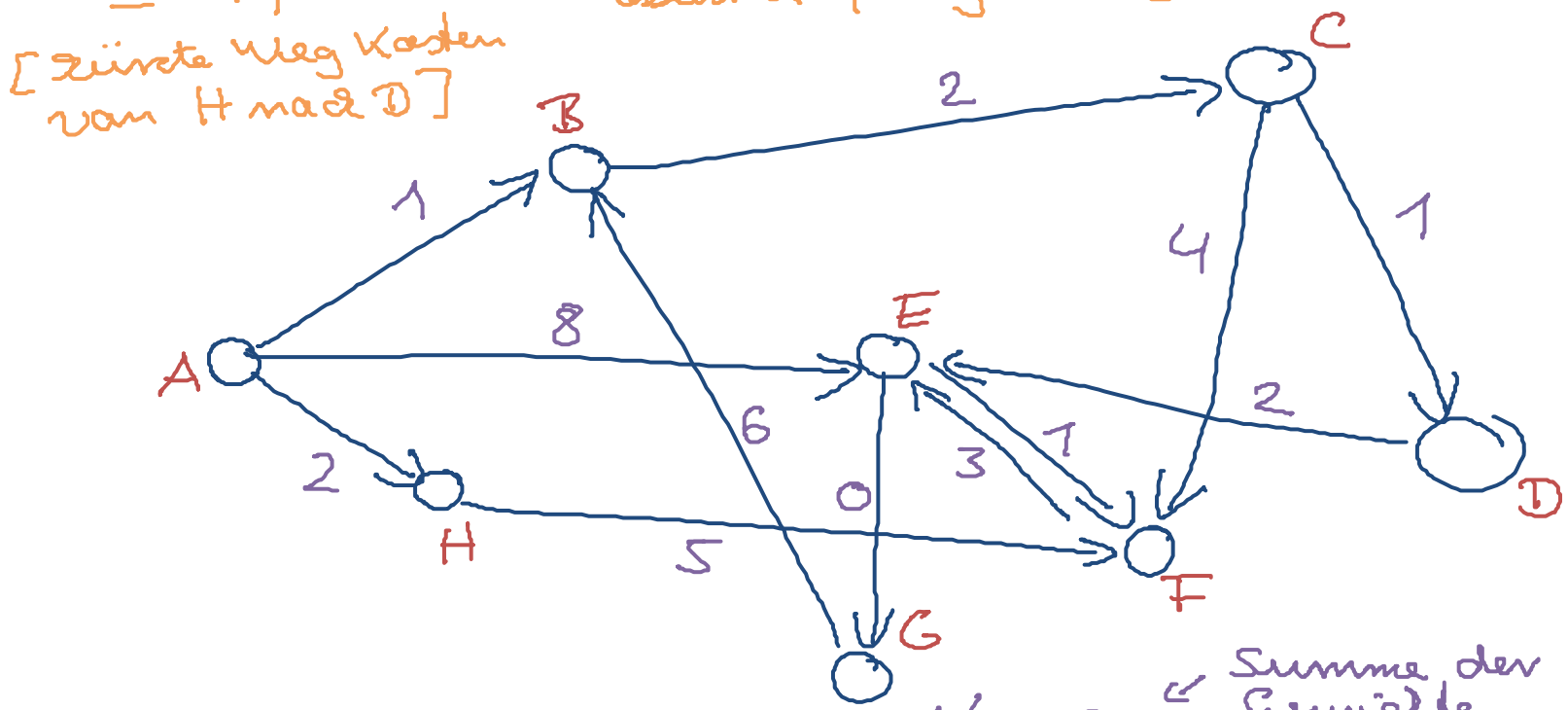


Graphen — Pfade



- Für einen 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 — Beispiel Pfade



Alle Pfade von A nach E:

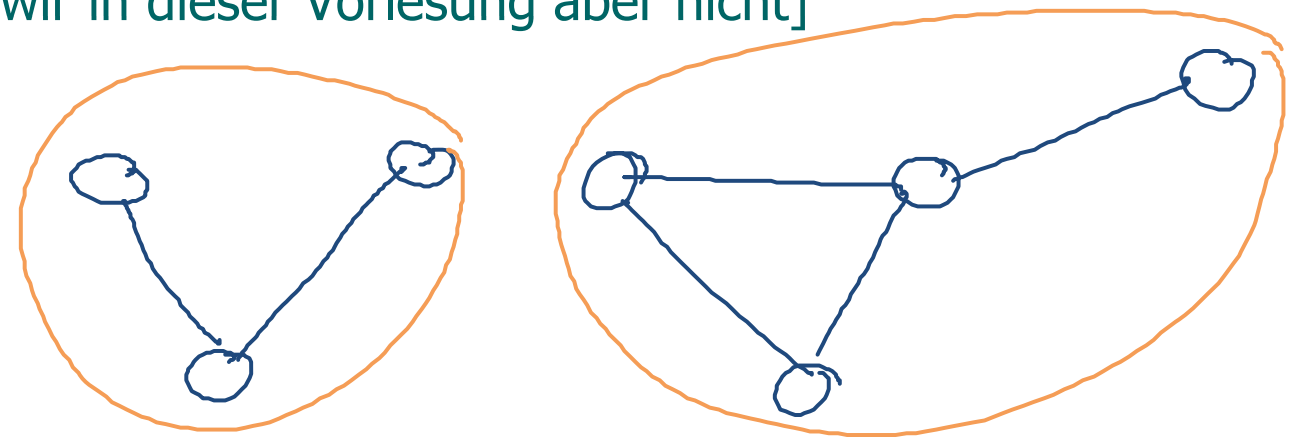
A → E
A → B → C → D → E
A → B → C → F → E
A → H → F → E

KOSTEN	← Summe der Gewichte
8	
6	← "kürzester" Pfad
10	
10	

Zusammenhangskomponenten

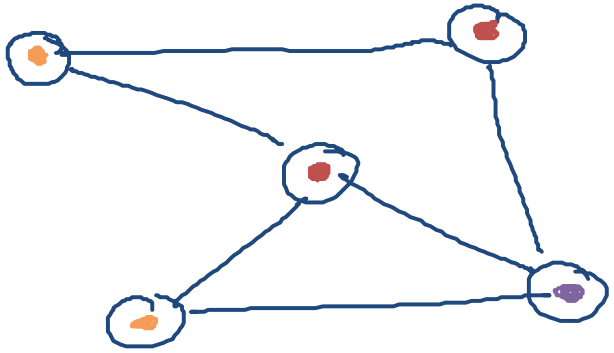
- Für einen **ungerichteten** Graphen $G = (V,E)$
  - Die Zusammenhangskomponenten bilden eine Partition von V, also $V = V_1 \cup \ldots \cup V_k$
  - Zwei Knoten u und v sind in derselben Zusammenhangskomponente, wenn es einen Pfad zwischen u und v gibt

[für **gerichtete** Graphen ist die Definition komplizierter, man spricht dann von **starken** Zusammenhangskomponenten, das machen wir in dieser Vorlesung aber nicht]



Graphenprobleme 1 / 5

- Graphfärbungsproblem (graph coloring)
  - Färbe die Knoten eines gegebenen Graphen so ein, dass keine zwei benachbarten Knoten (= durch eine Kante verbunden) dieselbe Farbe haben
  - Die minimale Anzahl benötigter Farben nennt man die chromatische Zahl des Graphen
  - Die Berechnung der chromatischen Zahl eines Graphen ist ein sogenanntes NP-vollständiges Problem



Chromatische Zahl = 3

Einschub: NP-Vollständigkeit

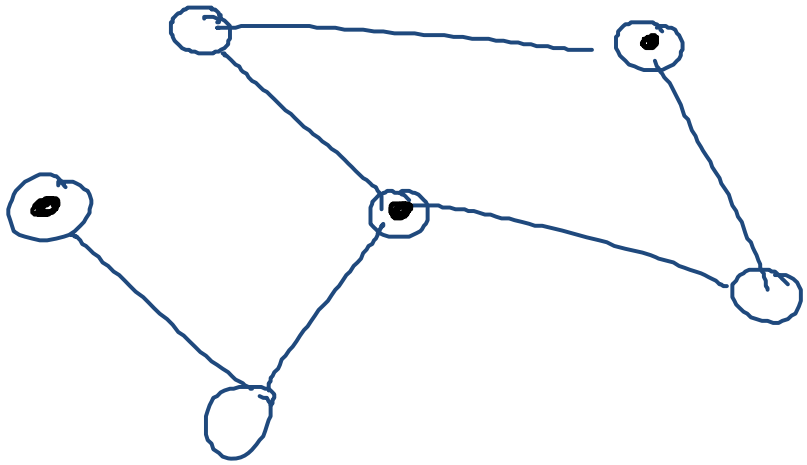
UNI FREIBURG

- Informaler 1-Folien-Crash-Kurs
  - Ein Problem ist in P, wenn es in Polynomialzeit lösbar ist
    - also in Zeit $O(n^k)$ für irgendeine Konstante k
  - Ein Problem ist in NP, wenn es für jede Lösung Zusatzinfo (sog. "Zeuge") gibt, mit deren Hilfe man sich in polynomieller Zeit von der Richtigkeit der Lösung überzeugen kann
  - Es ist eine offene Frage, ob $P = NP$ oder $P \neq NP$
  - Die meisten Leute vermuten aber $P \neq NP$
  - Ein Problem X ist NP-vollständig wenn
    - es in NP liegt, siehe oben
    - man jedes andere Problem in NP darauf reduzieren kann, das heißt wenn $X \in P$ dann ist schon $P = NP$

Graphenprobleme 2 / 5

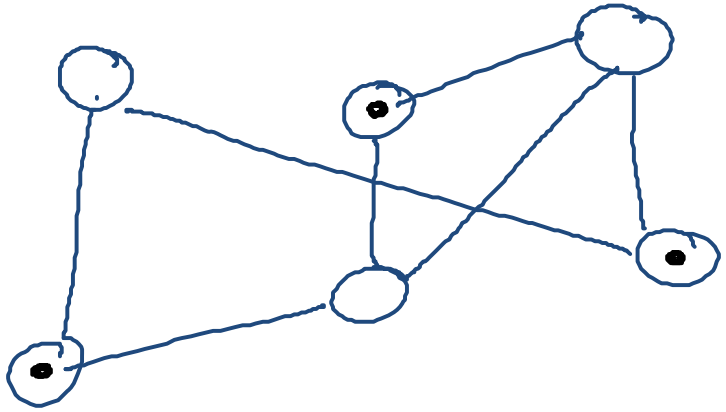
UNI FREIBURG

- Knotenüberdeckungen (vertex cover)
  - Für einen gegebenen Graphen G = (V, E) finde eine Teilmengen von Knoten V', so dass für jede Kante aus E mindestens einer ihrer beiden Endpunkte in V' liegt
  - Die Berechnung einer Knotenüberdeckung minimaler Größe ist ebenfalls ein NP-vollständiges Problem



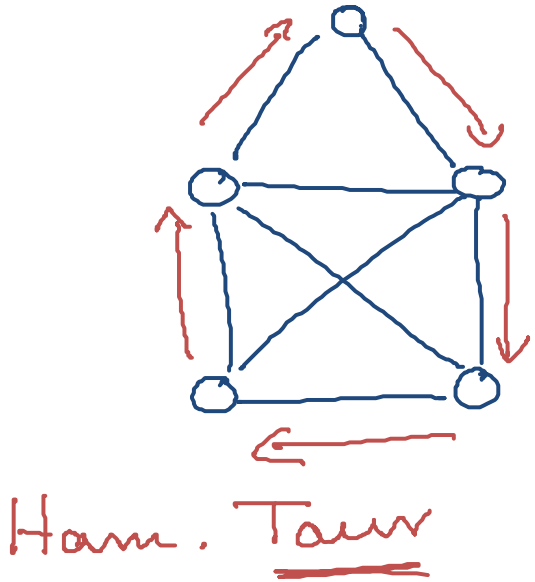
Graphenprobleme 3 / 5

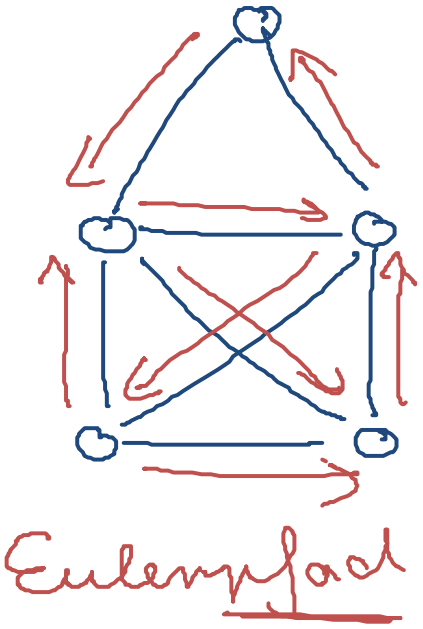
- Unabhängige Menge (independent set)
  - Für eine Graphen G = (V, E) finde eine Teilmenge von Knoten V', so dass zwischen keinem Paar u und v aus V' eine Kante aus E verläuft
  - Die Berechnung einer unabhängigen Menge maximaler Größe ist ebenfalls ein NP-vollständiges Problem



Graphenprobleme 4 / 5

- Hamiltonscher Pfad / Eulerpfad
  - Für einen Graph G = (V, E) ist ein Pfad auf dem jeder Knoten aus V genau einmal vorkommt ein **Hamiltonscher Pfad** und ein Pfad auf dem jede Kante aus E genau einmal vorkommt ein **Eulerpfad**
  - Wenn der Pfad wieder am Anfangsknoten ankommt, spricht man von einer **Tour**





Graphenprobleme 5 / 5

UNI FREIBURG

- Handlungsreisendenproblem (traveling salesman p.)
  - Für einen Graphen G = (V, E) mit Kantenlängen, finde die Hamiltonsche Tour kürzester Länge
  - Ebenfalls ein NP-vollständiges Problem

Graphexploration

UNI FREIBURG

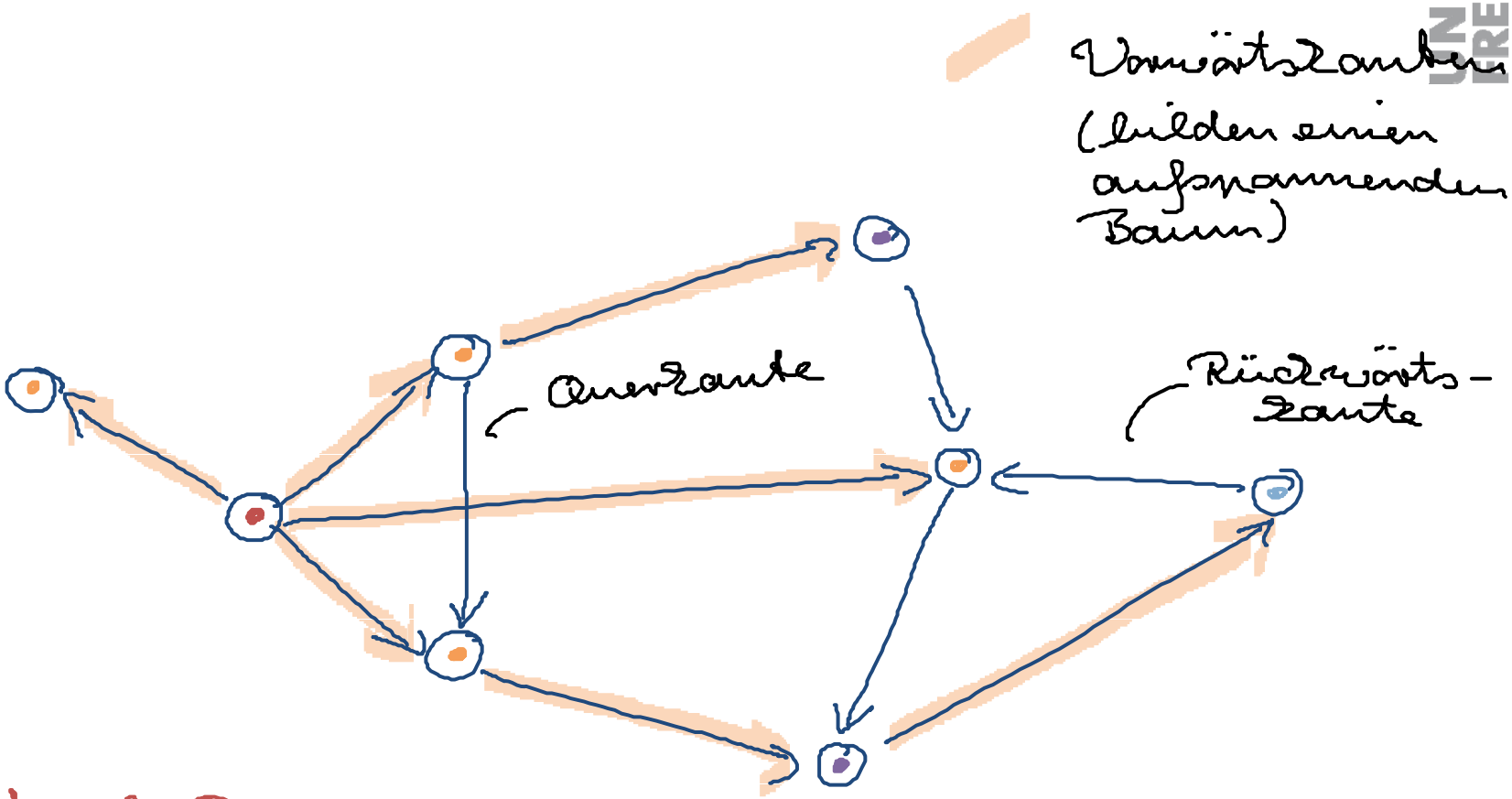
- Informale Definition
  - Gegeben ein Graph $G = (V, E)$ und ein Startknoten $s \in V$, besuche "systematisch" alle Knoten von V
  - Breitensuche = in der Reihenfolge der "Entfernung" von s
    - englisch breadth first search = BFS
  - Tiefensuche = erstmal "möglichst weit weg" von s
    - englisch depth first search = DFS
  - Das ist kein "Problem" an sich, taucht aber oft als Teil / Subroutine von anderen Algorithmen auf
    - zum Beispiel in der Übungsaufgabe, zur Berechnung des Durchmessers

Breitensuche / Breadth First Search / BFS

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 / Breadth First Search / BFS



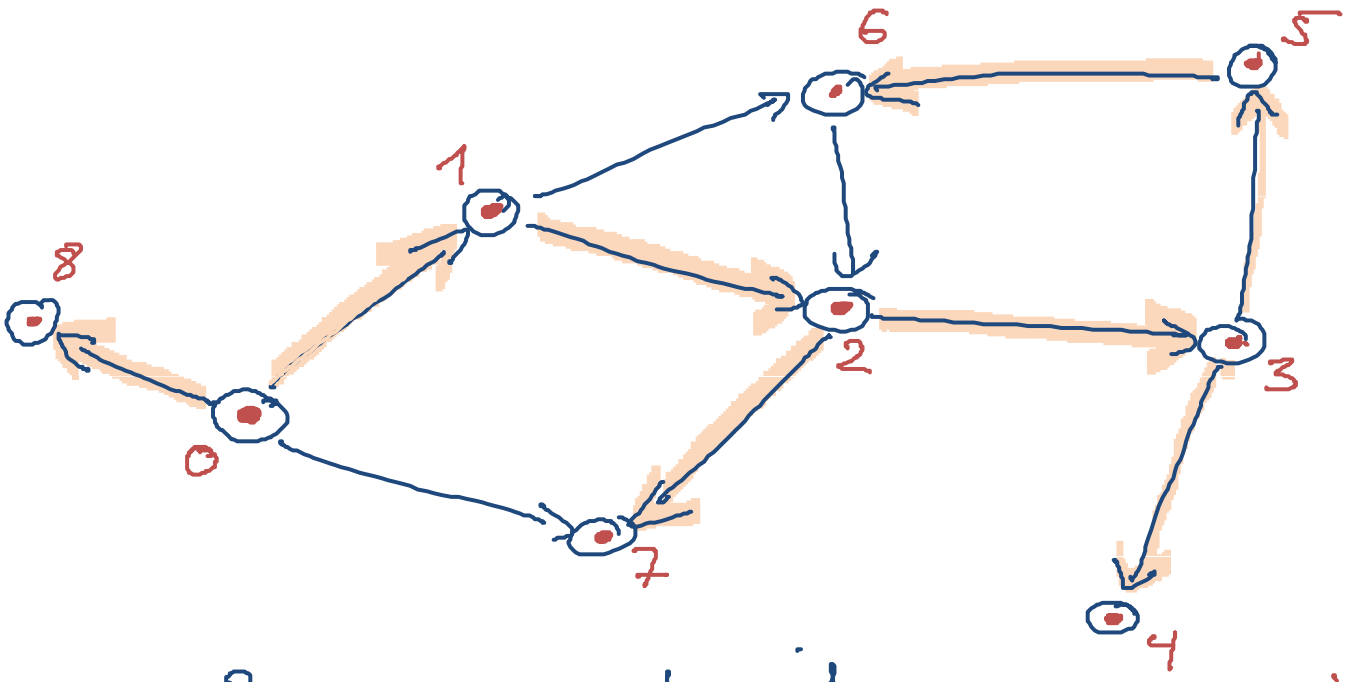
Tiefensuche / Depth First Search / DFS

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 Tiefensuche hat am Ende alle Knoten, die in derselben Zusammenhangskomponenten wie der Startknoten sind, markiert
  - Insbesondere kann man mit DFS **topologisch sortieren**

das ist eine Nummerierung der Knoten, so dass jede Kante von einer Nummer zu einer größeren Nummer geht.

Tiefensuche / Depth First Search / DFS



Kann man damit auch den Durchmesser berechnen?

Für keine topologische Sortierung wegen Kante (6,2) die DFS aber findet

wenn die Kante weg wäre, hätte man eine topologische Sortierung.

Komplexität von BFS und DFS

UNI FREIBURG

- Beide Verfahren schauen sich ...
  - ... jeden Knoten und jede Kante genau einmal an
  - Die Laufzeit ist also $O(|V| + |E|)$
  - Das kann man also (asymptotisch) nicht besser machen

Literatur / Links

- Graphen
  - In Mehlhorn/Sanders:

    8 Graph Representation
  - In Wikipedia

    http://en.wikipedia.org/wiki/Graph_(data_structure)

    http://pl.wikipedia.org/wiki/Reprezentacja_grafu
- Graphexploration
  - In Mehlhorn/Sanders:

    9 Graph Traversal
  - In Wikipedia

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

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