Algorithmen und Datenstrukturen (ESE)
Entwurf, Analyse und Umsetzung von Algorithmen (IEMS)
WS 2012 / 2013

Vorlesung 12, Dienstag, 22. Januar 2013
(Graphen, Breiten/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 Ü11 (a,b-Bäume)
- Graphen
  - Neben Feldern, Listen und Bäumen die häufigste Datenstruktur (Bäume sind eine spezielle Art von Graph)
  - Darstellung im Rechner
  - **Breitensuche** (Breadth First Search = BFS)
  - **Tiefensuche** (Depth First Search = DFS)
  - **Zusammenhangskomponenten** eines Graphen
  - **Übungsblatt 12:** Berechnung der größten Zusammenhangs-komponente in einem Straßengraphen mittels BFS oder DFS

Erfahrungen mit dem Ü11 (a,b-Bäume)

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 22. Januar 16:15
  - Vorlesung gut erklärt und Übungsblatt gut machbar
  - Man konnte sehr viel aus der Vorlesung übernehmen
  - Die meiste Arbeit war, es schön aufzuschreiben
  - Programmieraufgaben sind aber trotzdem interessanter
  - Manchmal etwas langatmig, wenn versucht wird, die anderthalb Stunden voll zu kriegen ... versuche ich nicht
  - Mehr Werbung wozu das alles gut ist, wie bei der MST

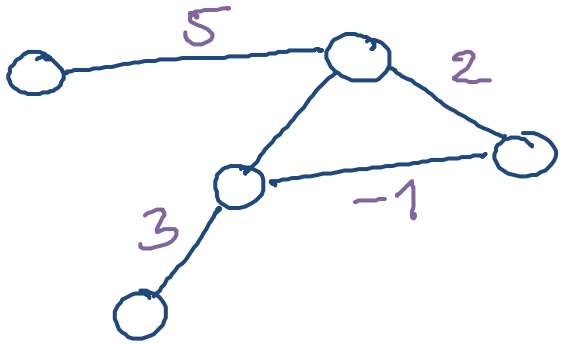
Graphen 1/6

UNI FREIBURG

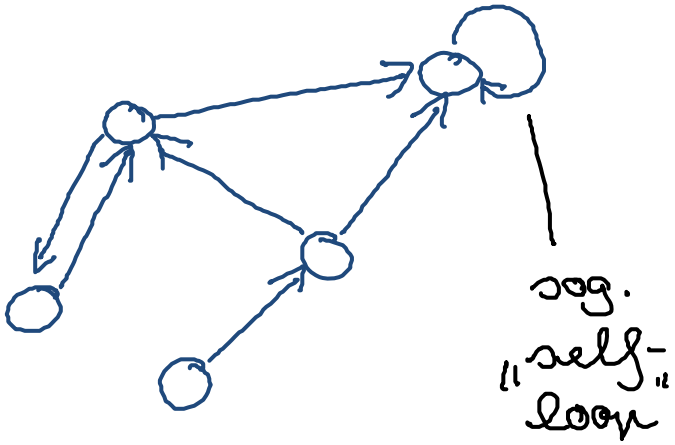
- Definition:
  - Ein Graph G besteht aus einer Menge V von Knoten ...
    - Englisch: vertices (daher V) oder nodes
  - ... und einer Menge E von Kanten
    - English: edges (daher E) oder arcs
  - Eine Kante e verbindet jeweils zwei Knoten u und v
    - ungerichtete Kante: $e = \{u, v\}$ (Menge)
    - 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

- Beispiele

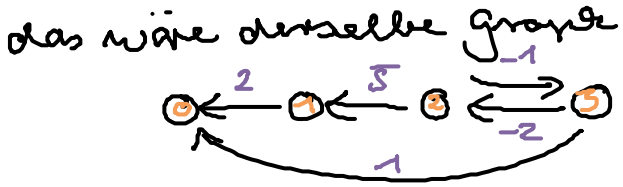


ungerichteter Graph mit Kanten-gewichten



gerichteter Graph (ohne Kanten-gewichte)

Graphen 3/6



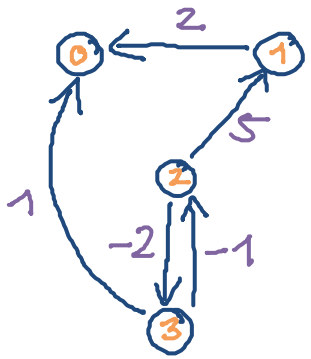
- Wie repräsentiert man Graphen im Rechner
  - Da gibt es zwei klassischen Arten

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

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

$|V| = 4, |E| = 5$

gerichteter Graph:



mit Kanten-gewichten

Adjazenz-matrix:

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

Bei ungerichteten Graphen: symm. Anzahl nicht-x Einträge = $|E|$

Adjazenz-listen:

0 : x

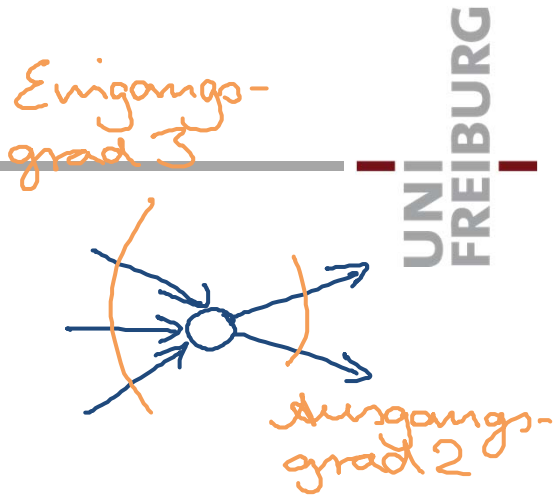
1 : 0, 2

2 : 1, 5 | 3, -2

3 : 0, 1 | 2, -1

Anzahl Elemente insgesamt = $|E|$

Graphen 4/6

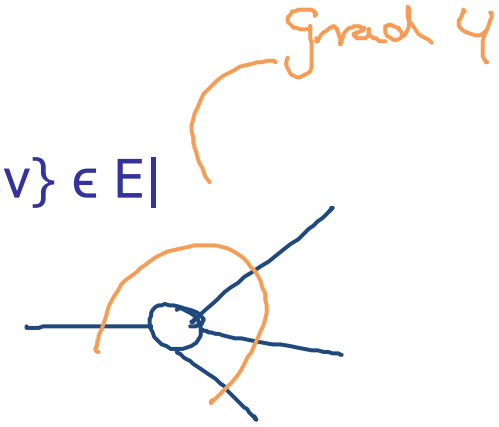


- Grade in einem Graphen G = (V, E)
  - Falls gerichtet
    - **Eingangsgrad** von einem Knoten u

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

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

      = Anzahl adjazenter Kanten = |{u,v} : {u,v} ϵ 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

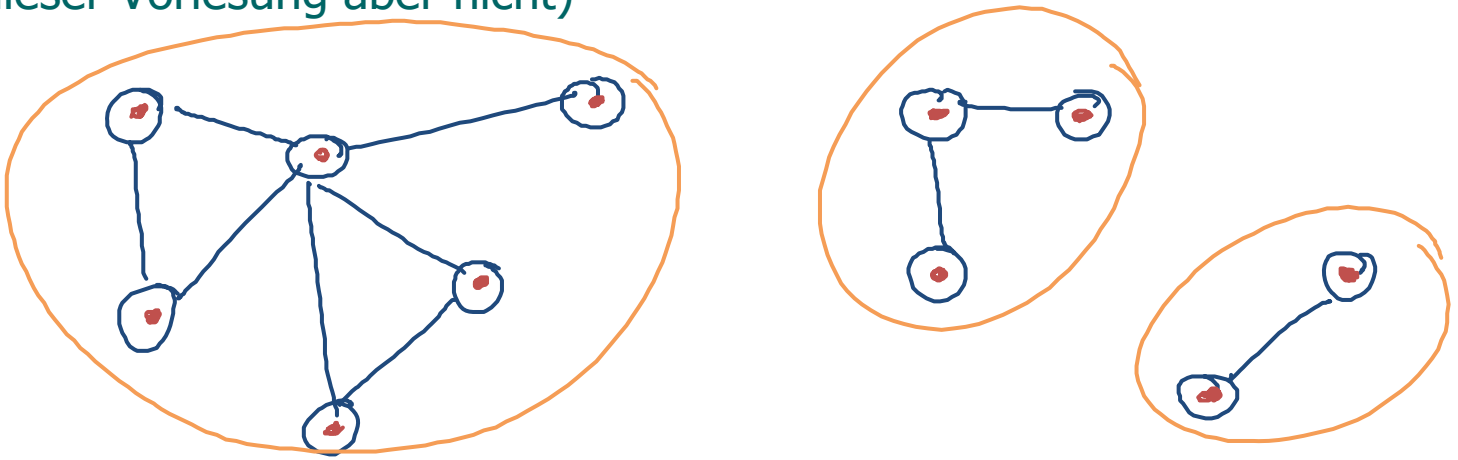
- Beispiel Pfade

Zusammenhangskomponenten

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)



Graphexploration

die erreichbar von s sind

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 der Zusammenhangskomponenten

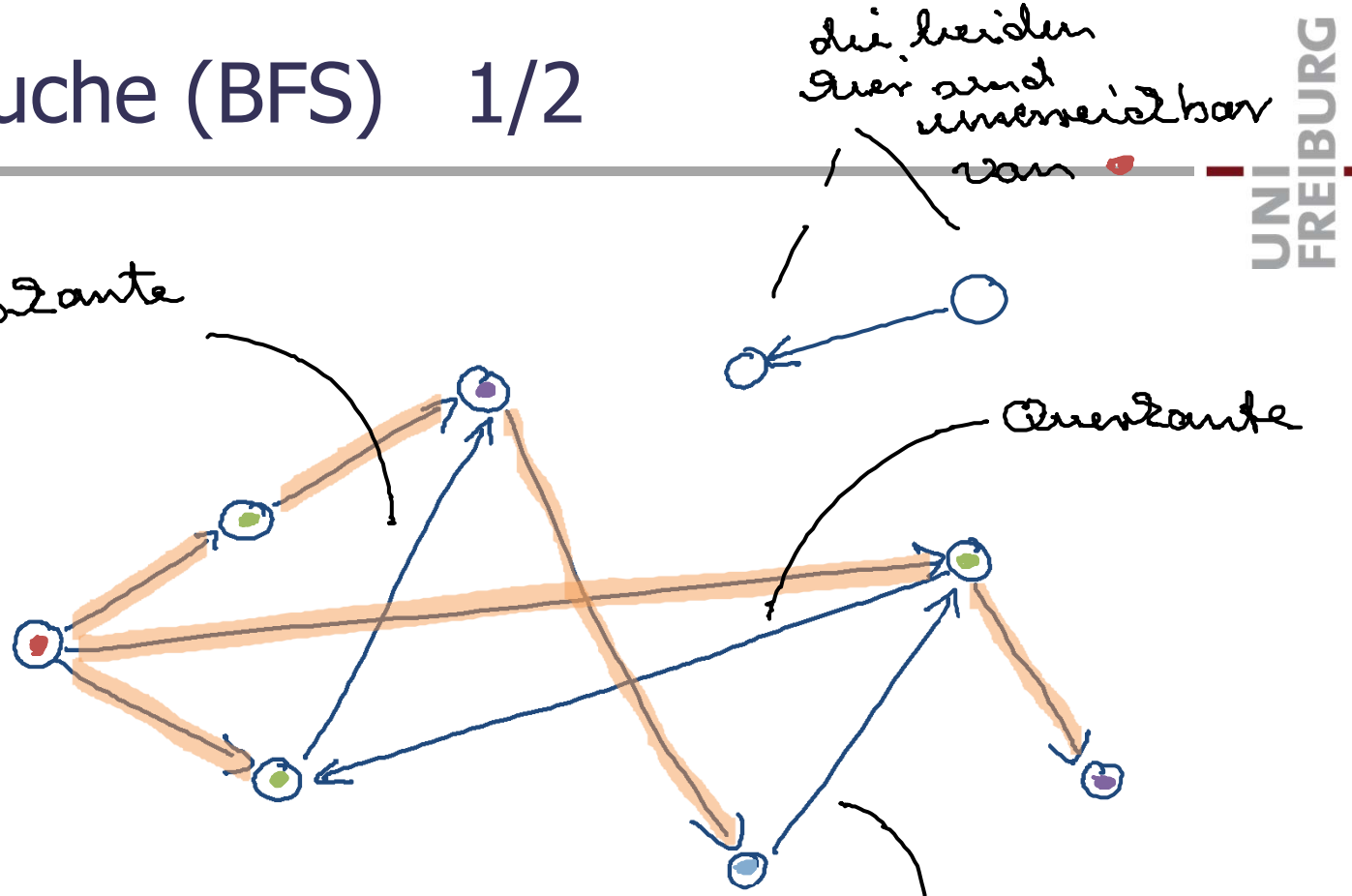
Breitensuche (BFS) 1/2

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



LEVEL 0 (START)
LEVEL 1
LEVEL 2
LEVEL 3

Diese Kanten bilden einen sogenannten aufspannenden Baum (spanning tree) = Baum der alle erreichbaren Knoten enthält

Tiefensuche (DFS) 1/2

UNI FREIBURG

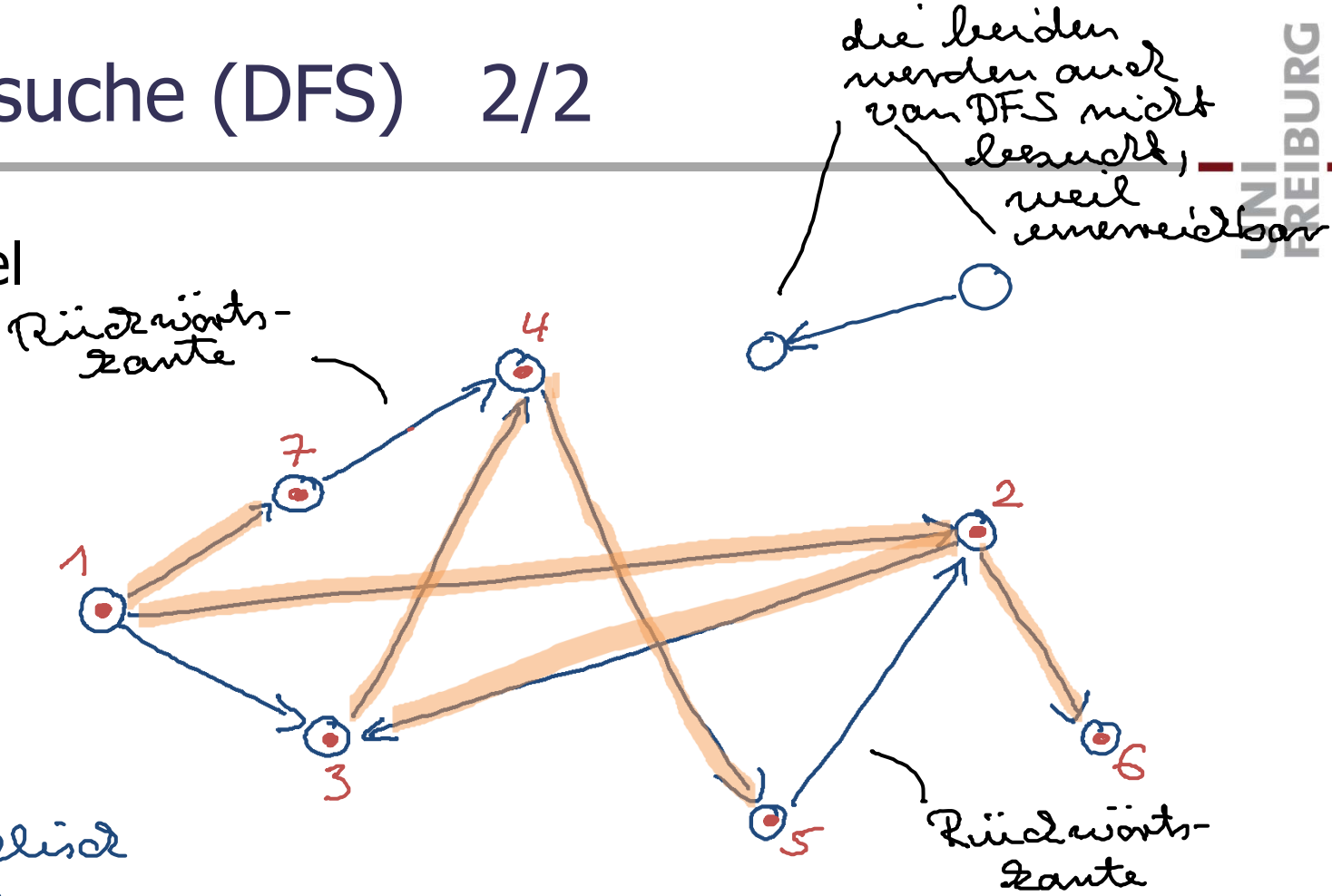
- Idee
  - Markierung für jeden Knoten, zu Beginn alle unmarkiert
  - Beginne mit einem Startknoten und markiere ihn
  - Gehe in irgendeiner Reihenfolge die zum Startknoten benachbarten Knoten durch und tue Folgendes:

    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

- Beispiel



Wenn der Graph azyklisch ist (z.B ohne die Kanten 7→4 und 5→2) dann sind die roten Zahlen eine topologische Sortierung

Auch wieder ein aufspannender Baum (aber ein anderer)

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 ZK sind, in der der Startknoten liegt
  - Das kann man also (bis auf einen konstanten Faktor) nicht besser machen

Literatur / Links

UNI FREIBURG

- Graphen
  - In Mehlhorn/Sanders:

    8 Graph Representation
  - In Wikipedia

    http://en.wikipedia.org/wiki/Graph_(mathematics)
- Graphexploration und Zusammenhangskomponenten
  - 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