Informatik II: Algorithmen und Datenstrukturen SS 2015

Vorlesung 10a, Dienstag, 30. Juni 2015
(Graphen, Exploration, Zusammenhang)

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)
  - Welche Fragen kommen in der Klausur
- Inhalt
  - Graphen … Terminologie, Adjazenzmatrix und -listen
  - Graphexploration … Breiten- und Tiefensuche + Code dazu
  - Zusammenhangskomponenten … Algorithmus + Code dazu
  - Ü10, Aufgabe 1: Bestimmen Sie die Laufzeit von Breitensuche und der Berechnung der Zusammenhangskomponenten

Erfahrungen mit dem Ü9 1/2

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 30. Juni 12:00
  - Aufgaben haben vielen gefallen, vor allem Aufgabe 1
  - changeKey vom Minimum aufwärts geht nicht in O(1)

    Gut beobachtet, aber braucht man nicht: kann man durch deleteMin und insert ersetzen … siehe Forum
  - Unklar ob man annehmen kann, ob M vorher gegeben ist

    Beides war ok, aber nicht gegeben war auch nicht schwerer
  - Aufgabe 2 ging auch mit einem Feld von Feldern
  - Ungläubig dass Aufgabe 1 in O(n log k) gehen soll
  - Jemand meinte (über sich): "Vielleicht färbt dieses Jahr meine unglaubliche Intelligenz auf die anderen Studenten ab"

Erfahrungen mit dem Ü9 2/2

UNI FREIBURG

- Lösungsskizze + Programm Aufgabe 1
  - Immer höchstens k Elemente in der PW
  - Dadurch Laufzeit pro Operation O(log k)
  - Falls schon k in der PW sind und ein neues Element dazukommt, das größer ist als das kleinste, was drin ist:

    Kleinstes Element rausschmeißen und neues dazu nehmen

    Der Trick ist immer mit dem **kleinsten** der aktuell k größten Elemente zu vergleichen
  - Am Ende die k Elemente nochmal richtig sortieren

    Sie stehen ja gerade "falschrum" in der PW (kleinstes zuerst) und außerdem nicht ganz sortiert, nur Heap-Eigenschaft

Graphen 1/4

UNI FREIBURG

- Definition:
  - Ein Graph G besteht aus zwei Mengen V und E

    V = Menge der **Knoten** … engl. "nodes" oder "**v**ertices"

    E = Menge der **Kanten** … engl. "**e**dges" oder "arcs"
  - Eine Kante e verbindet jeweils zwei Knoten u und v

    ungerichtete Kante: $e = \{u, v\}$ Menge

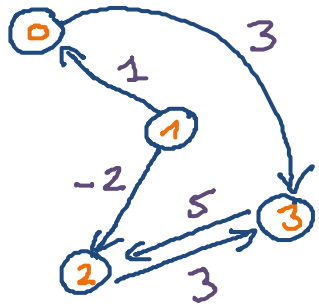
    gerichtete Kante: $e = (u, v)$ Tupel
  - Bei einem **gewichteten** Graph hat man für jede Kante ein Gewicht, auch Länge oder Kosten der Kante genannt

    Für Ü10, Aufgabe 2 ist das zum Beispiel die Reisezeit zwischen zwei Punkten

Graphen 2/4

- Beispiel + Repräsentation im Rechner
  - **Adjazenzmatrix** … Platzverbrauch $\Theta(|V|^2)$
  - **Adjazenzlisten** … Platzverbrauch $\Theta(|V| + |E|)$

4 Knoten
5 Kanten
mit Knotennummern
und Kantengewichten



ADJ. MATRIX

	0	1	2	3
0	×	×	×	3
1	1	×	-2	×
2	×	×	×	3
3	×	×	5	×

× = keine Kante
Anzahl nicht-× Einträge
= Anzahl Kanten

Für jeden Knoten die Liste aller von ihm ausgehenden Kanten
ADJ. LISTEN

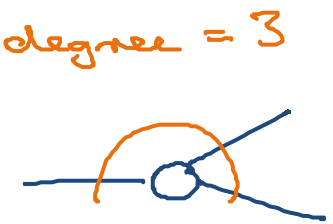
0 : [3|3]
1 : [0|1|2|-2]
2 : [3|3]
3 : [2|5]

soviele Einträge wie es Kanten gibt

Graphen 3/4

UNI FREIBURG

- Grad, Eingangsgrad, Ausgangsgrad



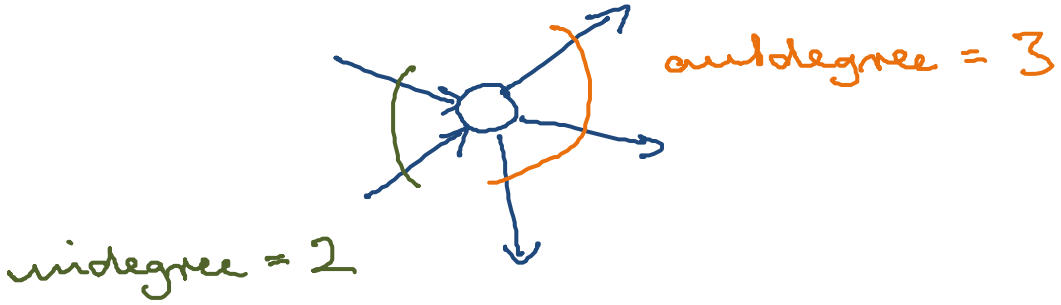
  - Bei einem ungerichteten Graph

    degree(u) = |{u, v} : {u, v} ∈ E| ... Grad

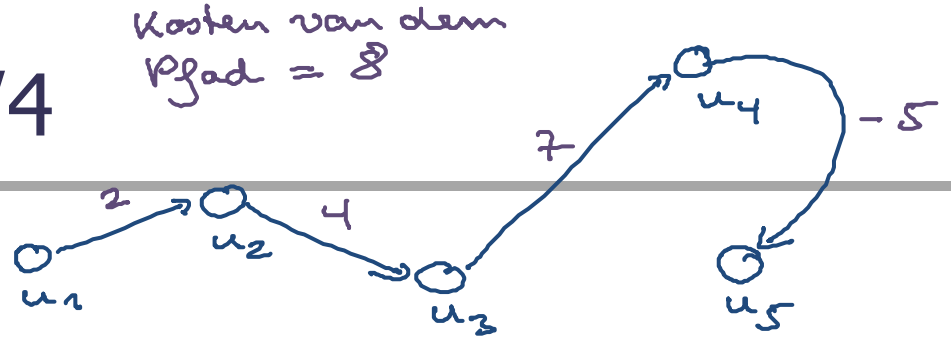
  - Bei einem gerichteten Graph

    outdegree(u) = |(u, v) : (u, v) ∈ E| ... Ausgangsgrad

    indegree(u) = |(v, u) : (v, u) ∈ E| ... Eingangsgrad



Graphen 4/4



- Pfade
  - 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** bzw. **Kosten** eines Pfades

    ohne Kantengewichte: Anzahl der Kanten

    mit Kantengewichten: Summe der Gewichte auf dem Pfad
  - Der **kürzeste Pfad** (engl. shortest path) zwischen zwei Knoten u und v ist der Pfad u, ..., v mit der kürzesten Länge

Graphexploration 1/7

UNI FREIBURG

- Informale Definition
  - Gegeben ein Startknoten s , 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 relevantes "Problem" an sich, taucht aber oft als Teil / Subroutine von anderen Algorithmen auf

    Zum Beispiel zur Berechnung der Zusammenhangs-komponenten eines Graphen, siehe Folien 16 – 18

Graphexploration 2/7

UNI FREIBURG

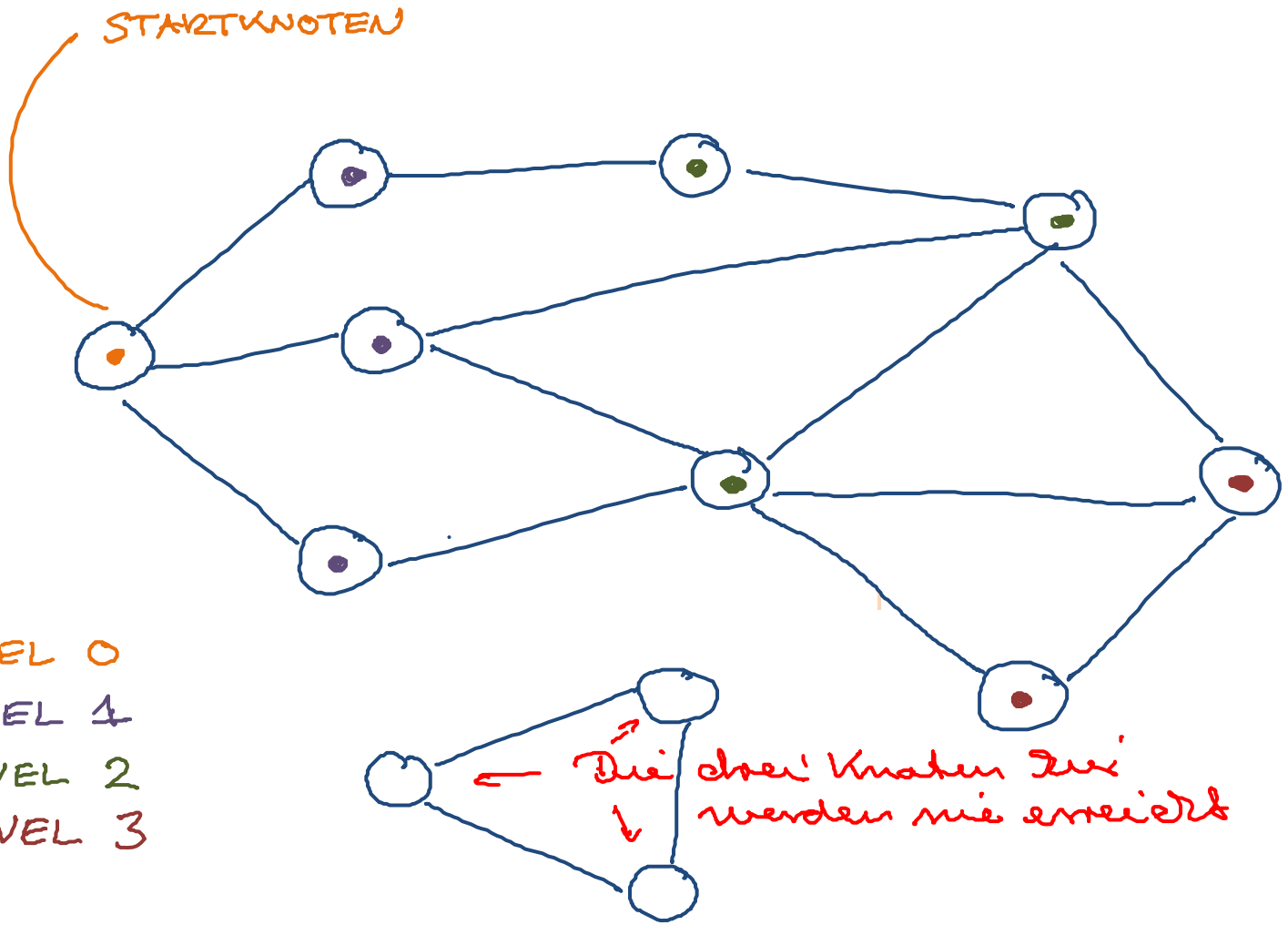
- Breitensuche, Idee
  - Markierung für jeden Knoten, zu Beginn alle unmarkiert
  - Beginne mit einem Startknoten und markiere ihn (Level 0)
  - Finde alle Knoten die zum Startknoten benachbart und noch nicht markiert sind und markiere sie (Level 1)
  - Finde alle Knoten, die zu einem Level 1 Knoten 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

Graphexploration 3/7

UNI FREIBURG

- Breitensuche, Beispiel ungerichteter Graph



Graphexploration 4/7

UNI FREIBURG

- Tiefensuche, 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

Graphexploration 5/7

- Tiefensuche, Beispiel

Graphexploration 6/7

UNI FREIBURG

- Tiefensuche, weitere Eigenschaft
  - Auf azyklischen Graphen liefert Tiefensuche eine sogenannte **topologische Sortierung**

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

    Man verstehe: mit einem Zyklus kann das nicht gehen

Graphexploration 7/7

UNI FREIBURG

- Komplexität von Breitensuche und Tiefensuche
  - Das ist gerade Aufgabe 1 vom Ü10, Teil 1
  - Es ist insbesondere Teil der Aufgabe herauszufinden, von welchen "Maßzahlen" die Laufzeit abhängt

    Das hatten wir ja jetzt schon öfter, z.B. Laufzeit von insert beim Cuckoo Hashing O(s), mit s = #Rauswürfe

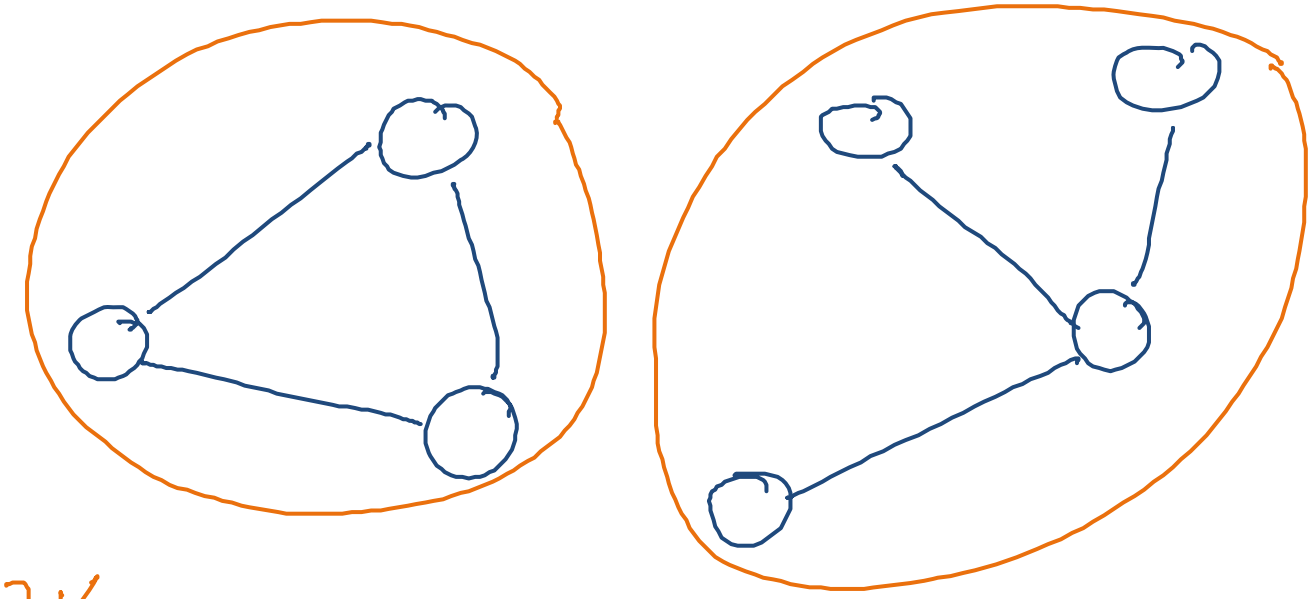
    Sie dürfen aber nicht schreiben: Laufzeit ist O(x), mit x = Anzahl der benötigten Operationen

Zusammenhangskomponenten 1/3

UNI FREIBURG

= gerichtet mit allen Kanten in beide Richtungen

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



2 ZK

Zusammenhangskomponenten 2/3

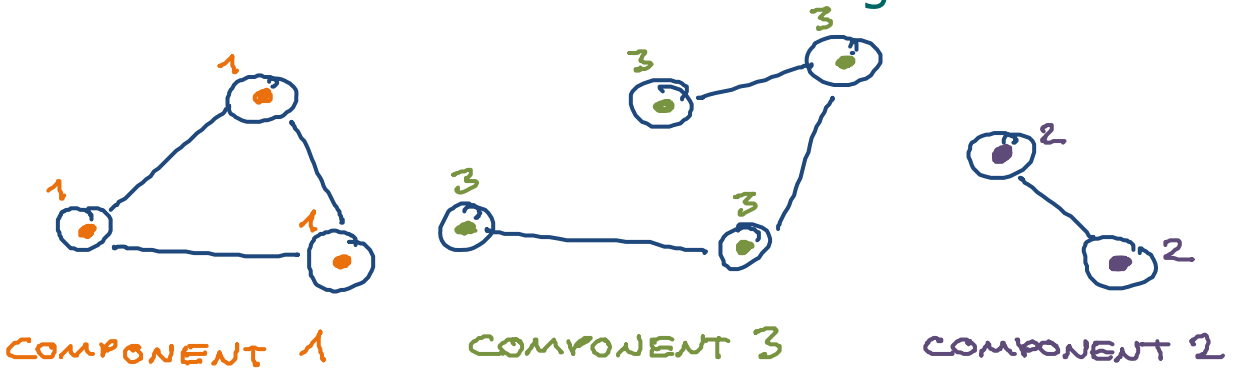
UNI FREIBURG

- Berechnung durch DFS oder BFS
  - Markierung für jeden Knoten, zu Beginn alle unmarkiert
  - Solange es noch einen unmarkierten Knoten x gibt:

    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 ob DFS oder BFS

    Die so von x erreichten Knoten bilden genau eine ZK



Zusammenhangskomponenten 3/3

UNI FREIBURG

- Definition für einen gerichteten Graphen
  - Man spricht dann von **starken** Zus.komponenten (ZK)
  - Eine starke ZK ist eine maximale Teilmenge von Knoten V', so dass es für alle $u, v \in V'$ eine Pfad von u nach v gibt

    Nicht mehr so intuitiv, wie bei ungerichteten Graphen

    Der Algorithmus ist auch komplizierter und machen wir hier nicht … geht aber sehr elegant mit Tiefensuche

Literatur / Links

- Graphen, Breitensuche, Tiefensuche, ZK
  - In Mehlhorn/Sanders:

    8 Graph Representation

    9 Graph Traversal
  - In Wikipedia

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

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

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

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