Informatik II: Algorithmen und Datenstrukturen SS 2017

Vorlesung 10a, Dienstag, 4. Juli 2017
(Graphen, Exploration, Zusammenhang)

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

Blick über die Vorlesung heute

- Organisatorisches
  - Offizielle Evaluation — kurze Erklärung dazu
  - Erfahrungen ÜB9 — Prioritätswürgeschlangen
- Inhalt
  - Graphen — Terminologie
  - Breitensuche — Algorithmus + Beispiel + Code
  - Tiefensuche — Algorithmus + Beispiel
  - Zusammenhangskomponenten — Algorithmus + Beispiel + Code
  - ÜB10: Routenplanung in Baden-Württemberg

Offizielle Evaluation der Veranstaltung

- Läuft über das zentrale **EvaSys** der Uni
  - Sie sollten gestern (Montag, 3. Juli) eine Mail vom System bekommen haben

    Falls nicht, bitte umgehend bei Axel Lehmann melden !
  - Nehmen Sie sich bitte Zeit und füllen Sie den Bogen **sorgfältig und gewissenhaft** aus

    Sie haben soviel Zeit in die Vorlesung investiert, dann können Sie auch 30 Minuten für die Evaluation aufwenden

    Sie bekommen außerdem 20 Punkte dafür, die die Punkte vom schlechtesten ÜB ersetzen … siehe ÜB10, Aufgabe 1
  - Uns interessieren besonders die **Freitextkommentare**

Erfahrungen ÜB9   1/4

- Zusammenfassung / Auszüge
  - Aufgabe 1 hat vielen gefallen (Nachdenken + wenig Code)
  - Ungläubig, dass Aufgabe 1 in $O(n \cdot \log k)$ gehen soll

    Siehe Lösungsskizze nächste Folie
  - Aufgabe 2 war auch gut machbar
  - Coden in C++ relativ aufwändig
  - Es sollte "im Wiki/Forum" heißen, nicht "auf dem Wiki/Forum"

Erfahrungen ÜB9 2/4

oder einfach sortieren

- Lösungsskizze + Programm Aufgabe 1
  - Idee: zu jeden Zeitpunkt höchstens k Elemente in der PW
  - Dadurch Laufzeit O(log k) / Operation, insgesamt O(n log k)
  - Falls schon k Elemente in der PW sind, das neue Element x mit dem aktuellen Min in der PW vergleichen

    Falls x > Min: deleteMin und insert(x); sonst: nix tun
  - Am Ende die k Elemente eins nach dem anderen mit deleteMin aus der PW holen, in Zeit O(k log k)

    Damit bekommt man sie in absteigender Reihenfolge, aber die kann man ja leicht umdrehen

Erfahrungen ÜB9 3/4

- Live-Vorlesung vs. Online-Vorlesung
  - Wer eh nur Aufzeichnungen schaut: **kein Unterschied**

    Das war mit Abstand der häufigste Kommentar
  - Einige hören die Vorlesung trotzdem lieber live, allerdings mit relativ "weichen" Argumenten

    Termin zu dem man hin muss, weniger Ablenkung
  - Aufzeichnung hat objektive Vorteile: man kann anhalten, zurückspulen, vorspulen, Geschwindigkeit erhöhen, etc.

    Ideales Tempo variiert stark zwischen Teilnehmern
  - Wichtig: Inhalt aktuell + die VL lebendig + das **Forum**
  - Gleiches T-Shirt wie SS 2015, aber schlechtere Beleuchtung :-)

Erfahrungen ÜB9 4/4

- Kaum Fragen in den letzten Wochen?
  - Die meisten Fragen kommen bei der Bearbeitung des ÜB
  - Bei Verständnisschwierigkeiten oft nicht leicht, sich auf die schnelle eine gute Frage zu überlegen + sich nicht trauen
  - Außerdem häufiges Feedback: die Erklärungen seien so gut
- Fazit (aus Sicht der Dozentin)
  - Qualität von Aufzeichnungen / VL / Forum stark gelobt
  - Vorteile der Live-Vorlesung eher "nice to have"
  - Vorteile der Video-Aufzeichnungen sind objektiver
  - Live-Vorlesung würde mehr Sinn machen, **nachdem** man sich mit dem Stoff / einer Aufgabe auseinandergesetzt hat

Graphen 1/5

- 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: $e = \{u, v\}$ $= \{v, u\}$ **Menge**

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

    Für ÜB10: die Reisezeit zwischen zwei Punkten

Graphen 2/5

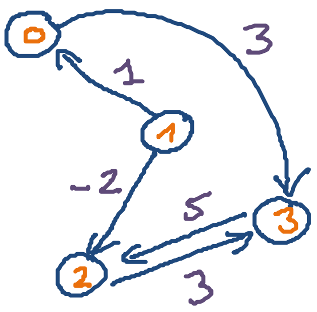
$n=6,\ m=0$

zusammenhängender Graph: $m \geq n-1$

generell $m \leq n^2$, falls keine „Multi"-Kanten

- Repräsentation gerichteter Graph
  - **Adjazenzmatrix** ... Platzverbrauch $\Theta(|V|^2)$
  - **Adjazenzlisten** ... Platzverbrauch $\Theta(\overset{n}{|V|} + \overset{m}{|E|})$

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 : $\boxed{3 | 3}$

1 : $\boxed{0 | 1}\ \boxed{2 | -2}$

2 : $\boxed{3 | 3}$

3 : $\boxed{2 | 5}$

soviele Einträge wie es Kanten gibt

Graphen 3/5

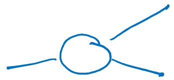
- Repräsentation ungerichteter Graph
  - Einen ungerichteten Graphen kann man einfach als gerichteten Graphen darstellen, bei dem es jede Kante in beide Richtungen gibt
  - Falls es Kantenkosten gibt, sind die Kosten dann in beiden Richtungen gleich

    So machen wir es auch für unseren Code nachher

Graphen 4/5

- Grad, Eingangsgrad, Ausgangsgrad
  - Grad einen Knotens u in einem ungerichteten Graph

    degree(u) = |{{u, v} : {u, v} ϵ E}|

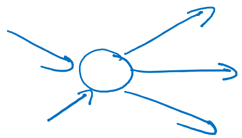
    

    Grad 3

  - Eingangs- und Ausgangsgrad eines Knotens u in einem gerichteten Graph

    in-degree(u) = |{(v, u) : (v, u) ϵ E}|

    out-degree(u) = |{(u, v) : (u, v) ϵ E}|

    E-Grad 2

    A-Grad 3

Graphen 5/5

- Pfade
  - Ein Pfad in G ist eine Folge $u_1, u_2, u_3, ..., u_l \in V$ mit

    $(u_1,u_2), (u_2,u_3), ..., (u_{l-1},u_l) \in E$ gerichteter Graph

    $\{u_1,u_2\}, \{u_2,u_3\}, ..., \{u_{l-1},u_l\} \in E$ 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 minimalen Kosten

    Dazu Beispiele und mehr in der VL10b (Dijkstra Algorithmus)

Graphexploration 1/7

- 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

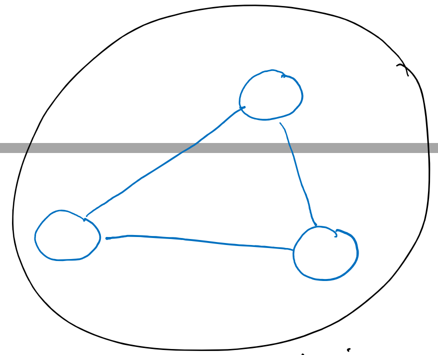
    Zum Beispiel zur Berechnung der Zusammenhangs-komponenten eines Graphen, siehe Folien 20 – 22

Graphexploration 2/7

- 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

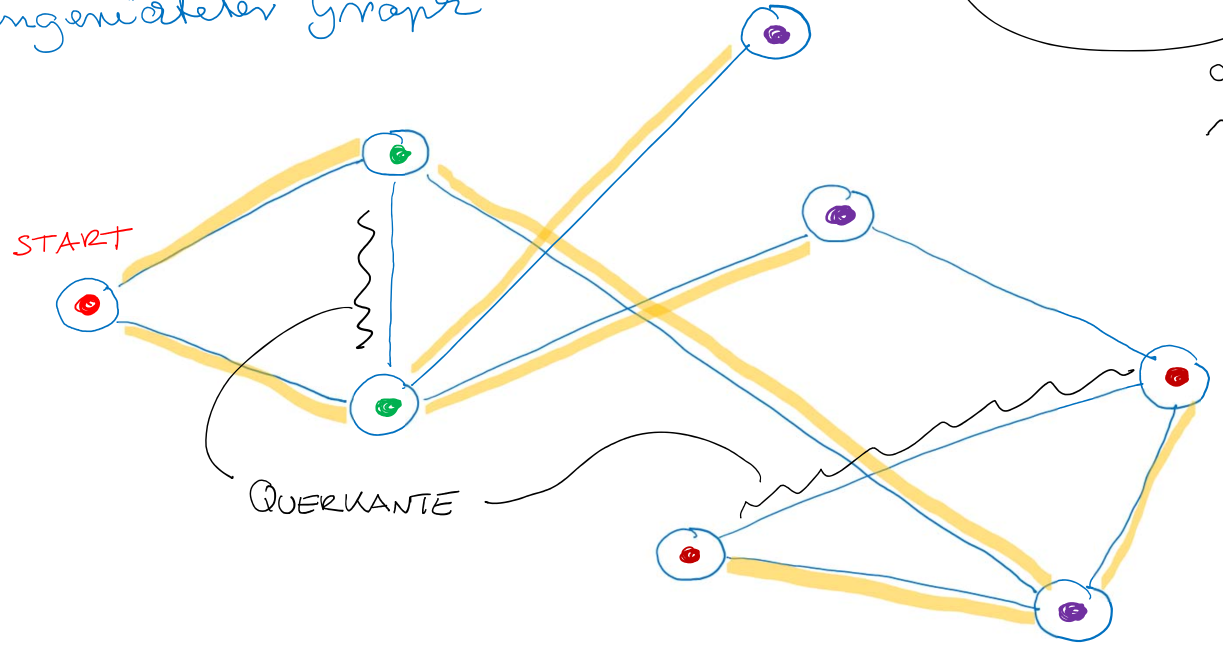
Graphexploration 3/7



- Breitensuche, Beispiel

ungerichteter Graph

die werden gar nicht erreicht



LEVEL 0
LEVEL 1
LEVEL 2
LEVEL 3

Die Kanten (von einem Level zum nächsten, nur eine Kante für jeden Knoten vom nächsten Level) bilden einen BAUM (einen sog. Spannbaum der ZK des Graphen)

Graphexploration 4/7

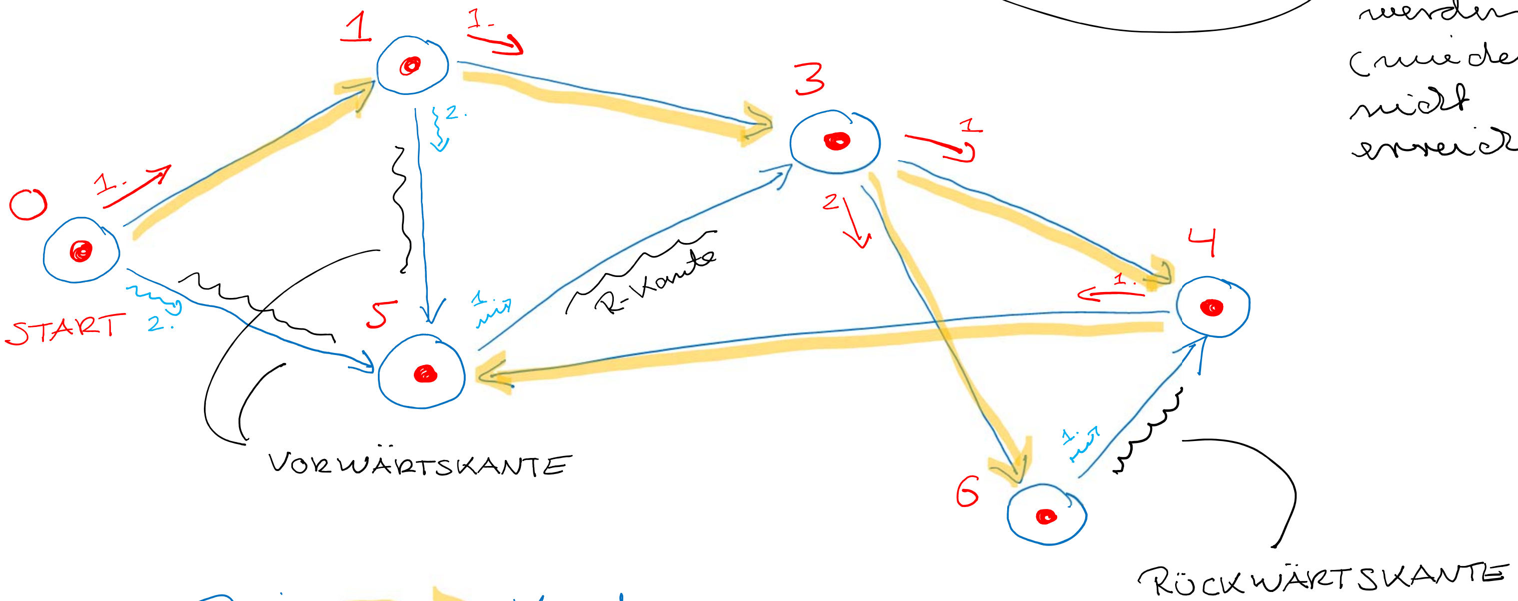
- Tiefensuche, Idee
  - Markierung für jeden Knoten, zu Beginn alle unmarkiert
  - Beginne mit einem Startknoten und markiere ihn
  - Gehe in irgendeiner Reihenfolge die zum Startknoten benachbarten Knoten durch und tue Folgendes:

    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

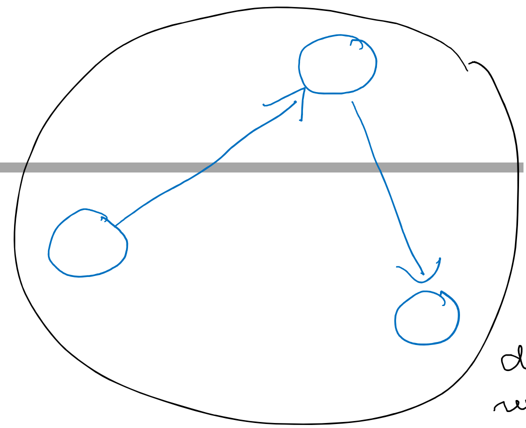
Graphexploration 5/7

- Tiefensuche, Beispiel

gerichteter Graph



Die → Kanten bilden wieder einen Baum (den Spannbaum)



die werden (wieder) nicht erreicht

Graphexploration 6/7

⟺ es gibt keine R-Kanten

- 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

    Wohlgemerkt: mit einem **Zyklus** kann das nicht gehen

Graphexploration 7/7

- Komplexität von BFS und DFS
  - Für beide Verfahren gilt:

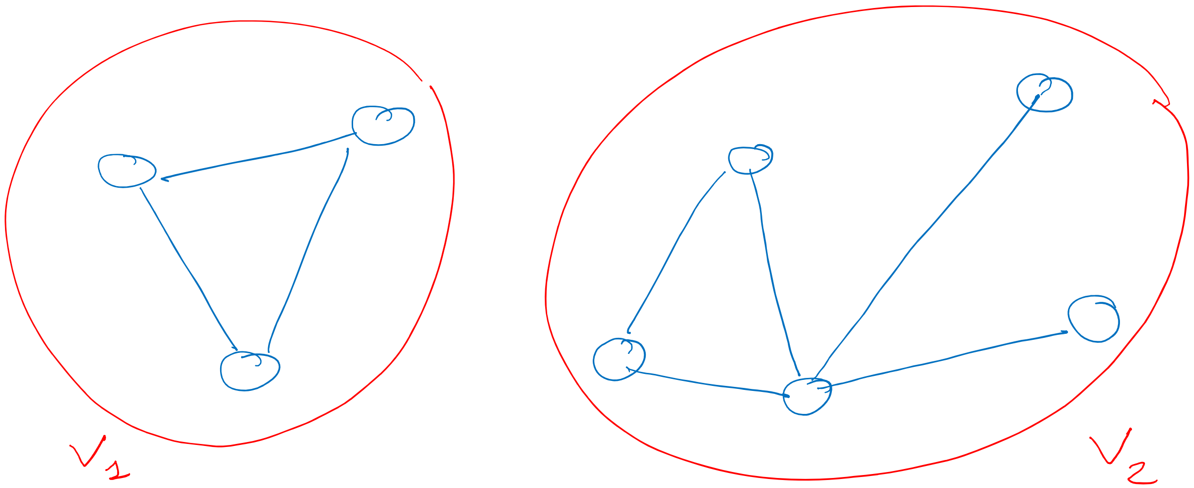
    1. Man folgt jeder Kante genau einmal

    2. Man "verarbeitet" jeden Knoten genau einmal (in dem Sinne, das man seinen adjazenten Kanten folgt)
  - Die Laufzeit ist also **O(|V'| + |E'|)**, wobei V' und E' die Anzahl Knoten und Kanten in der Zusammenhangs-komponente sind, in der der Startknoten liegt
  - Besser geht es offenbar nicht

    Weil man jeden Knoten und jede Kanten mindestens einmal "anschauen" muss

Zusammenhangskomponenten 1/3

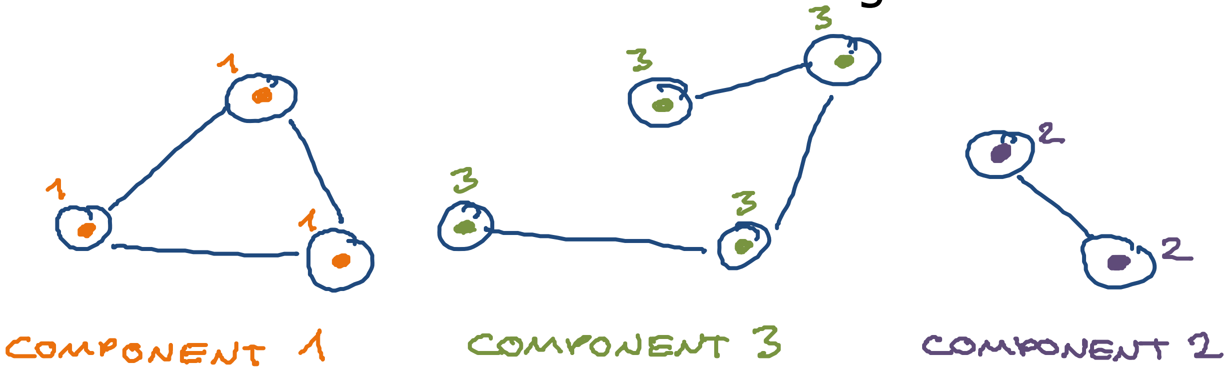
- Für einen **ungerichteten** Graphen
  - Die Zusammenhangskomponenten (ZK) bilden eine Partition von V, also $V = V_1 \uplus \ldots \uplus V_k$
  - Zwei Knoten u und v sind in derselben ZK genau dann, wenn es einen Pfad zwischen u und v gibt



Zusammenhangskomponenten 2/3

- 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

- Definition für einen **gerichteten** Graphen

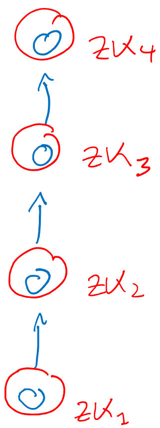
trotz dieser beiden Kanten, zwei starke ZK

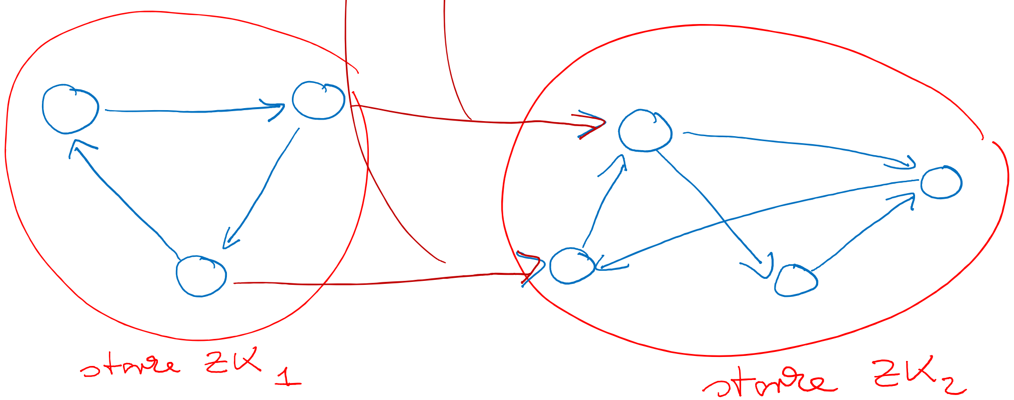
  – Man spricht dann von **starken** Zus.komponenten (ZK)

  – Eine starke ZK ist eine maximale Teilmenge von Knoten V', so dass es für alle u, v ∈ 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