Vorlesung Datenstrukturen

Graphen

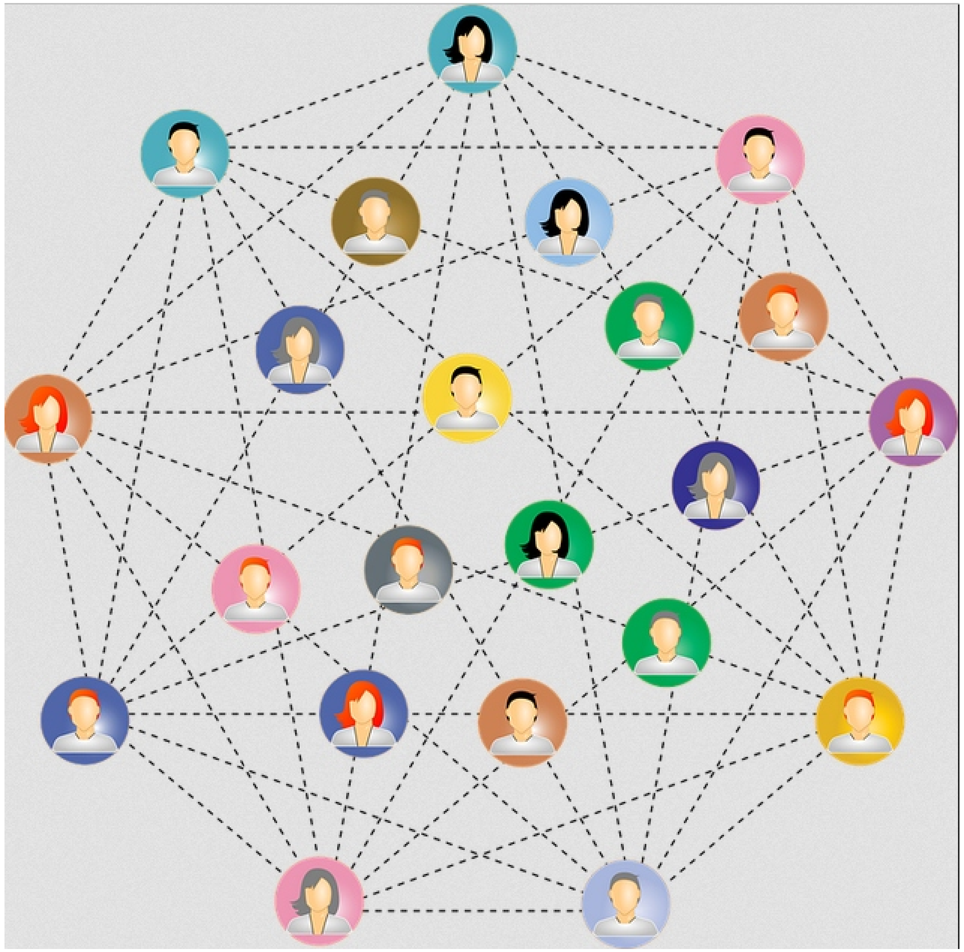
Maike Buchin

25.4.2017

2.9 Graphen

Modellierung von Objekten und ihren Beziehungen zueinander

Anwendungen: Strassennetzwerke, Kommunikationsnetzwerke, Soziale Netzwerke, Prozesse, ...



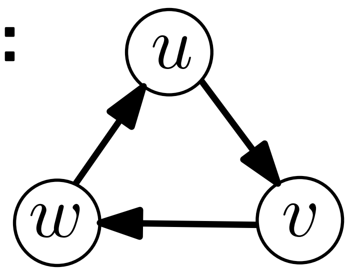
2.9 Graphen

RUB

Modellierung von Objekten und ihren Beziehungen zueinander

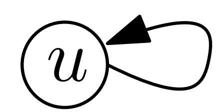
Formal: $G = (V, E)$ mit Knoten V und Kanten $E \subseteq V \times V$

Beispiele:

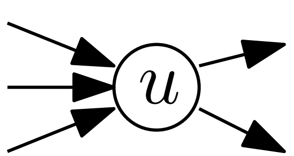


$G = (V, E)$ mit $V = \{u, v, w\}$ und
$E = \{(u, v), (v, w), (w, u), \}$
gerichtet ('Digraph')

- wir verwenden immer $n := |V|$ und $m := |E|$
- für Kante $e = (u, v) \in E$ nennen wir u den *Start*- und v den *Zielknoten*
- Kante e ist *inzident* mit u, v, die Knoten u, v sind *adjazent*.
- sogenannte *Schleifen* (u, u) sind i.d.R. verboten



- *Eingangs*- und *Ausgangsgrad* eines Knoten

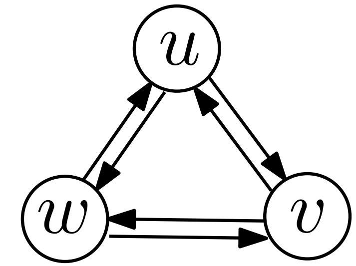


  indegree$(u) := |\{u \in V | (v, u) \in E\}|$
  outdegree$(u) := |\{u \in V | (u, v) \in E\}|$

Graphen

Doppelt-gerichteter Graph:

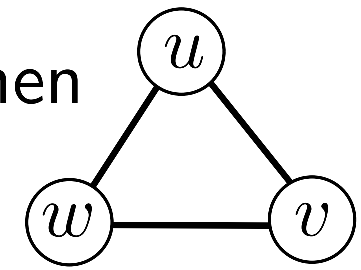
für jede Kante (u, v) ist auch (v, u) vorhanden



Ungerichteter Graph:

einfache Darstellung eines doppelt-gericht. Graphen

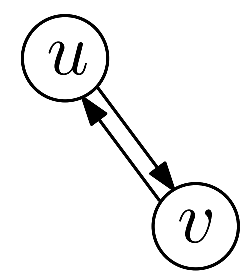
- $\{u, v\}$ statt $(u, v), (v, u)$ und $E = Pow_2(V)$
- degree$(u) := |\{u \in V \mid \{u, v\} \in E\}|$



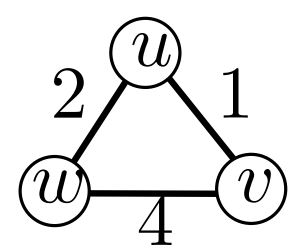
Teilgraph:

$G' = (V', E')$ ist ein Teilgraph von $G = (V, E)$
wenn $V' \subseteq V$ und $E' \subseteq E$

Teilmenge $V' \subseteq V$ induziert einen Teilgraphen $G' = (V', E \cap (V' \times V'))$

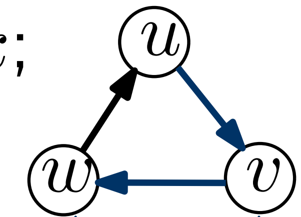


Kantengewichte oder *Kantenkosten* $c : E \to \mathbb{R}$



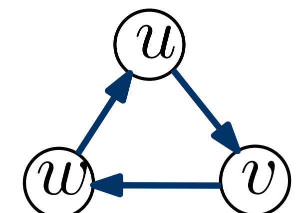
Pfade

Pfad oder *Weg* $p = \langle v_0, \dots, v_k \rangle$ in einem gerichteten Graphen ist eine Knotenfolge mit $(v_i, v_{i+1}) \in E$ für $0 \leq i < k$; k heisst die *Pfadlänge*; p *verläuft* von v_0 nach v_k



$\langle u, v, w \rangle$

p heisst *einfach*, wenn seine Knoten ausser evtl. v_0 und v_k paarweise verschieden sind



$\langle u, v, w, u \rangle$

Kreise oder *Zyklen* sind Wege mit $v_0 = v_k$

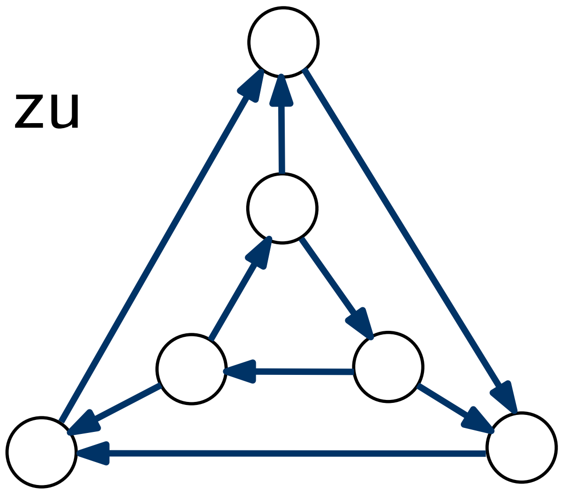
Ein *Hamiltonkreis* ist ein Kreis der alle Knoten von G besucht.

Ein gerichteter Graph ('Digraph') ohne Kreise heisst *azyklisch* oder *kreisfrei*, kurz *DAG*

Zusammenhangskomponenten

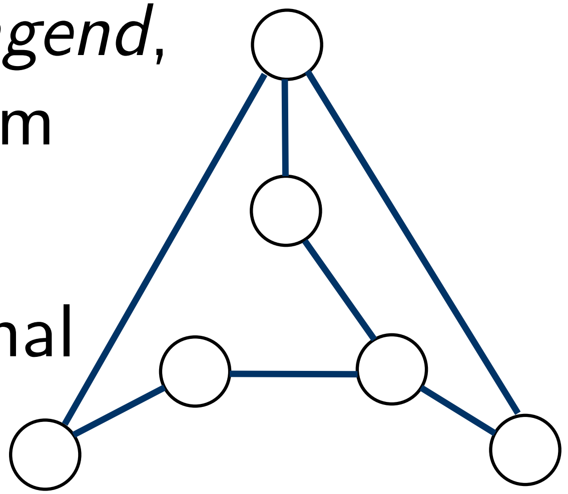
Ein Digraph heisst *stark zusammenhängend*, wenn es von jedem Knoten $u \in V$ einen Weg zu jedem anderen Knoten $v \in V$ gibt.

Eine *starke Zusammenhangskomponente* ist ein maximaler knoteninduzierter starkzusammenhängender Teilgraph von G.



Ein ungerichteter Graph heisst *zusammenhängend*, wenn es von jedem $u \in V$ einen Weg zu jedem anderen Knoten $v \in V$ gibt.

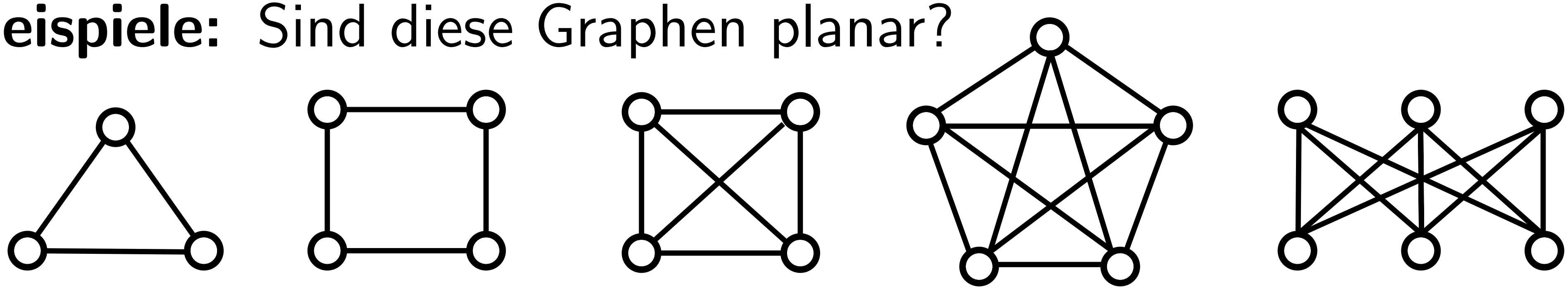
Zusammenhangskomponenten sind die maximal zusammenhängenden Komponenten.



Planarität

Ein Graph heisst *planar*, wenn er gezeichnet werden kann, ohne dass sich Kanten kreuzen.

Beispiele: Sind diese Graphen planar?



2.9.1 Test auf Azyklizität

RUB

Beobachtung: Ein Knoten v mit $\text{outdeg}(v) = 0$ kann nicht auf einem Kreis liegen.

Algorithmus: Streiche sukzessive alle Knoten $v \in V$ mit $\text{outdeg}(v) = 0$. Entweder:

- leerer Graph $\rightarrow$ azyklisch
- nicht-leer; jeder Knoten hat outdeg $> 0 \rightarrow$ enthält Zykel

Korrektheit:

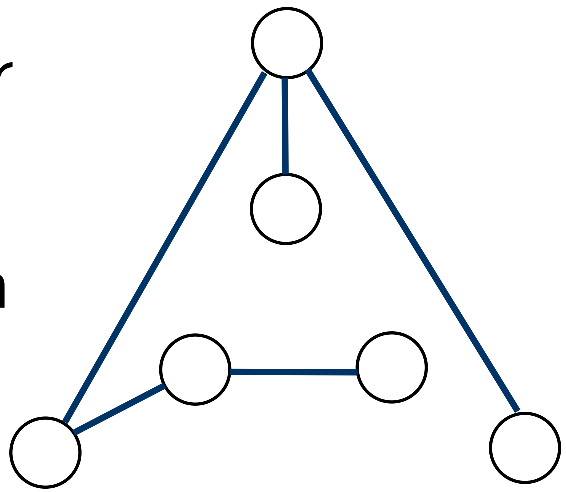
- folgt aus Beobachtung
- Zykel lässt sich leicht finden: starte beliebig bis ein bereits besuchter Knoten betreten wird

Laufzeit: Später werden wir sehen, dass sich dieser Algorithmus bei geschickter Darstellung eines Graphen in $O(n+m)$ Zeit realisieren läßt.

2.9.2 Bäume

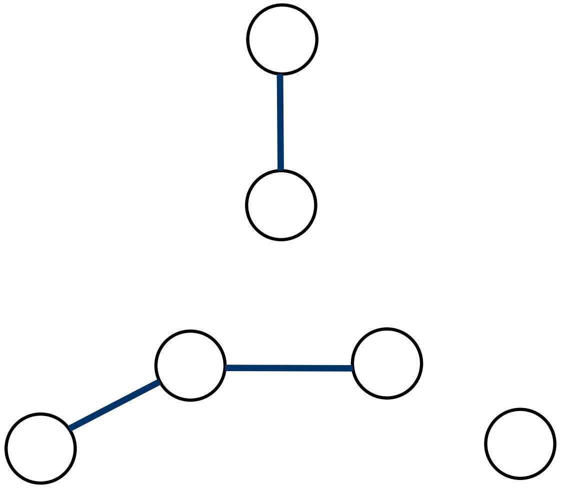
Ein ungerichteter Graph heißt *Baum*, wenn er kreisfrei und zusammenhängend ist.

Äquivalent: wenn es zwischen zwei beliebigen Knoten jeweils genau einen Weg gibt.



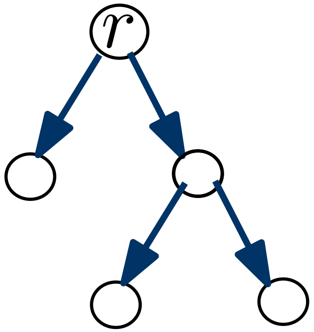
Ein ungerichteter Graph heißt *Wald*, wenn er aus einer Menge von Bäumen besteht.

Äquivalent: wenn es zwischen zwei beliebigen Knoten jeweils höchstens einen Weg gibt.

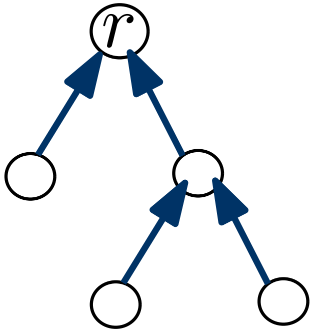


Bäume

Ein gerichteter Graph heisst *Out-Tree* mit Wurzel r wenn es von r aus zu jedem Knoten genau einen Weg gibt.

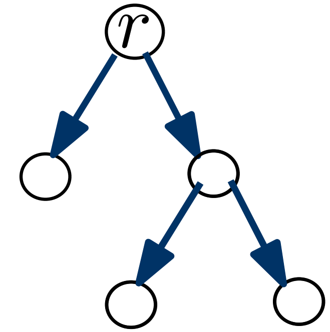


Ein gerichteter Graph heisst *In-Tree* mit Wurzel r wenn nach r hin von jedem Knoten genau einen Weg gibt.

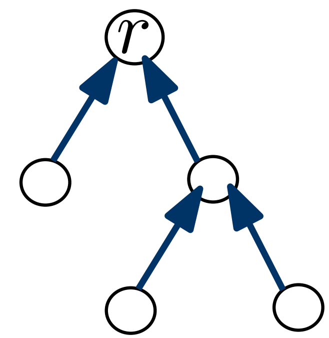


Bäume

Ein gerichteter Graph heisst *Out-Tree* mit Wurzel r wenn es von r aus zu jedem Knoten genau einen Weg gibt.



Ein gerichteter Graph heisst *In-Tree* mit Wurzel r wenn nach r hin von jedem Knoten genau einen Weg gibt.



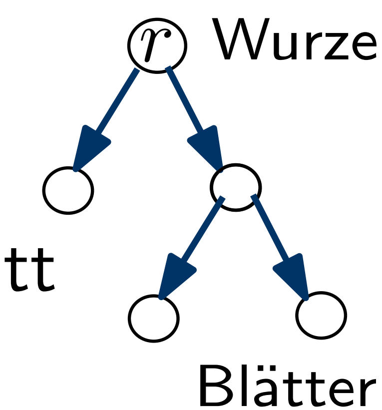
Tiefe(v) $:=$ Länge eines Weges von v zur Wurzel

Tiefe(B) $:= \max\{$Tiefe$(v) | v \in V\}$

Höhe(v) $:=$ Länge eines Weges von v zu einem Blatt

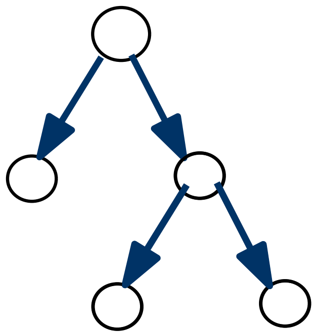
Höhe(B) $:=$ Höhe (Wurzel(B)) $=$ Tiefe(B)

Level oder *Ebene* in einem Baum: Knoten gleicher Tiefe



Binärbaum

Baum mit Grad ≤ 2



Vollständiger Binärbaum: Grad $= 2$ oder 0

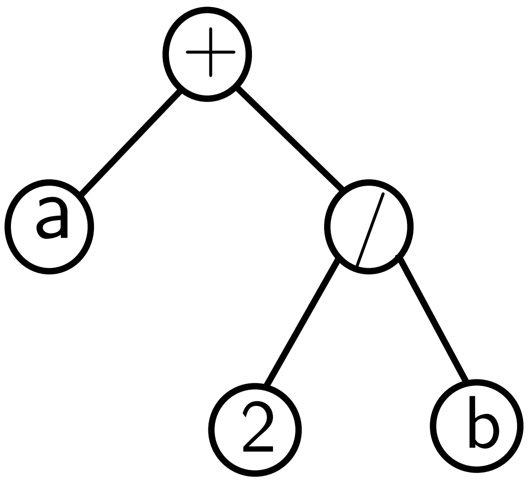
Mit Induktion lässt sich zeigen:

- Jeder Baum auf n Knoten hat $n-1$ Kanten
- Ein vollständiger Binärbaum der Höhe $n-1$ hat $2^n - 1$ Knoten und 2^{n-1} Blätter .

Geordnete Binärbäume

ein Binärbaum heisst *geordnet*, wenn die Kinder jedes inneren Knotens angeordnet sind

Beispiel: arithmetische Ausdrücke

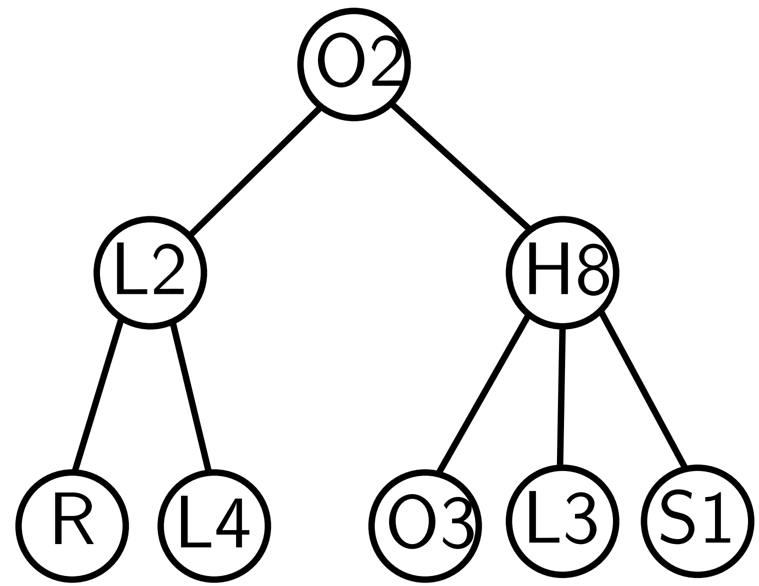


$$a + 2/b$$

Baumdurchläufe

verschiedene Möglichkeiten einen binären Baum zu *durchlaufen*:

- *Tiefensuche*:
  - pre-order
  - in-order
  - post-order
- *Breitensuche*: level-by-level



Beispielhaft:

```
preorder(node v)
    if (v is null) then return
    print v
    preorder (leftchild(v))
    preorder (leftchild(v))
```

Korrektheit: induktiv

Laufzeit: $O(n)$