7. Graphentheorie

Graphen vielseitig verwendbar zur Repräsentation von Zusammenhängen, etwa:

Städte	Verbindungswege
Personen	Relationen zwischen ihnen
Aktionen	zeitliche Abhängigkeiten
...	

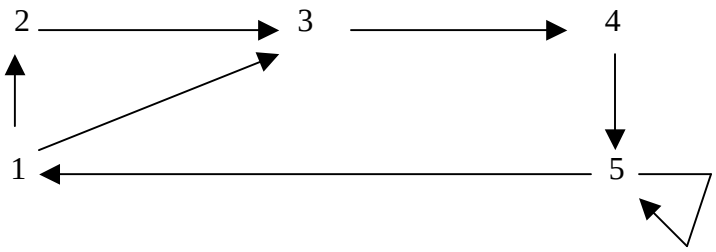
Def. 7.1: Ein gerichteter Graph G = (V, E) besteht aus

1. einer Menge von Knoten (vertices) V,
2. einer Menge von Kanten (edges) $E \subseteq V \times V$.

Gerichtete Graphen werden bildlich repräsentiert, indem man Knoten als Punkte und Kanten als Pfeile vom zugehörigen Anfangs- zum Endpunkt zeichnet.

Beispiel:

V = {1, 2, 3, 4, 5}
E = {(1,2), (1,3), (2,3), (3,4), (4,5), (5,5), (5,1)}



G heißt schlingenfrei, falls $(p,q) \in E$ impliziert $p \neq q$.

Anmerkung: in der Literatur findet sich manchmal auch die folgende Definition:

Def. 7.1a): Ein gerichteter Graph G = (V, A, f) besteht aus

1. einer Menge von Knoten (vertices) V,
2. einer Menge von Kanten A, so dass $V \cap A$ leer,
3. einer Abbildung f: A -> V x V, die jeder Kante ein Paar bestehend aus Anfangs- und Endknoten zuordnet.

Def. 7.1a) lässt mehrere Kanten zwischen denselben Knoten zu!
Da die oft irrelevant sind, wird meist (auch in dieser Vorlesung) die einfachere Definition verwendet!

Def. 7.2: Sei G ein gerichteter Graph.
Eine endliche Folge $(a_1, ..., a_m)$ heißt Weg (oder Pfad) in G, wenn für alle i, $1 \leq i < m$, gilt: $(a_i, a_{i+1}) \in E$.

Ein Weg $(a_1, ..., a_m)$ heißt zyklenfrei, wenn $i \neq j$ impliziert $a_i \neq a_j$. Ein Graph heißt zyklenfrei, wenn alle Wege in ihm zyklenfrei sind.

Repräsentation endlicher Graphen:

a) Adjazenzmatrix:

Speichere Graphen G durch |V| x |V| -Matrix A_G, wobei $a_{ij} = 1$ falls $(v_i,v_j) \in E$, 0 sonst.

Matrix für obiges Beispiel:

0	1	1	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1
1	0	0	0	1

b) Adjazenzlisten:

Listen L_i, $i \in \{1, ..., |V|\}$, L_i enthält alle Nachfolger von v_i.

für obiges Beispiel:

L_1: (2,3)
L_2: (3)
L_3: (4)
L_4: (5)
L_5: (5,1)

Oft sparsamer als Matrix.

Def.7.3: Ein ungerichteter Graph G = (V, E) besteht aus

1. einer Menge von Knoten V,
2. einer Menge von Kanten $E \subseteq \{\{p,q\} \mid p, q \in V\}$.

Def. 7.4: Sei G ein ungerichteter Graph.
a) Eine endliche Folge $(a_1, ..., a_m)$ heißt Weg (oder Pfad) in G, wenn für alle i, $1 \leq i < m$, gilt: $\{a_i, a_{i+1}\} \in E$.
b) Ein ungerichteter Graph heißt zusammenhängend, wenn gilt: p, q in V, $p \neq q$ impliziert es gibt einen Weg von p nach q.

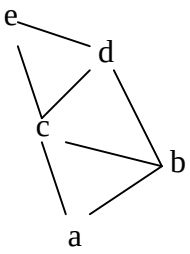
c) Ein gerichteter Graph G = (V, E) heißt zusammenhängend, wenn der zugehörige ungerichtete Graph G’ = (V’,E’), mit V’ = V und E’ = {{p,q} | (p,q) ∈ E} zusammenhängend ist.

Def.7.5: Sei G = (V, E) ein ungerichteter Graph. Der ungerichtete Graph G' = (V',E') heißt

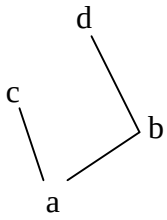
- Teilgraph von G gdw. V' ⊆ V und E' ⊆ E,
- Untergraph von G gdw. V' ⊆ V und E' = {{p,q} ∈ E | p, q ∈ V'}.

Def. für gerichtete Graphen analog.

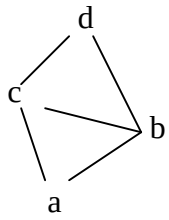
Beispiel:



G

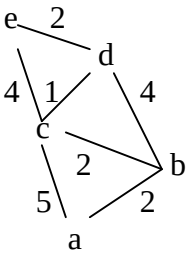


Teilgraph von G



Untergraph von G

Def.: Ein kantenbewerteter Graph ist ein ungerichteter Graph G = (V, E) mit einer Wertungsfunktion w: E -> R_+, die jeder Kante eine positive reelle Zahl als Kosten zuordnet.



G

Häufig gesucht: kürzester Weg, d.h. Weg von x nach y mit geringsten Gesamtkosten

Dijkstra-Algorithmus:
(Edsger Wybe Dijkstra, 1930 - 2002, einer der bedeutendsten Informatiker)

Sei G kantenbewerteter Graph mit n Knoten, u einer der Knoten.
gesucht: Teilgraph mit kürzesten Wegen von Knoten u zu von u aus erreichbaren Knoten
W: Liste der noch zu behandelnden Knoten
F: Liste von Kanten, die auf kürzestem Weg von u zu anderen Knoten liegen
l(v): kürzeste bisher gefundene Weglänge von u nach v

k(v): optimale zu v führende Kante

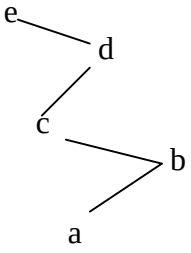
```
for v ∈ V do if {u,v} ∈ E then l(v) := w({u,v}); k(v) := {u,v} else l(v) := ∞;
W := V;
F := ∅;
l(u) := 0;
for i := 1 to n do
        finde einen Knoten v ∈ W, so dass l(v) minimal;
        W := W \{v};
        if v ≠ u und l(v) ≠ ∞ then F := F ∪ {k(v)};
        for alle Nachbarknoten v' von v mit v' in W do
                if l(v) + w({v,v'}) < l(v') then
                        l(v') :=  l(v) + w({v,v'});
                        k(v') := {v,v'};
```

Folge der für Beispielgraphen und u=a erzeugten Variablenbelegungen:

l(a): ∞, 0	k(a):
l(b): 2	k(b): {a,b}
l(c): 5, 4	k(c): {a,c}, {b,c}
l(d): ∞, 6, 5	k(d): {b,d}, {c,d}
l(e): ∞, 8, 7	k(e): {c,e}, {d,e}

v: a, b, c, d, e
W: {a,b,c,d,e}, {b,c,d,e},{c,d,e},{d,e},{e}, ∅
F: ∅, {{a,b}}, {{a,b},{b,c}}, {{a,b},{b,c}, {c,d}}, {{a,b},{b,c}, {c,d}, {d,e}}

Teilgraph (V,F) mit kürzesten Wegen von a aus:



Korrektheitsbeweis:
nach i Schleifendurchgängen sind die Längen von i Knoten, die am nächsten an u liegen, korrekt berechnet und diese Knoten sind aus W entfernt.
Induktionsanfang:
beim 1. Schleifendurchlauf wird u gewählt, l(u) = 0.
Induktionsschritt:
Nimm an, v wird im i+1 ten Durchlauf aus W genommen.
Der kürzeste Pfad zu v gehe über Vorgänger v' von v.
Da v' näher an u liegt, ist v' nach Induktionsvoraussetzung nach i Durchläufen mit richtiger Länge bereits entfernt. Da der kürzeste Weg zu v die Länge l(v') + w(v'.v) hat und dieser Wert bei Entfernen von v' bereits v zugewiesen wurde, wird v mit der richtigen Länge entfernt und die korrekte Kante in F eingefügt.

Def.: Sei G = (V, E) ein gerichteter Graph, v ∈ V. Wir definieren

indeg(v) = | {(x,y) ∈ E | y = v}| Eingangsgrad
outdeg(v) = | {(x,y) ∈ E | x = v}| Ausgangsgrad

v' heißt Nachfolger von v in G gdw. (v,v') ∈ E.

Def.: Sei G = (V, E) ein gerichteter Graph: G heißt Baum falls gilt:

1. der G entsprechende ungerichtete Graph G' ist zyklenfrei,
2. es gibt s∈ V, so dass indeg(s) = 0, und für alle v ≠ s: indeg(v) = 1

Der Knoten s mit indeg(s) = 0 heißt Wurzel des Baums
Knoten v mit outdeg(v) = 0 heißen Blätter.
Knoten, die nicht Blätter sind, heißen innere Knoten.

Anmerkung: in einem Baum gibt es genau 1 gerichteten Weg von der Wurzel zu jedem anderen Knoten.

Die Tiefe eines Knotens v in einem Baum G ist die Länge (= Anzahl der Kanten) des gerichteten Weges von der Wurzel zu v.
Die Tiefe eines Baumes G ist die Länge des längsten gerichteten Weges in G.
Die Ordnung eines Baumes ist die maximale Anzahl der Nachfolger eines Knotens.

Def.: Ein Baum G = (V, E) der Ordnung 2 heißt Binärbaum.

Def.: Sei G ein Baum der Ordnung k mit Tiefe m. G heißt vollständig, wenn es keinen Baum der Ordnung k mit Tiefe m gibt, der mehr Knoten als G besitzt.

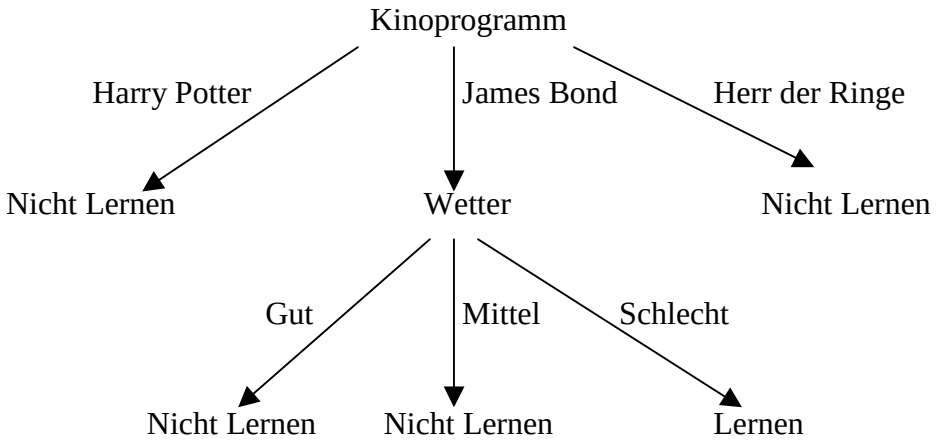
Def.: Ein Baum heißt geordnet, wenn die Nachfolger eines jeden Knotens geordnet sind (1. , 2. 3. Nachfolger usw., im Fall von Binärbäumen linker, rechter Nachfolger).

Def.: Sei G = (V, E) ein Baum, M eine Menge von Markierungen. G heißt kantenmarkiert, wenn es eine Abbildung m: E -> M gibt, die jeder Kante eine Markierung aus M zuordnet. (z.B. Kosten).

Bäume tauchen immer wieder auf in der Informatik: Beispiele

a) Ableitungsbäume in der Logik
b) Syntaxbäume
c) Entscheidungsbäume
d) Suchbäume beim Problemlösen

zu c) Innere Knoten entsprechen Attributen, Kanten von A nach B sind mit Werten des Attributs A markiert. Blätter entsprechen Aktionen, z.B.:



zu d) Sei S eine Menge von Zuständen, wobei ein Zustand z.B. beschrieben sein kann durch Menge von aussagenlogischen Formeln

Ein Problem ist charakterisiert durch

Anfangszustand s0
Menge von Zielzuständen G
Operatoren, die bei Vorliegen bestimmter Anwendbarkeitsbedingungen Zustände in Nachfolgezustände überführen.

Gesucht: Folge von Operatoren, die s0 in einen der Zielzustände überführt.

Erzeuge schrittweise Suchbaum: markierter Baum mit Wurzel s0, Kante von si zu sj mit Markierung op falls op anwendbar in si und op erzeugt sj aus si.

Beispiel STRIPS-Planen

Zustände: Mengen von atomaren Formeln
Operator op:
a) Vorbedingungen pre: Atome, die in s sein müssen, damit op angewendet werden kann
b) Delete Liste del: Atome, die im Nachfolgezustand nicht mehr enthalten sind
c) Add Liste add: Atome, die in den Nachfolgezustand aufgenommen werden

falls pre $\subseteq$ s, dann erzeugt op aus s den Zustand op(s) = (s\del) $\cup$ add

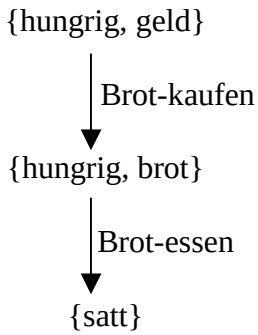
s0: hungrig, geld
G = {s | satt $\in$ s}

Operationen:

Brot-kaufen:
pre: geld
del: geld
add: brot

Brot-essen:
pre: brot
del: brot, hungrig
add: satt

Suchbaum:



Anmerkung: Planen ist nicht immer so einfach!!