# 7. Graphalgorithmen

## 7.0 Überblick, Definitionen

Erinnerung: Wir betrachten hier nur endliche Graphen ohne Mehrfachkanten. Graphen bestehen aus einer Knotenmenge V und einer Kantenmenge E (englisch: Knoten = **v**ertex oder node, Kante = **e**dge).

G = (V, E) heißt ungerichteter Graph $\Leftrightarrow$
1. V ist ein endliche nicht-leere Menge,
2. $E \subseteq \{ \{x, y\} \mid x, y \in V, x \neq y \} \cup \{ \{x\} \mid x \in V \}$.

G = (V, E) heißt gerichteter Graph (oder Digraph) $\Leftrightarrow$
1. V ist ein endliche nicht-leere Menge,
2. $E \subseteq V \times V$.

[Digraph kommt von "directed graph" = gerichteter Graph.]



Erinnerung: Gerichtete / ungerichtete Version eines Graphen

Es sei G = (V, E) ein ungerichteter Graph. Der gerichtete Graph $G_{ger} = (V, E_{ger})$ mit
$E_{ger} = \{(x,y), (y,x) \mid \{x, y\} \in E\} \cup \{ (x,x) \mid \{x\} \in E \}$
heißt gerichtete Version des Graphen G.

Es sei G = (V, E) ein gerichteter Graph. Der ungerichtete Graph $G_{ung} = (V, E_{ung})$ mit
$E_{ung} = \{\{x,y\} \mid (x,y) \in E \text{ oder } (y,x) \in E\} \cup \{\{x\} \mid (x,x) \in E \}$
heißt ungerichtete Version des Graphen G.

Im ungerichteten Fall gehen wir also zu beiden Richtungen über, im gerichteten Fall ignorieren wir die Richtung.

Ein gerichteter Graph H heißt Orientierung oder Ausrichtung des ungerichteten Graphen G, wenn G die ungerichtete Version von H ist.

Erinnerung: Teilgraph, induzierter Teilgraph

Es sei G = (V, E) ein ungerichteter bzw. gerichteter Graph. Ein ungerichteter bzw. gerichteter Graph G' = (V',E') heißt Teilgraph von G, wenn $V' \subseteq V$ und $E' \subseteq E$ gilt.

Es sei G = (V, E) ein ungerichteter bzw. gerichteter Graph und es sei $V' \subseteq V$. Der ungerichtete bzw. gerichteter Graph G' = (V',E') heißt der von V' induzierte Teilgraph von G, wenn im gerichteten Fall

$$E' = \{\{x,y\} \mid \{x,y\} \in E \text{ und } x,y \in V'\}$$

bzw. im gerichteten Fall

$$E' = \{(x,y) \mid (x,y) \in E \text{ und } x,y \in V'\}$$

gilt.



Erinnerung: neighbour, successor, predecessor

Jede Kante {x, y} bzw. (x,y) heißt inzident zu ihren Knoten x und y. Zwei Knoten x, y mit $\{x,y\} \in E$ (ungerichteter Fall) bzw. $(x,y) \in E$ oder $(y,x) \in E$ (gerichteter Fall) heißen adjazent.

*Ungerichteter Fall:* Die Menge $N(x) = \{ y \in V \mid \{x,y\} \in E \}$ heißt die Menge der Nachbarn (oder der Nachfolger) von x.

*Gerichteter Fall:* $N(x) = \{ y \in V \mid (x,y) \in E \text{ oder } (y,x) \in E\}$ heißt die Menge der Nachbarn von x.
$S(x) = \{ y \in V \mid (x,y) \in E\}$ heißt Menge der Nachfolger von x,
$P(x) = \{ y \in V \mid (y,x) \in E\}$ heißt Menge der Vorgänger von x.

Eine Kante {x} im ungerichteten Fall bzw. (x,x) im gerichteten Fall heißt Schlinge.

Erinnerung: Grad d, Eingangs-, Ausgangsgrad, geordnet

*Ungerichteter Fall:* Für einen Knoten x heißt
$d(x) = |\{ y \in V \mid x \neq y, \{x, y\} \in E \}|$, falls $\{x\} \notin E$ bzw.
$d(x) = |\{ y \in V \mid x \neq y, \{x, y\} \in E \}| + 2$, falls $\{x\} \in E$
der (Knoten-) Grad von x ("degree"). Der maximale Knotengrad heißt Grad d(G) des Graphen G.

*Gerichteter Fall:* Für einen Knoten x heißt
$d^+(x) = |\{ y \in V \mid (x, y) \in E \}|$ der Ausgangsgrad und
$d^-(x) = |\{ y \in V \mid (y,x) \in E \}|$ der Eingangsgrad von x. Der Wert
$d(x) = d^+(x) + d^-(x)$ heißt auch der Grad von x.

Ein Graph heißt geordnet, wenn für jeden Knoten x im ungerichteten Fall die Menge der Nachbarn N(x) bzw. im gerichteten Fall die Menge der Nachfolger S(x) geordnet ist (und damit sind zugleich die zugehörigen Kanten geordnet).



Erinnerung: Weg, geschlossener Weg, doppelpunktfrei / einfach

*Ungerichteter Fall:* Eine Folge von Knoten $(x_0, x_1, ..., x_k)$ heißt Weg der Länge k von $x_0$ nach $x_k$, wenn $x_{i-1}$ und $x_i$ adjazent sind für i=1,..., k.

*Gerichteter Fall:* Eine Folge von Knoten $(x_0, x_1, ..., x_k)$ heißt Weg der Länge k von $x_0$ nach $x_k$, wenn $(x_{i-1}, x_i) \in E$ für i=1,..., k.

Ein Weg $(x_0, x_1, ..., x_k)$ heißt doppelpunktfrei, wenn $x_i \neq x_j$ für alle $1 \leq i < j \leq k-1$ gilt. (Beachte: Der letzte Knoten darf gleich dem ersten Knoten sein.) Statt "doppelpunktfrei" nennt man solche Wege in der Literatur auch "einfache Wege".

Ein Weg $(x_0, x_1, ..., x_k)$ heißt geschlossen, wenn $x_0 = x_k$ gilt.

Erinnerung: Kantenzug, Hamiltonscher, Eulerscher Weg

Ein Weg heißt Kantenzug, wenn je zwei seiner Kanten $(x_{i-1}, x_i)$ und $(x_{j-1}, x_j)$ für $i \neq j$ verschieden sind.

Ein Weg $(x_0, x_1, ..., x_{n-1})$ in einem Graphen G mit n Knoten heißt Hamiltonscher Weg, wenn er jeden Knoten genau einmal enthält. Ein Weg $(x_0, x_1, ..., x_{n-1}, x_0)$ heißt Hamiltonscher Kreis, wenn $(x_0, x_1, ..., x_{n-1})$ ein Hamiltonscher Weg ist.

Ein Weg $(x_0, x_1, ..., x_k)$ heißt Eulerscher Weg, wenn jede Kante des Graphen genau einmal in ihm vorkommt. Er heißt Eulerscher Kreis, wenn zusätzlich $x_k = x_0$ ist.

[Ein ungerichteter Graph besitzt einen Eulerschen Kreis genau dann, wenn jeder Knoten einen geraden Grad hat. (vgl. Kap. 3)]



Erinnerung: Zyklus/Kreis, azyklisch, DAG

Ein doppelpunktfreier Weg $(x_0, x_1, ..., x_k)$ heißt Zyklus oder Kreis, wenn $k \geq 3$ ist und $x_0 = x_k$ gilt.

Ein ungerichteter Graph heißt kreisfrei oder zyklenfrei oder azyklisch, wenn er keine Schlingen und keinen Zyklus besitzt.

Ein gerichteter Graph heißt kreisfrei oder zyklenfrei oder azyklisch, wenn er keine doppelpunktfreien geschlossenen Wege besitzt. [Der Unterschied zum ungerichteten Fall liegt in den geschlossenen Wegen der Länge 2, die wir hier explizit ausschließen müssen.]

Ein gerichteter azyklischer Graph wird auch als DAG bezeichnet ("directed acyclic graph").

Definition: Abstand/Distanz, Durchmesser

Der Abstand oder die Distanz dist(x,y) eines Knotens x zum Knoten y ist die Länge des kürzesten Weges von x nach y.

Formal: dist: $V \times V \rightarrow \mathbb{N}_0 \cup \{\infty\}$ mit

dist(x,y) = 0, für x=y,
dist(x,y) = k, es gibt einen Weg der Länge k von x nach y, aber keinen solchen Weg der Länge k-1,
dist(x,y) = $\infty$, es gibt keinen Weg von x nach y.

Die maximale Distanz in einem Graphen bezeichnet man auch als den Durchmesser des Graphen.

*Hinweis:* Wenn die Kanten Entfernungen besitzen (siehe unten: markierte Graphen), so bezeichnet man meist die Länge des kürzesten Weges als den Abstand zwischen zwei Knoten.



Erinnerung: Zusammenhang, Zusammenhangskomponenten

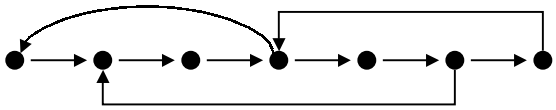
*Ungerichteter Fall:* Ein Graph G = (V, E) heißt zusammenhängend, wenn es für je zwei verschiedene Knoten x und y einen Weg von x nach y gibt.

Es sei x ein Knoten. Der Teilgraph G(x) = (Z(x), E'(x)) mit
$Z(x) = \{y \in V \mid$ Es gibt einen Weg von x nach y in $G\} \cup \{x\}$,
$E'(x) = \{\{y_1,y_2\} \mid \{y_1,y_2\} \in E$ und $y_1,y_2 \in Z(x)\}$
heißt Zusammenhangskomponente (des Knotens x) von G.
Hierfür genügt es, die Knotenmenge Z(x) anzugeben, so dass man auch die Menge Z(x) als Zusammenhangskomponente bezeichnet.
Für alle $y \in Z(x)$ gilt $Z(x) = Z(y)$. Jeder Graph lässt sich eindeutig in seine Zusammenhangskomponenten zerlegen.

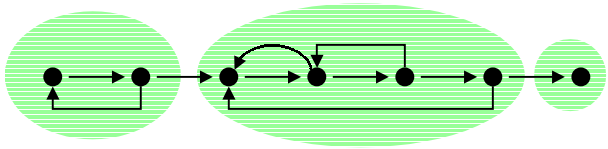
Erinnerung: Zusammenhang, starker Zusammenhang

*Gerichteter Fall:* Ein gerichteter Graph G = (V, E) heißt zusammenhängend, wenn seine ungerichtete Version $G_{ung}$ zusammenhängend ist.

Ein gerichteter Graph heißt stark zusammenhängend, wenn es für je zwei verschiedene Knoten x und y einen (gerichteten!) Weg von x nach y gibt.



Dieser Graph ist stark zusammenhängend.



Dieser Graph besitzt drei starke Zusammenhangs-komponenten.



Definition: starke Zusammenhangskomponente

*Gerichteter Fall:*

Es sei x ein Knoten. Der Teilgraph G(x) = (SZ(x), E'(x)) mit
SZ(x) = {y∈ V | Es gibt einen Weg von x nach y in G} ∪ {x},
E'(x) = {$(y_1,y_2)$ | $(y_1,y_2)$∈ E und $y_1,y_2$∈ SZ(x)}
heißt starke Zusammenhangskomponente (des Knotens x) von G. Hierfür genügt es, die Knotenmenge Z(x) anzugeben, so dass man auch die Menge Z(x) als starke Zusammenhangs-komponente bezeichnet.
Für alle y∈ SZ(x) gilt SZ(x) = SZ(y).
Jeder gerichtete Graph lässt sich eindeutig sowohl in seine Zusammenhangskomponenten als auch in seine starken Zusammenhangskomponenten zerlegen.

Definition: Transitive Hülle, topologische Sortierung von DAGs

Zu einem Graphen G = (V, E) heißt G* = (V, E*) die transitive Hülle, wenn im ungerichteten Fall gilt
$E^* = \{\{x,y\} \mid x \neq y$ und es gibt einen Weg von x nach y in $G\} \cup E$
bzw. im gerichteten Fall gilt
$E^* = \{(x,y) \mid x \neq y$ und es gibt einen Weg von x nach y in $G\} \cup E$.

Es sei G=(V,E) gerichtet. Eine Abbildung ord: $V \rightarrow \mathbb{N}$ mit
$\forall\, x,y \in V$ mit $x \neq y$ gilt: $(x,y) \in E^* \Rightarrow ord(x) < ord(y)$
heißt topologische Sortierung von G.
[Man ordnet also die Knoten so an, dass jeder von x aus erreichbare Knoten auch eine höhere Nummer als x bekommt.]

Jeder DAG besitzt eine topologische Sortierung; sie ist nicht eindeutig, siehe Abschnitt 7.2.



Definition: Vollständiger Graph $K_n$, einfache Kreise $C_n$

Ein Graph G = (V, E) heißt vollständig, wenn
$E = \{\{x, y\} \mid x, y \in V, x \neq y\}$ bzw. im gerichteten Fall
$E = \{(x, y) \mid x, y \in V, x \neq y\}$.

Ein vollständiger Graph mit n Knoten ist bis auf Umbenennung eindeutig, so dass man ihn mit dem Symbol $K_n$ bezeichnet. [Ein vollständiger Graph besitzt $n \cdot (n-1)/2$ bzw. $n \cdot (n-1)$ Kanten, $n \geq 1$.]

Ein Graph G = (V, E) mit $n \geq 3$ Knoten heißt einfacher Kreis, wenn $V = \{x_0, x_1, x_2, ..., x_{n-1}\}$ und im ungerichteten Fall
$E = \{\{x_{i-1},x_i\} \mid i=1, 2, ..., n-1\} \cup \{\{x_{n-1},x_0\}\}$
bzw. im gerichteten Fall
$E = \{(x_{i-1},x_i) \mid i=1, 2, ..., n-1\} \cup \{(x_{n-1},x_0)\}$ ist.
Auch diese Graphen sind bis auf Umbenennung eindeutig; man bezeichnet sie mit $C_n$.

Erinnerung: Markierte oder gewichtete Graphen

In Anwendungen sind Graphen meist "markiert" oder "gewichtet", d.h., ihre Knoten und / oder ihre Kanten sind mit Werten ("Markierung" oder "Gewicht") aus einer Wertemenge W bzw. W' versehen:
$\mu: V \rightarrow W$ und $\delta : E \rightarrow W'$. Meist sind die Markierungen ganze oder reelle Zahlen. Man schreibt dann $G = (V, E, \mu)$ bzw. $G = (V, E, \delta)$ bzw. $G = (V, E, \mu, \delta)$.

Kantenmarkierungen $\delta : E \rightarrow W'$ mit
W' = Menge der nichtnegativen reellen Zahlen
bezeichnet man auch als Entfernungen.

[Die Summe aller Entfernungen heißt das Gewicht des Graphen, siehe unten bei "minimaler Spannbaum".]



Definition: Länge von Wegen, Abstand zwischen Knoten

Eine reellwertige Abbildung $\delta : E \rightarrow \mathbb{R}$ wird auf die Menge der Wege fortgesetzt durch
$\delta((x_0, x_1, ..., x_k)) := \delta((x_0, x_1)) + \delta((x_1, x_2)) + ... + \delta((x_{k-1}, x_k))$.
Dieser Wert heißt auch die Länge des Weges $(x_0, x_1, ..., x_k)$.

In solchen Graphen bezeichnet man als den Abstand eines Knotens x zum Knoten y den Wert $\delta: V \times V \rightarrow \mathbb{N}_0 \cup \{\infty\}$ mit

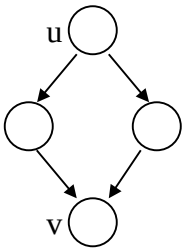
$\delta(x,y) = 0$, für $x=y$,
$\delta(x,y) = \text{Min} \{\delta((x_0, x_1, ..., x_k)) \mid x = x_0, y = x_k\}$, sofern es mindestens einen Weg von x nach y gibt,
$\delta(x,y) = \infty$, falls es keinen Weg von x nach y gibt.

Die Aufgabe, den Abstand $\delta(x,y)$ vom Knoten x zum Knoten y oder einen Weg von x nach y mit der Länge $\delta(x,y)$ zu ermitteln, bezeichnet man als das Kürzeste-Wege-Problem.

Erinnerung: Wurzel

Es sei G ein gerichteter oder ungerichteter Graph ohne Schlingen. Ein Knoten w von G heißt Wurzel, wenn es von w zu jedem anderen Knoten des Graphen *genau einen* doppelpunktfreien Weg gibt.

Nach dieser Definition ist im einfachen gerichteten (!) Kreis $C_n$ jeder Knoten Wurzel. Man verwendet den Begriff der Wurzel meist nur im Zusammenhang mit azyklischen Graphen.




In dem nebenstehenden Beispiel ist der Knoten u keine Wurzel, da es zwei verschiedene Wege zum Knoten v gibt.



Erinnerung: Baum, Wald

Ein Graph G = (V, E) heißt Baum, wenn er azyklisch ist und eine Wurzel w besitzt.

Ein Graph G = (V, E) heißt Wald, wenn jede seiner Zusammenhangskomponenten ein Baum ist.

Es gilt: In jedem gerichteten Wald muss es Knoten ohne Vorgänger geben (d.h. Knoten mit Eingangsgrad 0).

Ein ungerichteter zusammenhängender Graph G=(V,E) mit n =|V| und m =|E| ist genau dann ein Baum, wenn m=n-1 gilt.

Im ungerichteten Fall gilt: In einem Baum ist jeder Knoten Wurzel.
Im gerichteten Fall ist die Wurzel eines Baums eindeutig bestimmt (nur sie besitzt den Eingangsgrad 0).

Erinnerung: Blatt, innerer Knoten, binärer Baum

Es sei G ein ungerichteter Baum. Ein Knoten x von G heißt Blatt, wenn x genau einen Nachbarn y≠x besitzt.

Es sei G ein gerichteter Baum. Ein Knoten x von G heißt Blatt, wenn x keinen Nachfolger besitzt.

Die Knoten, die nicht Blätter sind, heißen innere Knoten des Baums.

Die Begriffe "Grad", "Ausgangsgrad", "geordnet" gelten auch auf für Bäume. Ein gerichteter geordneter Baum, dessen Ausgangsgrad höchstens 2 ist und in dem jeder Nachfolger eines inneren Knotens entweder die Nummer 1 oder 2 besitzt (hat ein innerer Knoten zwei Nachfolger, so müssen diese verschiedene Nummern besitzen), heißt binärer Baum.



Erinnerung: Eltern-, Kindknoten, Level, Tiefe/Höhe h(G)

Im gerichteten Fall ist die Wurzel eindeutig, im ungerichteten Fall wähle man für die Definitionen auf dieser Folie eine Wurzel w fest aus.

In einem Baum mit Wurzel w gibt es genau einen Weg von w zu jedem Knoten. Daher besitzt jeder Knoten x≠w genau einen Vorgängerknoten, genannt Elternknoten (oder Vaterknoten oder Mutterknoten). Die (übrigen) Nachfolgerknoten heißen die Kinder (oder Söhne oder Töchter) von x.

Die um 1 erhöhte Länge des Weges von w nach x heißt Level (oder "Tiefe" oder "Höhe" oder "Niveau") des Knotens x im Baum. Die Wurzel besitzt also das Level 1.

Die um 1 erhöhte Länge des längsten Weges von der Wurzel zu einem Blatt heißt die Höhe h(G) des Baums (auch Tiefe des Baums genannt). h(G) ist also das größte Level im Baum.

Definition: Spannbaum, Gewicht, minimaler Spannbaum

Sei G=(V,E) ein zusammenhängender gerichteter oder ungerichteter Graph. Ein (gerichteter bzw. ungerichteter) Teilgraph $B=(V,E_B)$, der ein Baum ist, heißt Spannbaum oder aufspannender oder spannender Baum von G (beachte: in B kommen alle Knoten des Graphen vor).

Es sei $G=(V,E,\delta)$ ein Kanten markierter Graph mit $\delta: E \rightarrow \mathbb{R}$. Die Summe aller seiner Entfernungen $\delta(G)$

$$\delta(G) := \sum_{e \in E} \delta(e)$$

heißt das Gewicht des Graphen G.

Ein Spannbaum $B=(V,E_B,\delta)$ des Graphen $G=(V,E,\delta)$ heißt minimaler Spannbaum von G, wenn $\delta(B) \leq \delta(B')$ für alle Spannbäume B' von G gilt.



Erinnerung: Darstellung von Graphen (vgl. Kapitel 3)

Graphen kann man durch ihre Adjazenzmatrix A, durch Adjazenzlisten oder durch Inzidenzlisten darstellen.

Es sei $G = (V,E)$ mit $V = \{x_1, x_2, ..., x_n\}$ ein Graph.

*Ungerichteter Fall:* Die Adjazenzmatrix $A = (a_{i,j})$ ist definiert durch $a_{i,j} = 1$, falls $\{x_i, x_j\} \in E$, und $a_{i,j} = 0$ sonst (i, j = 1, ..., n).

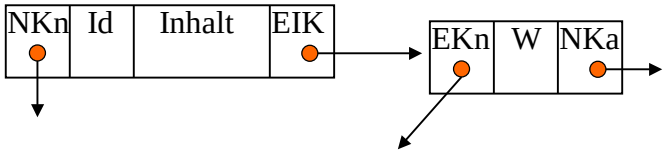
*Gerichteter Fall:* Die Adjazenzmatrix $A = (a_{i,j})$ ist definiert durch $a_{i,j} = 1$, falls $(x_i, x_j) \in E$, und $a_{i,j} = 0$ sonst (i, j = 1, ..., n).

Die *erweiterte Adjazenzmatrix* A' ist für $i \neq j$ gleich der Adjazenzmatrix, allerdings wird die Hauptdiagonale auf 1 gesetzt, d.h. $a'_{i,j} = a_{i,j}$ für $i \neq j$ und $a'_{ii} = 1$ (für i, j = 1, ..., n).

Datentyp der Graphen: (vgl. Skript Plödereder, Folie 338)

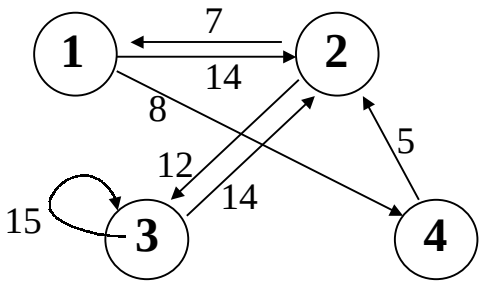
Jeder Knoten erhält einen Identifikator "Knoten_Index" KI.
In der Regel ist dies eine natürliche Zahl.
Die Knoten werden in Listen zusammengefasst und besitzen daher neben Index und Inhalten noch weitere Komponenten:
- Verweis auf den **N**ächsten **K**noten in der Liste "NKn"
- Verweis auf die **E**rste **I**nzidente **K**ante "EIK".

Die von einem Knoten ausgehende Kante muss enthalten den "Endknoten der Kante" (EKn), ihr Gewicht W ("weight") und einen Verweis auf die nächste Kante "NKa".



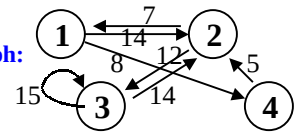


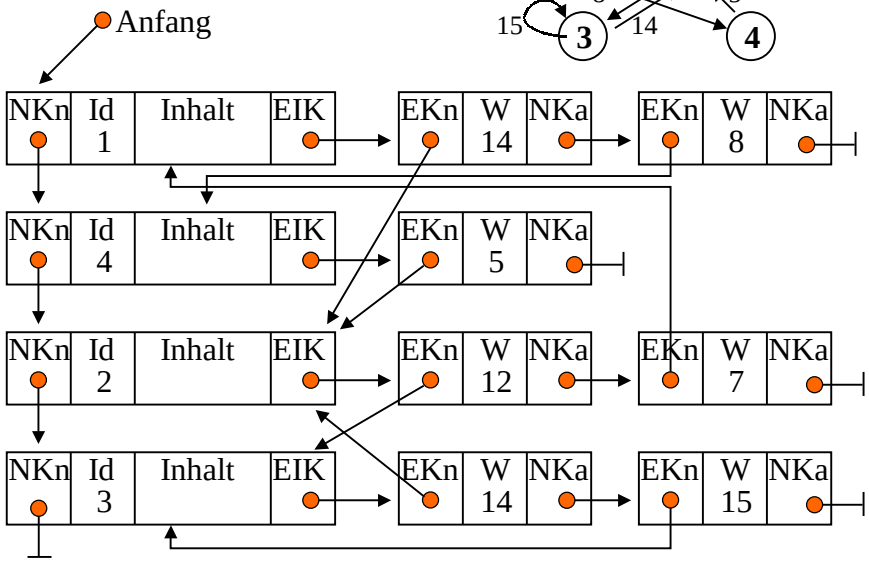

## Beispiel: Gerichteter gewichteter Graph:



**Beispiel**

Zugehöriger Graph:






Die meisten Anwendungen notieren gewisse Informationen in den Graphen. Oft merkt man sich zwei Zahlen, die Ergebnisse von Durchlaufprozeduren sind, sowie einen Booleschen Wert "besucht", der angibt, ob dieser Knoten bereits zuvor erreicht ("besucht") worden ist. Wir fügen diese Komponenten zu den Knoten hinzu und erhalten folgende Datentypen, formuliert in Ada:

```
type NextKnoten is access Knoten;
type NextKante is access Kante;

type Knoten is record
        Id: Knoten_Index;
        besucht: Boolean; zahl1, zahl2: Integer;
        Inhalt: <weitere Komponenten>;
        NKn: NextKnoten;
        EIK: NextKante;
        end record;

type Kante is record
        W: <Typ des Gewichts>;
        EKn: NextKnoten;
        NKa: NextKante;
        end record;
```



## 7.1 Durchsuchen eines Graphens

Ein gerichteter oder ungerichteter Graph soll unter Beachtung seiner Kanten durchlaufen werden. *Gesucht* ist also ein Algorithmus, der mit irgendeinem Knoten x beginnt und dann alle von x aus erreichbaren Knoten und Kanten besucht.
Im ungerichteten Fall wird hierbei die zu x gehörige Zusammenhangskomponente G(x) = (Z(x), E'(x)) durchlaufen.
Im gerichteten Fall wird der von x aus erreichbare Teil der starken Zusammenhangskomponente (speziell: alle Knoten y mit $(x,y) \in E^*$, siehe transitive Hülle) besucht. Werden auf diese Weise nicht alle Knoten des Graphen erreicht, so wird dieses Vorgehen mit irgendeinem bisher noch nicht besuchten Knoten fortgesetzt.

Ausgehend von einem Knoten x werden die Knoten und Kanten meist nach zwei Strategien aufgesucht, die uns bereits bei Bäumen begegnet sind:

Tiefensuche (**DFS** = depth first search): Erreicht man einen Knoten y, so wird ab hier rekursiv weitergesucht, indem man allen von y ausgehenden Kanten folgt. Stößt man hierbei auf einen bereits besuchten Knoten, so wird in dieser Richtung nicht weitergesucht (Abbruch der Rekursion).

Breitensuche (**BFS** = breadth first search): Man durchläuft den Graphen, ausgehend von x, schalenförmig gemäß des Abstandes, d.h., man besucht zunächst alle Knoten y mit dem Abstand dist(x,y) = 1, dann alle Knoten y mit dem Abstand dist(x,y) = 2 usw.



Tiefensuche (**DFS** = depth first search) umgangssprachlich. Gerichteter Fall:

```
procedure besuche(u) is
begin  "bearbeite den Knoten u;"  markiere u als besucht;
       for alle v aus der Menge der Nachfolger S(u) loop
           if v noch nicht besucht then besuche(v); end if;
       end loop;
end besuche;
```

Im ungerichteten Fall ersetzt man die Menge S(u) durch die Menge der Nachbarn N(u).
Wir betrachten im Folgenden nur den gerichteten Fall.

Rekursives Programm zur Tiefensuche, gerichteter Fall.

```
procedure DFS (Anfang: in NextKnoten) is
k: NextKnoten;
procedure besuche(u: in NextKnoten) is
    edge: NextKante; v: NextKnoten;
    begin  u.besucht := true; edge := u.EIK;
        while  edge /= null  loop  v := edge.EKn;
            if not v.besucht then besuche(v); end if;
            edge := edge.NKa;  end loop;
    end besuche;
begin k := Anfang;
    while k /= null loop  k.besucht := false;  k:=k.NKn;  end loop;
    k := Anfang;
    while k /= null loop  if not k.besucht then besuche(k); end if;
        k := k.NKn;  end loop;
end DFS;
```



Iteratives Programm zur Tiefensuche, gerichteter Fall. Zu den Kellern vgl. Abschnitt 2.6 sowie Abschnitt 2.7, Folie 111.

```
procedure DFS (Anfang: in NextKnoten) is
k: NextKnoten; "KK: new Keller bestehend aus NextKnoten;"
begin k := Anfang;
    while k /= null loop  k.besucht := false; k:=k.NKn;  end loop;
    k := Anfang ;
    while k /= null loop
    if not k.besucht then newstack(KK); push(KK,k);
        while not isempty(KK) loop
            u := top(KK); pop(KK); u.besucht := true; edge := u.EIK;
            while  edge /= null  loop  v := edge.EKn;
                if not v.besucht then push(KK,v); end if;
                edge := edge.NKa;
            end loop;
        end loop;
    end if;
    k := k.NKn;
    end loop:
end DFS;
```

Iteratives Programm zur Breitensuche, gerichteter Fall: Ersetze den Keller durch eine Schlange! Zu Schlangen vgl. Abschnitt 2.6.

```
procedure BFS (Anfang: in NextKnoten) is
k: NextKnoten; "KK: new Schlange bestehend aus NextKnoten;"
begin k := Anfang;
    while k /= null loop  k.besucht := false; k:=k.NKn;  end loop;
    k := Anfang ;
    while k /= null loop
    if not k.besucht then newqueue(KK); enqueue(KK,k);
        while not isempty(KK) loop
            u := first(KK); dequeue(KK); u.besucht := true; edge := u.EIK;
            while  edge /= null  loop  v := edge.EKn;
                if not v.besucht then enqueue(KK,v); end if;
                edge := edge.NKa;
            end loop;
        end loop;
    end if;
    k := k.NKn;
    end loop:
end BFS;
```

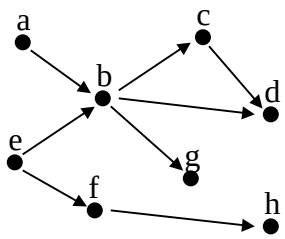


Tiefensuche mit der Adjazenzmatrix A als Darstellung:

```
procedure DFS_Adj is                    -- n ist global
A: array (0..n-1, 0..n-1) of integer;
besucht: array(0..n-1) of Boolean;
procedure besuche(j: in 0..n-1) is
    begin  besucht(j) := true;
        for k in 0..n-1 loop
            if A(j,k) /= 0 and not besucht(k)
            then besuche(k); end if;
        end loop;
    end besuche;
begin  ...  -- die Adjazenzmatrix A möge aufgebaut worden sein
    for i in 0..n-1 loop  besucht(i) := false;  end loop;
    for i in 0..n-1 loop
        if not besucht(i) then besuche(i); end if; end loop;
end DFS_Adj;
```

## 7.2 Topologisches Sortieren

Topologisches Sortieren ist das Einbetten einer Halbordnung in eine totale Ordnung. Die Halbordnung ist durch einen azyklischen Graphen (bzw. dessen transitiver Hülle) gegeben.




Angabe zweier Funktionen ord, vgl. Folie 13:

| a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|
| 1 | 4 | 5 | 7 | 2 | 3 | 6 | 8 |

| a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|
| 4 | 5 | 7 | 8 | 1 | 2 | 6 | 3 |

Es gibt noch weitere Anordnungen ord.



Satz:

Es sei G ein DAG, dann gibt es mindestens einen Knoten mit Eingangsgrad 0 und mindestens einen Knoten mit Ausgangsgrad 0.

Diese Aussage ist "klar": Folge von irgendeinem Knoten den auslaufenden Kanten. Da man nicht wieder auf einen bereits besuchten Knoten stoßen darf, muss jede Kantenfolge nach spätestens n-1 Schritten enden, womit man einen Knoten mit Ausgangsgrad 0 gefunden hat. Analog beim Rückwärts-Verfolgen von Kanten.