5 Grundlagen der Graphentheorie

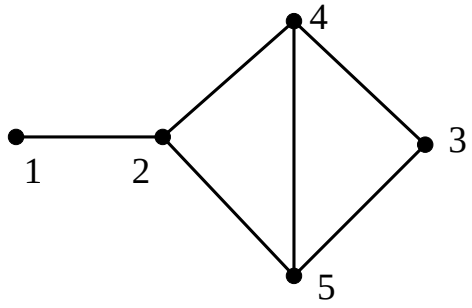
5.1 Graphen und ihre Darstellungen

Ein *Graph* beschreibt Beziehungen zwischen den Elementen einer Menge von Objekten. Die Objekte werden als Knoten des Graphen bezeichnet; besteht zwischen zwei Knoten eine Beziehung, so sagen wir, dass es zwischen ihnen eine Kante gibt.
Definition: Für eine Menge V bezeichne $\binom{V}{2}$ die Menge aller zweielementigen Untermengen von V. Ein *einfacher, ungerichteter Graph* $G = (V, E)$ (kurz *Graph* genannt) besteht aus einer endlichen Menge V von *Knoten* , auch *Ecken (Vertex)* genannt, und einer Menge $E \subseteq \binom{V}{2}$ von *Kanten (Edge)*. Hier sind Kanten ungeornete Paare von Knoten, d.h. $\{u, v\}$ und $\{v, u\}$ sind zwei verschiedene Schreibweisen für ein und dieselbe Kante. Im Gegensatz dazu ist *gerichteter Graph* G ein Paar (V, E) bestehend aus einer endlichen Knotenmenge V und einer Kantenmenge E von geordneten Knotenpaaren $e = (u, v)$, mit $u, v \in V$.

Ist $e = \{u, v\}$ eine Kante von G, dann nennt man die Knotenen u und v zueinander *adjazent* oder *benachbart* und man nennt sie *inzident* zu e. Die Menge $N(v) = \{u \in V \mid \{u, v\} \in E\}$ der zu einem Knoten v benachbarten Knoten wird die Nachbarschaft von v genannt. Der Grad eines Knotens v wird durch $deg(v) = |N(v)|$ definiert.
Die Anzahl der Knoten $|V|$ bestimmt die *Ordnung* und die Anzahl der Kanten $|E|$ die *Größe* eines Graphen.

Im Folgenden werden verschiedene Darstellungen von Graphen an einem Beispiel demonstriert.

1) Darstellung als Zeichnung:



2) Darstellung als Adjazenzmatrix. Jedem Knoten wird eine Zeile und eine Spalte zugeordnet und der Eintrag in der Zeile von u und der Spalte von v wird 1 gesetzt, wenn $\{u, v\}$ eine Kante des Graphen ist (sonst 0):

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

3) Darstellung als Adjazenzliste. Für jeden Knoten wird die Liste seiner Nachbarn angegeben (Liste von Listen):
$1:2;\ 2:1,4,5;\ 3:4,5;\ 4:2,3,5;\ 5:2,3,4$; oder mit anderer Syntax
$(2),(1,4,5),(4,5),(2,3,5),(2,3,4)$
4) Darstellung als Inzidenzmatrix. Jedem Knoten wird eine Zeile und jeder Kante eine Spalte zugeordnet und der Eintrag in der Zeile von u und der Spalte von e wird 1 gesetzt, wenn u eine Ecke der Kante e ist (sonst 0):

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 9 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Für die Behandlung algorithmischer Probleme sind oft Adjazenzlisten die Datenstruktur der Wahl, weil sie die wahre Größe eines Graphen (Anzahl der Kanten) widerspiegeln, während Adjazenzmatrizen per Definition $|V|^2$ Einträge haben. Andererseits haben Adjazenzmatrizen den Vorteil, dass Anfragen, ob zwei Knoten u und v durch eine Kante verbunden sind in konstanter Zeit (ein Speicherzugriff) beantwortet werden können, während man Adjazenzlisten durchsuchen muss.
Während die Adjazenzmatrix eines ungerichteten Graphen symmetrisch ist (Diagonale als Symmetrieachse) sind Adjazenzmatrizen von gerichteten Graphen im allgemeinen nicht symmetrisch.

Da Graphen über eine sehr einfache Struktur verfügen, finden sie bei der Modellierung und algorithmischen Lösung vieler praktischer Probleme Anwendung, wie z.B.

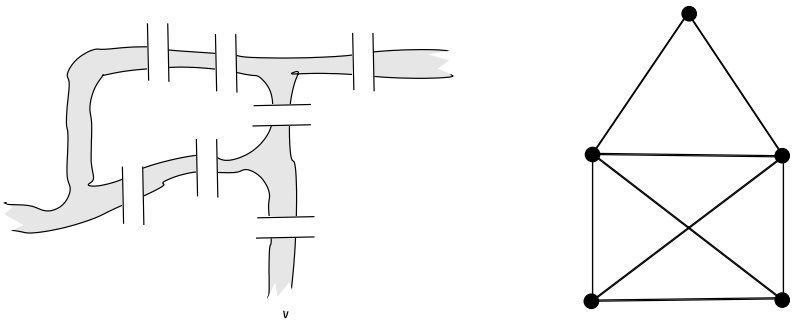
- Modellierung von Straßen-, Flug- und Telefonnetzen
- Darstellung von Molekülen
- Interpretation von Relationen (Beziehungsgeflechten)
- Gerüste von Polyedern (lineare Optimierung)
- Entwurf von Mikrochips (VLSI-Design)

Die folgenden Beispiele für graphentheoretische Aufgabenstellungen haben die die Entwicklung der Graphentheorie stark beeinflusst und unterstreichen die praktische Relevanz dieser Struktur:

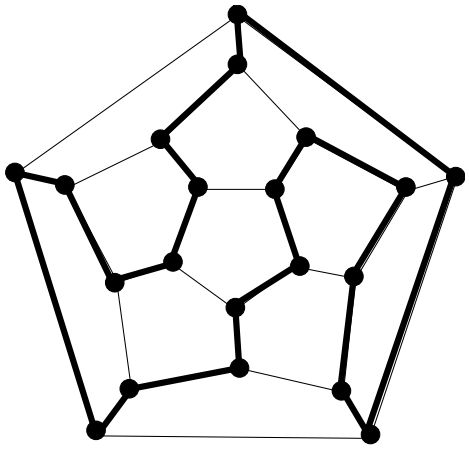
1. Das 4-Farben-Problem: Man stelle sich die Welt mit einer beliebigen politischen Landkarte vor. Wir definieren einen Graphen, indem wir jedem Land einen Knoten zuordnen und zwei Knoten mit einer Kante verbinden, wenn sie einen gemeinsamen Grenzabschnitt haben. Wie viele Farben braucht man, um die Länder so einzufärben, dass benachbarte Länder verschiedene Farben haben? Man hat (mit Computerhilfe) bewiesen, dass vier Farben immer ausreichen! Einen Beweis ohne Computer gibt es bis heute nicht.

2. Welche Graphen treten als Ecken-Kanten-Gerüst von Polyedern auf? Solche Graphen spielen insbesondere bei der Lösung von linearen Optimierungsproblemen eine große Rolle.

3. Eulersche Kreise: Man charakterisiere jene Graphen, bei denen man die Kanten so durchlaufen kann, dass man jede Kante einmal benutzt und man am Schluss wieder am Ausgangspunkt steht. Der Ausgangspunkt für diese Frage war das von Euler gelöste sogenannte Königsberger Brückenproblem, bei dem es um die Frage geht, ob man einen Rundgang durch die Stadt machen kann, bei dem jede Brücke genau einmal überquert wird. Die Antwort ist negativ, weil auf jeder Flußseite und auf Insel eine gerade Anzahl von Brücken ankommen müsste, um eine solchen Rundgang realisieren zu können. Wenn man auf die Bedingung der Gleichheit von Anfangs- und Endpunkt verzichtet, können auch zwei Knoten ungeraden Grads auftreten. Man spricht dann von Eulerschen Wegen. Auf der rechten Seite ist ein solcher Graph abgebildet, dessen Eulerscher Weg aus dem "Haus vom Nikolaus"-Prinzip bekannt sein sollte.



4. Hamiltonsche Graphen: Dies sind Graphen, die man so durchlaufen kann, dass man jeden Knoten genau einmal besucht bis man zum Ausgangsknoten zurückkehrt. Während man für das vorherige Problem effiziente algorithmische Lösungen kennt, ist dieses algorithmisch schwer (NP-vollständig).



5. Travelling Salesman Problem (TSP): Oft hat man es mit bewerteten Graphen zu tun, das heißt Kanten und/oder Knoten haben zusätzliche Informationen wie Gewichte, Längen, Farben etc.
   Ein Beispiel ist das TSP. Wir haben n Städte. Für jedes Paar $\{u, v\}$ von Städten kennt man die Kosten, um von u nach v zu kommen. Man entwerfe für den Handelsreisenden eine geschlossene Tour, die alle Städte besucht und minimale Gesamtkosten hat. Auch dies ist ein algorithmisch schweres Problem.

6. Planare Graphen: Welche Graphen lassen sich so in der Ebene zeichnen, dass sich Kanten nicht schneiden, also sich höchstens in Knoten berühren? Wie kann man sie charakterisieren und algorithmisch schnell erkennen?

7. Netzwerke: Welchen Graphen sollte man der Architektur eines Rechnernetzes zu Grunde legen, wenn die Kanten beschränkte Kapazität haben, aber trotzdem schneller Informationsaustausch gewährleistet werden soll?

Satz (Handschlaglemma): Für jeden Graph $G = (V, E)$ gilt $\sum_{v \in V} deg(v) = 2|E|$, d.h. $\sum_{v \in V} deg(v)$ ist eine gerade Zahl.
Beweis: Bei Betrachtung der Inzidenzstruktur zwische Knoten und Kanten ergibt sich die Aussage durch doppeltes Abzählen: Für jeden Knoten v ist die Anzahl inzidenter Kanten $deg(v)$ und jede Kante ist zu ihren zwei Eckknoten inzident. Damit erhalten wir $\sum_{v \in V} deg(v) = \sum_{e \in E} 2 = 2|E|$.

Folgerung: Die Anzahl der Knoten mit ungeraden Grad ist in jedem Graphen eine gerade Zahl.

Definition: Seien $G = (V, E)$ und $G' = (V', E')$ zwei Graphen. Eine Abbildung $\varphi : V \longrightarrow V'$ wird *Graphhomomorphismus* genannt, falls für alle Kanten $\{u, v\} \in E$ auch $\{\varphi(u), \varphi(v)\} \in E'$ gilt. Ist darüber hinaus φ eine bijektive Abbildung und φ^{-1} auch ein Graphhomomorphismus, so nennt man φ einen *Graphisomorphismus* (und G, G' zueinander *isomorph*).

Die folgenden Standardbeispiele beschrieben formal gesehen nicht einzelne Graphen, sondern Isomorphieklassen.

1. Mit K_n ($n \geq 1$) bezeichnet man den *vollständigen Graphen* der Ordnung n, d.h. eine Knotenmenge V mit $|V| = n$ und der vollen Kantenmenge $\binom{V}{2}$.

2. Mit C_n ($n \geq 3$) bezeichnet man den *Kreis* der Länge n, d.h. eine Knotenmenge $V = \{v_1, v_2, \ldots, v_n\}$ mit der Kantenmenge $E = \{\{v_1, v_2\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$.

3. Mit Q_n ($n \geq 1$) bezeichnet man den *n-dimensionalen Würfel* mit Knotenmenge $\{0, 1\}^n$ (Menge aller n-Tupel über $\{0, 1\}$) wobei zwei Tupel dann und nur dann adjazent sind, wenn sie sich an genau einer Stelle unterscheiden.
   Die Bestimmung der Anzahl der Kanten des Q_n ist eine schöne Anwendung des

Handschlaglemmas. Wie man leicht sieht, hat jeder Knoten in Q_n den Grad n und folglich ist die Gradsumme aller Knoten gleich $n \cdot 2^n$. Damit ist $2|E| = n \cdot 2^n$, also $|E| = n \cdot 2^{n-1}$.

4. Mit $K_{n,m}$ $(n, m \geq 0)$ bezeichnet man den vollständigen, bipartiten Graphen, dessen Eckenmenge V die disjunkte Vereinigung von zwei Mengen A und B mit $|A| = n,\ |B| = m$ ist und dessen Kantenmenge aus allen Paaren $\{a, b\}$ mit $a \in A,\ b \in B$ besteht.
   Die Graphen $K_{1,m}$ und $K_{2,m}$ sind planar. Dagegen ist der $K_{3,3}$ und alle $K_{n,m}$ mit $n, m \geq 3$ nicht planar.

Definition: Man nennt $G' = (V', E')$ einen *Untergraph* von $G = (V, E)$, wenn $V' \subseteq V$ und $E' \subseteq E$ gilt. Ist außerdem $E' = E \cap \binom{V}{2}$, so wird G' als *induzierter Untergraph* von G bezeichnet.

Definition: Ein Graph $G = (V, E)$ wird *bipartit* genannt, wenn er Untergraph eines vollständigen, bipartiten Graphen ist. Etwas anschaulicher kann man formulieren, daß ein Graph genau dann bipartit ist, wenn man die Knoten so mit zwei Farben einfärben kann, daß keine gleichfarbigen Ecken benachbart sind.

Definition: Das Komplement eines Graphen $G = (V, E)$ ist der Graph
$\overline{G} = (V, \binom{V}{2} \setminus E)$.

Definition: Eine Folge von paarweise verschiedenen Ecken $v_1, v_2, \ldots, v_k$ eines Graphen $G = (V, E)$ repräsentiert einen *Weg der Länge* $k - 1$, falls $\{v_i, v_{i+1}\} \in E$ für alle $1 \leq i < k$. Ist außerdem $\{v_k, v_1\} \in E$, so repräsentiert die Folge auch einen *Kreis der Länge* k.
Man sagt, daß v von u erreichbar ist, falls ein Weg $v_1, v_2, \ldots, v_k$ mit $v_1 = u$ und $v_k = v$ in G existiert. Die Länge eines kürzesten Weges zwischen zwei Knoten nennt man ihren Abstand in G.

Lemma: Die Relation der Erreichbarkeit in der Knotenmenge V eines Graphen ist eine Äquivalenzrelation.

Definition: Die Äquivalenzklassen der Erreichbarkeitsrelation nennt man die *Zusammenhangskomponenten* (kurz *Komponenten*) des Graphen. G wird zusammenhängend genannt, wenn er genau eine Komponente hat.

Stellt man G durch eine Zeichnung dar, so kann man diesen Begriff anschaulich erklären: Zwei Knoten gehören zur selben Komponente, wenn man sie durch einen geschlossenen Kantenzug verbinden kann, wobei die Kanten nur in Knoten und nicht auf Kantenschnitten in der Zeichnung gewechselt werden dürfen.

Satz: Sind zwei Graphen isomorph so sind jeweils die Ordnung, die Größe, die sortierten Gradfolgen und die Anzahl der Komponenten der beiden Graphen gleich.

Definition: Seien u, v Knoten in einem ungerichteten Graphen $G = (V, E)$. Sind u und v in einer gemeinsamen Zusammenhangskomponente von G, so definieren wir ihren *Abstand* $d(u, v)$ als Länge eines kürzesten Weges (Anzahl der Kanten des Wegs) von u nach v. Gehören sie zu verschiedenen Komponenten, so setzen wir $d(u, v) = \infty$.

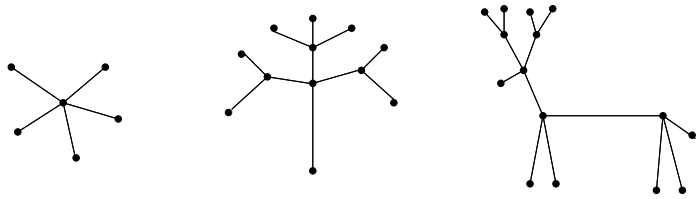
Definition: Der *Durchmesser* $D(G)$ des Graphen ist definiert als das Maximum über alle paarweisen Abstände zwischen Knoten.

Satz: Ein Graph ist genau dann bipartit, wenn alle in ihm als Untergraph enthaltenen Kreise gerade Länge haben.

Beweis: Zunächst überlegt man sich, dass wir den Graphen als zusammenhängend voraussetzen können, ansonsten führt man den folgenden Beweis für jede Zusammenhangskomponente.
Sei $G = (V, E)$ bipartit, das heißt, $V = A \cup B$ mit $A \cap B = \emptyset$ und Kanten verlaufen nur zwischen Knoten aus A und Knoten aus B. Sei des weiteren C ein Kreis in G. C benutzt abwechselnd Knoten aus A und B und hat somit gerade Länge.
Wir zeigen die andere Richtung. Wir fixieren einen beliebigen Knoten $u \in V$. Wir definieren: $A = \{v \in V |\ d(u, v)\ \text{ gerade }\}$, $B = V \setminus A$ Zu zeigen, es gibt keine Kanten zwischen Knoten aus A (bzw. aus B). Wir führen einen indirekten Beweis:
Wir nehmen an, es gibt eine Kante $\{v, w\}, v, w \in B$ (für A analog) und finden einen Widerspruch zur Annahme, dass alle Kreise gerade Länge haben. Wir betrachten kürzeste Wege von u zu v und zu w. Diese Wege haben gleiche Länge! (wegen der Kante zwischen v und w) Sei x der letzte gemeinsame Knoten auf beiden Wegen. Dann bilden die beiden Wegabschnitte von x nach v bzw. nach w zusammen mit der Kante $\{v, w\}$ einen Kreis ungerader Länge.

5.2 Bäume

Definition: Ein Graph G heißt ein Baum, wenn er zusammenhängend ist und keine Kreise enthält. Ein Graph, dessen Komponenten jeweils Bäume sind, wird ein Wald genannt. Ein Graph ist also genau dann ein Wald, wen er keine Kreise enthält.



Die Abbildung zeigt drei Bäume, die zusammen einen Wald bilden.

Satz: Sei $G = (V, E)$ ein Graph, dann sind die folgenden drei Bedingungen äquivalent:

1. G ist ein Baum.
2. Je zwei Ecken von G sind durch genau einen Weg verbunden.
3. G ist zusammenhängend und $|E| = |V| - 1$.

Beweis: (1) $\Rightarrow$ (2) und (2) $\Rightarrow$ (1) sind einfache indirekte Schlüsse. Die erste Implikation sieht man z.B. wie folgt. Angenommen es gibt zwei Knoten u, v, zwischen denen nicht genau ein Weg verluft.Dies knnte kein Weg sein, dann ist der Graph nicht

zusammenhngend, oder mindestens zwei Wege, dann entsteht ein Kreis. In jedem Fall ist aber G kein Baum.

Wir zeigen (1) $\Rightarrow$ (3):
Zunächst hat jeder Baum, der nicht nur ein einzelner Knoten ist, Knoten vom Grad 1, diese nennt man *Blätter*. Das sieht man wie folgt. Seien $u_1, u_2, \ldots, u_i$ die Knoten eines längsten Weges in G. Alle Nachbarn von u_1 liegen auf diesem Weg, sonst wäre er nicht längster Weg. Nur u_2 kann Nachbar sein, sonst gäbe es einen Kreis.
Wir entfernen u_1 und die Kante $\{u_1, u_2\}$ aus G und erhalten einen zusammenhängenden Restgraphen G'. Dieser hat genau einen Knoten und eine Kante weniger als G. Wenn wir dies iterieren, bleibt zum Schluss genau ein Knoten ohne Kanten übrig. Also $|E| = |V| - 1$.

(3) $\Rightarrow$ (1): Sei $T = (V, E')$ aufspannender Baum von G, damit $|V| - |E'| = 1$. Aber nach Vorraussetzung gilt auch $|V| - |E| = 1$. Allerdings ist $E' \subseteq E$ und die einzige Möglichkeit hierfür ist $E = E'$.

Definition: Sei $G = (V, E)$ ein zusammenhängender Graph. Ein *aufspannender Baum* von G ist ein Untergraph mit gleicher Knotenmenge, der ein Baum ist. Analog ist ein *aufspannerder Wald* eines beliebigen (auch unzusammenhängenden) Graphen G ein Untergraph mit den gleichen Komponenten wie G, der ein Wald ist.

Für den Informatiker sind Bäume ein sehr wichtiges Werkzeug. Dabei stellt die Tatsache, daß sich eine Reihe praktischer Problemstellungen auf die Konstruktion von aufspannenden Bäumen (bzw. Wäldern) zurückführen lassen, nur einen Teilaspekt des gesamten Anwendungsspektrums dar. In der theoretischen Informatik dienen sogenannte Entscheidungsbäume als Modell für Berechnungen. Schließlich spielen Bäume als Datenstruktur für Such- und Sortierprozesse eine wichtige Rolle.

Oft ist es nützlich, einen Baum als *gerichteten Graphen* anzusehen, d.h. es werden gerichtete Kanten (Formalisierung durch geordnete Paare) betrachtet. Ist $G = (V, E)$ ein Baum und $r \in V$ eine beliebiger, aber fixierter Knoten, den wir die *Wurzel (Root)* nennen werden, so kann man die Kanten von G in eindeutiger Weise so richten, daß für alle Ecken $v \neq r$ die Kanten auf dem (eindeutigen!) Weg von v nach s zur Wurzel hin gerichtet werden.
Es gibt zwei Verfahren, die Breitensuche (BFS - Breadth-first search) und die Tiefensuche (DFS -Depth-first search), die für einen gegebenen zusammenhängenden Graphen $G = (V, E)$ und einem gegebenen Knoten $r \in V$ einen aufspannenden Baum von G mit Wurzel (Root) r ausgeben. Beide Verfahren durchsuchen G beginnend von r und unterscheiden sich nur durch die für die Zwischenspeicherung verwendete Datenstruktur, durch die festgelegt wird, von welcher bereits erreichten Ecke u die Suche fortgesetzt wird: Eine Warteschlange Q (Prinzip: first in first out) bei BFS und ein Kellerspeicher (Stapelprinzip: last in first out) bei DFS. Der zu bestimmende Baum wird durch ein Feld (Array) π repräsentiert, das für jeden Knoten $v \in V$ den ersten Nachfolger von v auf dem gerichteten Weg zur Wurzel r speichert. Für die Wurzel selbst wird der Wert NIL eingetragen.

Zuerst werden alle Knoten weiß (unberührt) markiert. Dann wird r in den Zwischenspeicher (Warteschlage/Kellerspeicher) geladen und grau (berührt) markiert. In jedem weiteren Schritt wird der nächste Knoten u aus dem Zwischenspeicher (also grau) ausgelesen und getestet, ob er noch einen weißen Nachbarn besitzt. Bei negativer Antwort entfernt man u durch schwarze Markierung (erledigt) aus dem Speicher. Hat u einen weißen Nachbarn v, so wird dieser auch gespeichert (grau gefärbt) und $\pi(v) = u$ gesetzt. Dieser Schritt ist damit gleichzusetzen, dass die Kante $\{v, u\}$ in die ungerichtete Version des Baums aufgenommen wird.
Beide Verfahren stoppen, wenn keine grauen Ecken mehr vorhanden sind.
Hier sind die Pseudocodes der beiden Algorithmen:

```
BFS(G, r)
for all u ∈ V
    do Farbe[u] ← weiß                    Initialisierung
Farbe[r] ← grau
π[r] ← nil                                r als Wurzel festlegen
Insert r in Q                             Wurzel in Q eintragen
while Q≠ ∅                                sonst abbrechen
    do ū ← Kopf[Q]                        u an erster Stelle in Q
        for all v ∈ Adj[u]
            do if Farbe[v] == weiß        v noch nicht berührt
                then Farbe[v] ← grau      v wird berührt
                     π[v] ← u             Kante von v zu u
                     Insert v in Q        v in Q eintragen
        Delete Head(Q)                    u aus Q streichen
        Farbe[u] ← schwarz
```

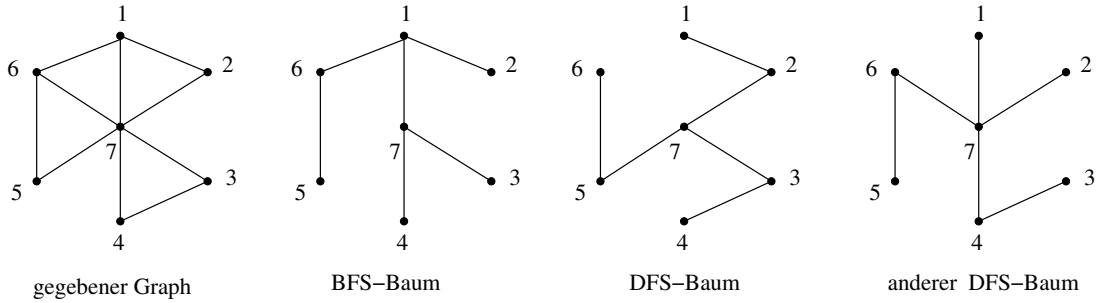
```
DFS(G, r)
for all u ∈ V Farbe[u] ← weiß     Initialisierung
π[r] ← nil                        r als Wurzel festlegen
DFS-visit(r)                      Aufruf einer rekursiven Prozedur
```

```
Procedure DFS-visit(u)
Farbe[u] ← grau
for all v ∈ Adj[u]
    do if Farbe[v] == weiß        v noch nicht berührt
        then π[v] ← u             Kante von v zu u
             DFS-visit(v)         gehe von v weiter in die Tiefe
Farbe[u] ← schwarz
```

Beide Algorithmen können auch zur Erzeugung eines aufspanneden Waldes von unzusammenhängenden Graphen verwendet werden. Dazu ist nur die folgende Änderung notwendig: Hat man einen aufspannenden Baum für eine Komponente erzeugt (keine

grauen Punkte mehr), so wählt man, falls noch vorhanden, einen weißen Knoten als Wurzel für eine weitere Komponente aus und startet neu.

Die Ergebnisse der beiden Methoden sind in der Regel sehr unterschiedlich: Bei der Tiefensuche DFS werden in der Regel sehr lange Wege erzeugt. Es ist zu beachten, dass die berechneten DFS–Bäume nicht nur von der Wahl der Wurzel r abhängen, sondern auch die Reihenfolge der zu u benachbarten Knoten in der Adjazenzliste $Adj[u]$ einen Einfluss auf das Ergebnis hat. So gibt es für Graphen mit einem Hamiltonkreis bei geeigneter Ordnung der Adjazenzlisten immer einen DFS-Baum der Tiefe $|V|-1$ (also ein Weg). Eine Änderung dieser Ordnung kann (eventuell) zu DFS-Bäumen geringerer Tiefe führen. Das folgende Beispiel zeigt den BFS– und den DFS–Baum eines Graphen, bei Start im Knoten 1 und aufsteigend geordneten Adjazenzlisten. Ganz rechts sieht man einen anderen DFS–Baum, der bei gleichem Startknoten und absteigend geordneten Adjazenzlisten entsteht.



Obwohl auch das Ergebnis der Breitensuche von der Ordnung der Adjazenzlisten abhängt, haben für eine festgelegte Wurzel r alle möglichen BFS–Bäume eine gemeinsame Eigenschaft, die im folgenden Lemma formuliert wird.

Lemma: In einem BFS–Baum T ist der Abstand eines beliebigen Knotens $v \in V$ zur Wurzel r genau so groß wie im ursprünglichen Graphen G, kurz geschrieben:

$$d_T(v,r) = d_G(v,r).$$

Man kann dieses Lemma mit vollständiger Induktion über $d_G(v,r)$ beweisen.

In vielen Anwendungen sind die Kanten eines Graphen $G = (V,E)$ mit positiven Zahlen gewichtet, d.h. zusätzlich ist eine Kostenfunktion $w : E \longrightarrow \mathbb{R}^+$ gegeben. Nun besteht die Aufgabe darin, einen aufspannenden Baum (bzw. Wald) zu bestimmen, so daß das Gesamtgewicht der Kanten des Baums (Waldes) minimal ist (Minimum Spanning Tree - MST).

Der nachfolgend beschriebene Algorithmus von Kruskal löst dieses Problem. Man ordnet die Kanten nach aufsteigendem Gewicht in einer Liste L, die dynamisch verwaltet wird. Der Aufbau des Baumes T erfolgt beginnend vom Graphen $(V,\emptyset)$, indem Schritt für Schritt neue Kanten aufgenommen werden:
Dazu wird die erste Kante e aus der aktuellen Liste L (also eine mit minimalem Gewicht) gestrichen und getestet, ob durch Hinzunahme von e zu den bereits in

T aufgenommenen Kanten ein Kreis entsteht. Bei negativer Antwort wird auch e aufgenommen, sonst nicht.
Intuitiv sollte klar sein, dass am Ende (wenn L leer ist) T ein aufspannender Wald mit minimalem Gewichts ist. Natürlich kann man diese Tatsache auch beweisen.
Zur effektiven Implementierung dieser Methode werden sogenannte *union-find-Datenstrukturen* eingesetzt.

5.3 Matchings

In vielen Anwendungsaufgaben, die sich mit Graphen modellieren lassen, geht es darum, möglichst große Paarungen von adjazenten Knoten zu bilden.
Beispiel: Wir betrachten eine Menge $P = \{p_1, p_2, \ldots, p_n\}$ von Personen und eine Menge $S = \{s_1, s_2, \ldots, s_m\}$ von Jobs und beschreiben durch einen bipartiten Graphen, welche Personen für welche Jobs geeignet sind. Ziel ist es, eine maximale Anzahl von Personen mit Jobs zu versorgen. Wir suchen also eine maximale Anzahl von Kanten M, so dass jedes p_i und jedes s_j zu höchstens einer Kante aus M inzident ist, denn jeder Job kann nur einmal vergeben werden und jede Person kann nur einen Job übernehmen. Diese Aufgabenstellung ist ein typisches Beispiel für ein Matching–Problem.

Definition: Sei $G = (V, E)$ ein Graph. Eine Untermenge $M \subseteq E$ wird *Matching* von G genannt, wenn jeder Knoten $v \in V$ zu höchstens einer Kante aus M inzident ist. Mit $m(G)$ bezeichnen wir die maximale Größe eines Matchings von G und nennen M ein *Maximum–Matching*, wenn $|M| = m(G)$.

Definition: Ist M ein Matching von $G = (V, E)$, dann nennt man einen Knoten *M–saturiert* (bzw.*M–unsaturiert*), wenn er zu einer (bzw. zu keiner) Kante $e \in M$ inzident ist. Ein Weg p in G wird *M–augmentierend* genannt, wenn er mit M–unsaturierten Knoten beginnt und endet und wenn sich auf den Weg Kanten aus M und aus $E \setminus M$ sich gegenseitig ablösen (alternieren).

Lemma: Ist M ein Matching von G und ist p ein M–augmentierender Weg, dann ist M **kein** Maximum–Matching.

Beweis: Tauscht man alle Kanten aus M, die auf p liegen gegen die Nichtmatching–Kanten auf p aus, so erhält man ein neues Matching M' mit $|M'| = |M| + 1$. Damit erklärt sich auch der Begriff augmentierend (= erweiternd).

Wir betrachten zunächst Matchings in bipartiten Graphen. Für eine Teilmenge $X \subseteq V$ bezeichne $N(X)$ die Menge aller zu X benachbarten Knoten, d.h. $N(X) = \bigcup_{v \in X} N(v)$.
Heiratssatz: Sei $G = (A \cup B, E)$ ein bipartiter Graph. Dann ist $m(G) = |A|$ genau dann, wenn für jede Teilmenge $X \subseteq A$ die Ungleichung $|X| \leq |N(X)|$ gilt.

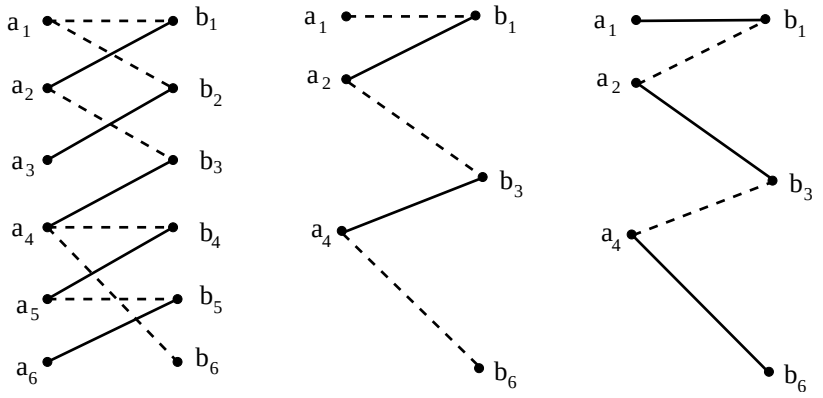
Beweis: Ist $m(G) = |A|$, dann können alle Knoten aus A gematcht werden und jeder Teilmenge $X \subseteq A$ steht mindestens die Menge der entsprechenden Matchingpartner aus B gegenüber.

Nun gehen wir von $|X| \leq |N(X)|$ für alle $X \subseteq A$ aus, betrachten ein Maximum–Matching M und müssen $|M| = |A|$ zeigen. Das beweisen wir mit Widerspruch, indem aus der Annnahme $|M| < |A|$ die Existenz eines M–augmentierenden Wegs p abgeleitet wird (Widerspruch zur Maximalität von M):

Angenommen $|M|$ ist kleiner als $|A|$, dann gibt es ein M–unsaturiertes $a_1 \in A$, das aber wegen $1 = |\{a_1\}| \leq |N(\{a_1\})|$ mindestens einen Nachbarn $b_1 \in B$ hat. Wäre b_1 M–unsaturiert, dann bildet bereits die Kante $\{a_1, b_1\}$ einen M–augmentierenden Weg.

Anderenfalls ist b_1 M–saturiert und folglich gibt es eine Kante $\{a_2, b_1\} \in M$. Wegen $2 = |\{a_1, a_2\}| \leq |N(\{a_1, a_2\})|$ gibt es ein von b_1 verschiedenes $b_2 \in N(\{a_1, a_2\})$. Ist b_2 M–saturiert, dann gibt es eine Kante $\{a_3, b_2\} \in M$ und man kann wegen $3 = |\{a_1, a_2, a_3\}| \leq |N(\{a_1, a_2, a_3\})|$ ein weiteres $b_3 \in N(\{a_1, a_2, a_3\})$ finden, u.s.w.
Da G endlich ist, findet man irgendwann ein M–unsaturiertes $b_k \in N(\{a_1, a_2, \ldots a_k\})$ und beginnt den Aufbau des M–augmentierenden Wegs p von dort. Sei $i_1 \leq k$ der kleinste Index, so dass $\{a_{i_1}, b_k\} \in E$ ist (diese Kante ist nicht aus M!). Im Fall $i_1 = 1$, ist der Weg $p = b_k, a_1$ bereits M–augmentierend. andernfalls setzen wir mit der Kante $\{a_{i_1}, b_{i_1-1}\}$ aus M fort und finden den kleinsten Index i_2, so dass $\{a_{i_2}, b_{i_1-1}\} \in E$ ist. Diese Kante ist nicht aus M und $i_2 \leq i_1 - 1$. Wiederum ist man im Fall $i_2 = 1$ bereits fertig. Andernfalls wird dieser Schritt wiederholt, bis man bei $i_l = 1$ stoppen kann. Wegen $k \geq i_1 > i_2 > \ldots$ muss dieser Prozess terminieren und damit ist die Existenz eines M–augmentierenden Wegs bewiesen.

Die folgende Abbildung illustriert diesen Beweis. Auf der linken Seite findet man die Grundkonstruktion, wobei die Knoten in der Reihenfolge $a_1, b_1, a_2, b_2, a_3, \ldots, b_6$ bestimmt werden. Die durchgezeichneten Kanten sind aus M, die gestrichelten aus $E \setminus M$. In der Mitte ist der M–augmentierende Weg dargestellt. Durch Austausch von Matchingkanten gegen Nichtmatchingkanten (rechts) wird M vergrößert.



Die Verwendung von M–augmentierenden Wegen zur schrittweisen Vergrößerung von Matchings kann man zur Berechnung von Maximum–Matchings in Graphen verwenden. Man beginnt mit einem beliebigen Matching M ($M = \emptyset$ geht immer) und sucht einem M–alternierenden Weg. Solange ein solcher Weg existiert, kann man Austausch von Matching–Kanten gegen Nichtmatching–Kanten das Matching vergrößern

(augmentieren) und rekursiv fortfahren. Der folgende Satz garantiert, dass man im anderen Fall fertig ist.

Satz: Sei M ein beliebiges Matching und M' ein Maximum–Matching von einem Graphen $G = (V, E)$. Ist $|M| < |M'|$, dann gibt es einem M– augmentierenden Weg in G.

Beweis: Wir betrachten den Graphen $H = (V, F)$, wobei F die symmetrische Differenz von M und M' ist (das sind alle Kanten, die entweder in M oder in M' liegen). Da die Knoten in H nur vom Grad $0, 1$ oder 2 sein können, sind die Zusammenhangskomponenten von H isolierte Punkte, gerade Kreise mit M/M'–alternierenden Kanten oder Wege mit M/M'–alternierenden Kanten. Wegen $|M| < |M'|$ muss mindestens einer dieser Wege mehr Kanten aus M' als aus M enthalten und ist folglich M–augmentierend.

Leider sind effiziente Suchverfahren nach M–augmentierenden Wegen im allgemeinen sehr kompliziert, wobei ungerade Kreise das Haupthindernis darstellen. In bipartiten Graphen $G = (A \cup B, E)$ kann man das Problem auf die folgende Art und Weise lösen: Wir führen zwei zusätzliche Knoten s, t ein und bauen einen gerichtetenen Graphen G_M auf, indem alle Kanten aus M von A nach B und alle Kanten aus $E \setminus M$ von B nach A gerichtet werden. Zusätzlich wird für alle M–unsaturierten Knoten $v \in A$ (bzw. $u \in B$) eine Kante von v nach t (bzw. von s nach u) eingefügt. Offensichtlich existiert ein M–augmentierender Weg genau dann, wenn ein gerichteter Weg von s nach t in G_M existiert. Ein solcher Weg kann duch Tiefensuche (gerichtete Variante) gefunden werden.

Die folgende Abbildung illustriert den vorgestellten Algorithmus. Auf der linken Seite ist die Konstruktion des gerichteten Graphen dargestellt. Der gerichtete Weg von s nach t auf der rechten Seite liefert unmittelbar einen M–augmentierenden Weg, wenn man die erste (von s ausgehende) Kante und die letzte (zu t führende) Kante streicht.