

5.4 Modellierung mit Bäumen

In einem **ungerichteten Baum** gibt es **zwischen zwei beliebigen Knoten genau einen Weg**.

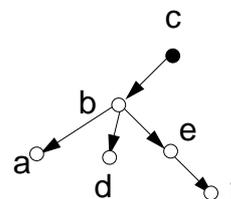
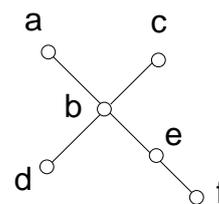
Ein gerichteter, **azyklischer** Graph G ist ein **gerichteter Baum**, wenn alle Knoten einen **Eingangsgrad ≤ 1** haben und es genau einen Knoten mit **Eingangsgrad 0** gibt, **er ist die Wurzel** von G . G ist ein **gewurzelter Baum**.

Man kann aus einem **ungerichteten Baum** in eindeutiger Weise einen gerichteten machen, indem man **einen Knoten zur Wurzel bestimmt**.

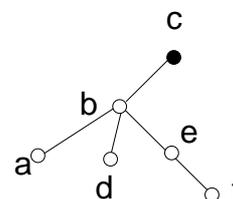
Deshalb wird in gewurzelter Bäumen häufig die **Kantenrichtung nicht angegeben**.

In einem gewurzelter Baum ist die **Höhe eines Knotens v** die größte Länge eines Weges von v zu einem Blatt. Die Höhe der Wurzel heißt **Höhe des Baumes**.

Knoten, die weder Wurzel noch Blatt sind heißen **innere Knoten**.



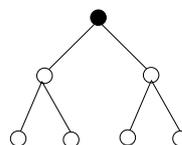
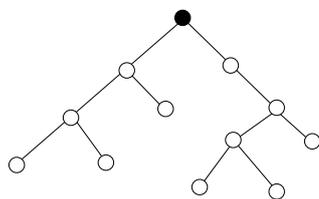
Wurzel



Binärbäume

Ein gewurzelter Baum heißt **Binärbaum**, wenn seine Knoten einen **Ausgangsgrad von höchstens 2** haben.

Ein **Binärbaum** heißt **vollständig**, wenn jeder Knoten außer den Blättern den **Ausgangsgrad 2** hat und die **Wege zu allen Blättern gleich lang** sind.



Höhe 2

Knoten: 7

Blätter: 4

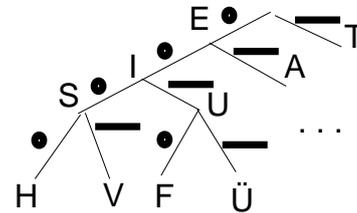
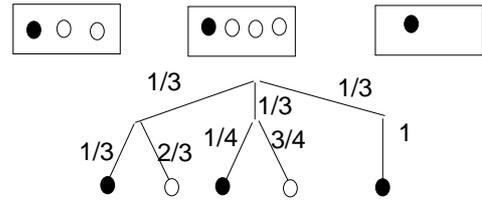
Ein vollständiger **Binärbaum** der Höhe h hat 2^h **Blätter** und $2^{h+1}-1$ **Knoten**

Modellierung von Entscheidungsbäumen

Knoten modelliert **Zwischenstand** einer mehrstufigen Entscheidungsfolge

Kante modelliert eine der wählbaren **Alternativen**

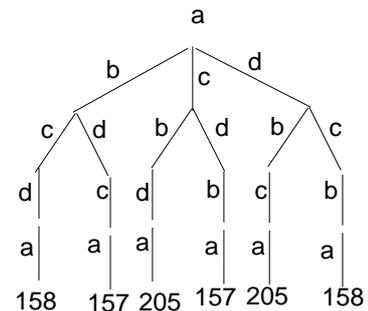
1. **Wahrscheinlichkeiten**,
z. B. erst Schachtel, dann Kugel ziehen:
2. **Codierungen**,
Z. B. Morse-Code



3. **Lösungsbaum** für kombinatorische Probleme,
z. B. Traveling Salesman's Problem (Mod-5.13)
Blätter repräsentieren einen Rundwege von a aus,
Kanten sind mit Entscheidungen markiert

4. **Spielzüge**, z. B. Schach (ohne Bild)

Wird **derselbe Zwischenstand** durch verschiedene Entscheidungsfolgen erreicht, kann man **Knoten identifizieren**.
Es entsteht ein azyklischer oder zyklischer Graph.



Modellierung von Strukturen durch Bäume

Knoten modelliert ein **Objekt**.

Kante modelliert **Beziehung** „besteht aus“, „enthält“, „spezialisiert zu“, ...

Beispiele:

- **Typhierarchie:** Typ - Untertypen
- **Klassenhierarchie:** Oberklasse als Abstraktion ihrer Unterklassen (Mod-5.21)
Vererbungshierarchie: Unterklassen erben von ihrer Oberklasse
- **Objektbaum:** Objekt enthält (Referenzen auf) Teilobjekte
- **Kantorowitsch-Baum:** Operator mit seinen Operanden (Mod-5.22)
- **Strukturbaum:** (Programm-)Struktur definiert durch eine kontextfreie Grammatik (Mod-5.23)

Identifikation gleicher Teilbäume führt zu azyklischen Graphen (DAGs).

Vorsicht:

Identifikation muss mit der **Bedeutung der Kanten verträglich** sein;
z. B. Ein Gegenstand kann nicht dasselbe Objekt mehrfach als Teil enthalten,
wohl aber mehrere Objekte derselben Art.

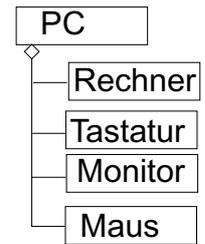
Klassen- und Objekthierarchien

Kompositionsbeziehung im Klassendiagramm (UML, Folie 6.19ff):

Knoten: **Klassen**

Kanten: definieren, **aus welcher Art von Objekten** ein Objekt **besteht**
 z. B. ein Objekt der Klasse PC **besteht aus**
 einem Rechner-Objekt, einem Tastatur-Objekt, ...

Diese Beziehung zwischen den Klassen könnte **auch ein allgemeiner Graph** sein

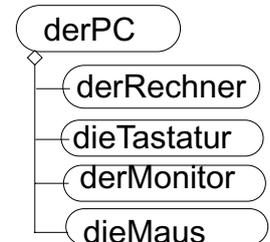


Objektbaum im Objektdiagramm (fast UML):

Knoten: **Objekte**

Kanten: definieren, **aus welchen Objekten** ein Objekt **besteht**
 z. B. dieser PC besteht aus, diesem Rechner, ...

Diese Beziehung muss **konzeptionell ein Baum** sein.



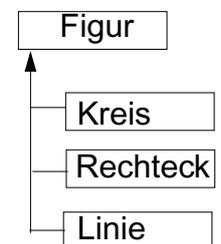
Vererbungsbeziehung im Klassendiagramm (UML Notation):

Knoten: **Klassen**

Kanten: Unterklasse **erbt von** -> Oberklasse
 Oberklasse **ist Abstraktion** <- ihrer Unterklassen
 Kanten sind zur Wurzel hin gerichtet

Baum bei Einfachvererbung (Java)

azyklischer Graph bei Mehrfachvererbung (C++)



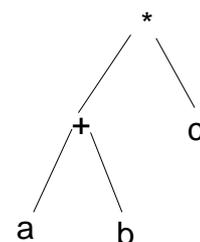
Kantorowitsch-Bäume

Darstellung der Struktur von Termen, Formeln, Ausdrücken
 (siehe Mod-3.6)

Knoten: Operator, Blattoperand

Kanten: Verbindung zu den Operanden eines Operators
 Die Kanten sind geordnet (Kantenmarkierung):
 erster, zweiter, ... Operand

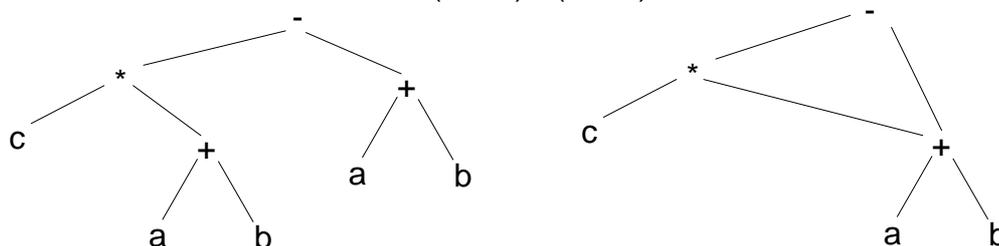
$$(a + b) * c$$



Identifikation gleicher Teilbäume führt zu azyklischen Graphen (DAGs):

Z. B. identifizieren Übersetzer gleiche Teilbäume, um Code zu erzeugen,
 der sie nur einmal auswertet:

$$c * (a + b) - (a + b)$$



Strukturbäume zu kontextfreien Grammatiken

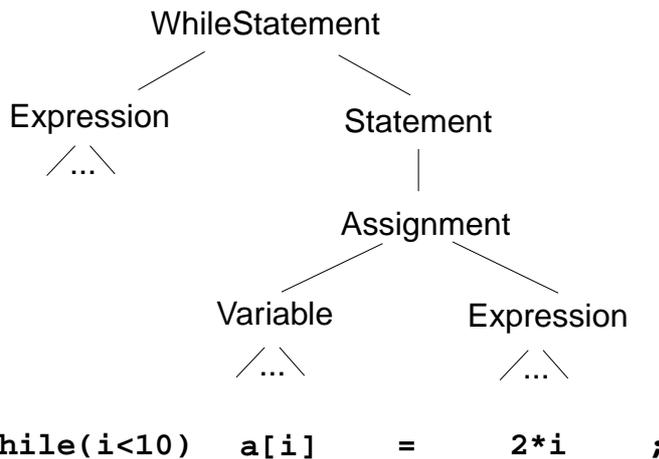
Kontextfreie Grammatiken definieren die Struktur von Programmen, Texten oder Daten. Ein Programm, Text oder strukturierte Daten werden als Strukturbaum dargestellt.

Knoten: Programmkonstrukt (Nichtterminal der Grammatik)

Kante: Bezug zu Bestandteilen des Programmkonstruktes (Produktion der Grammatik)

Für die Repräsentation von Texten sind die **Kanten geordnet** (Kantenmarkierung)

Strukturbaum:



Produktionen aus der kontextfreien Grammatik:

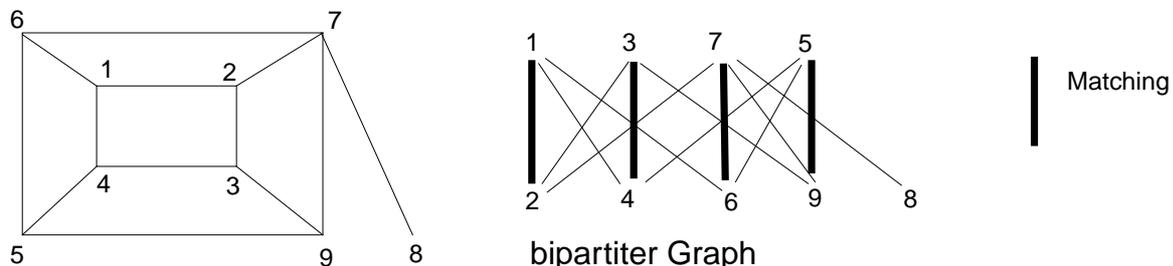
- Statement ::= Assignment
- Statement ::= WhileStatement
- ...
- WhileStatement ::= Expression Statement
- Assignment ::= Variable Expression

5.5 Zuordnungsprobleme

Aufgabenklasse **paarweise Zuordnung (Matching)**:

Im ungerichteten Graphen $G = (V, E)$ modelliert eine Kante $\{a, b\}$ „a passt zu b“, ggf. mit einer Kantenmarkierung als Abstufung

Gesucht ist eine **maximale Menge unabhängiger Kanten**, das ist ein Teilgraph M mit allen Knoten aus V und möglichst vielen Kanten aus E , so dass der **Grad der Knoten höchstens 1** ist. M heißt ein **Matching** der Knoten von G .



Graph G heißt **bipartit**, wenn V in **2 disjunkte Teilmengen** $V = V_1 \cup V_2$ zerlegt werden kann, so dass jede Kante zwei Knoten aus verschiedenen Teilmengen verbindet.

Häufig liefert die Aufgabenstellung schon bipartite Graphen, sogenannte **Heiratsprobleme**:

Mann - Frau

Aufgabe - Bearbeiter

Verbraucher - Produkte

Konfliktfreie Knotenmarkierung (Färbung)

Aufgabenklasse **konfliktfreie Knotenmarkierung (Färbung):**

Im ungerichteten Graphen $G = (V, E)$ modelliert eine Kante $\{a, b\}$ „**a ist unverträglich mit b**“,

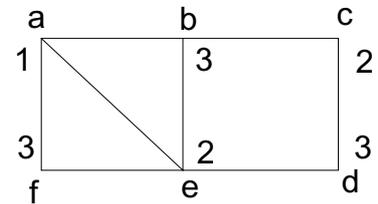
Gesucht ist eine Knotenmarkierung Färbung: $V \rightarrow \mathbb{N}$ („Farben“),
so dass durch eine **Kante verbundene Knoten verschiedene Marken haben**

Die **chromatische Zahl** eines Graphen G ist die minimale Zahl verschiedener „Farben“, die nötig ist, um G konfliktfrei zu markieren.

Es gilt: chromatische Zahl $\leq 1 + \text{maximaler Knotengrad}$

Anwendungen:

Knoten:	Kante:	Farbe / Marke:
Staat auf Landkarte	gemeinsame Grenze	Farbe
Partygast	unverträglich	Tisch
Kurs	haben gemeinsame Teilnehmer	Termin
Prozess	benötigen gleiche Ressource	Ausführungszeitpunkt
Variable im Programm	gleichzeitig lebendig	Registerspeicher



5.6 Abhängigkeitsprobleme

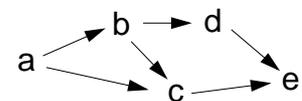
Graphen modellieren **Abhängigkeiten** zwischen Operationen und **Ausführungsreihenfolgen** von Operationen.

Abhängigkeitsgraph: gerichtet, azyklisch, voneinander abhängige Operationen.

Aufgaben dazu: sequentielle oder parallele **Anordnungen finden** (engl. **scheduling**).

Knoten: Operation, Ereignis; ggf. mit Dauer markiert

Kante: $a \rightarrow b$ a ist **Vorbedingung** für b oder b **benutzt** Ergebnis von a oder a liest oder schreibt Ressource bevor b sie überschreibt



Anwendungen:

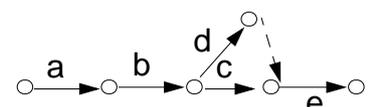
- **Projektplanung** mit abhängigen Teilaufgaben (PERT, CPM)
- abhängige **Transaktionen** mit einer Datenbank
- **Anordnung von Code** für die parallele Auswertung von Ausdrücken (Übersetzer)

Kritischer Pfad: längster Weg von einem Anfangsknoten zu einem Endknoten

Duale Modellierung:

Knoten: Ereignis, Anfang und Ende einer Operation

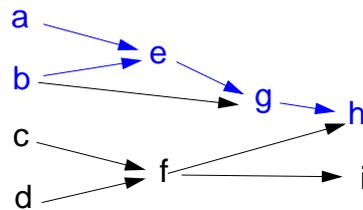
Kante: Operation, ggf. mit Dauer markiert



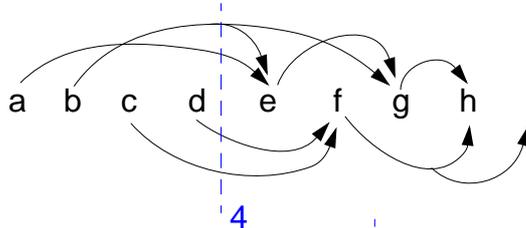
Anordnung von Abhängigkeitsgraphen

Anordnungsaufgaben:
 gegebener Abhängigkeitsgraph

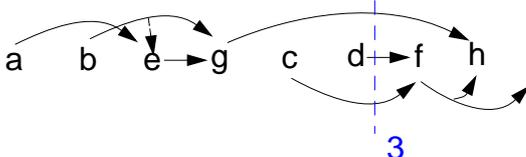
kritischer Pfad



sequentielle Anordnung der Knoten,
 so dass **alle Kanten vorwärts** zeigen.

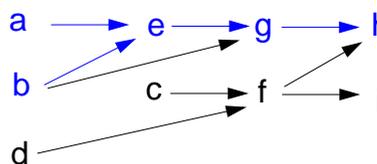


Meist sollen **Randbedingungen** erfüllt werden,
 z. B. geringste **Anzahl gleichzeitig benötigter
 Zwischenergebnisse im Speicher**



parallele Anordnung mit
 beschränkter Parallelität 3

Länge: 4 Schritte (Operationen)

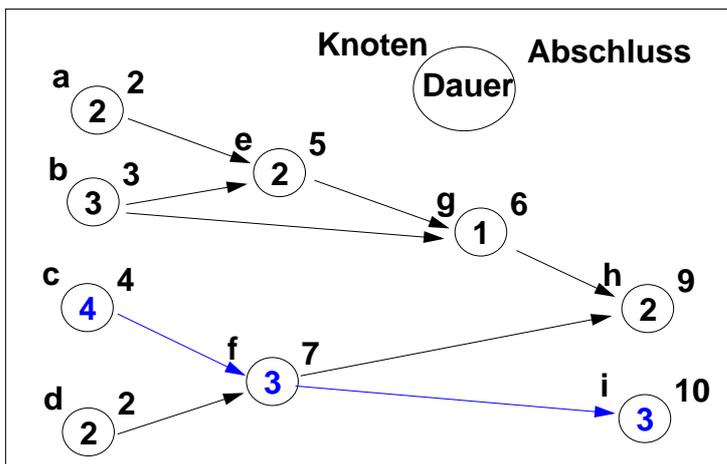


Operationen unterschiedlicher Dauer

Zwei **Knotenmarkierungen:**

Dauer der Operation und
frühester Abschlusstermin
 = max. Abschluss der Vorgänger
 + Dauer des Knotens

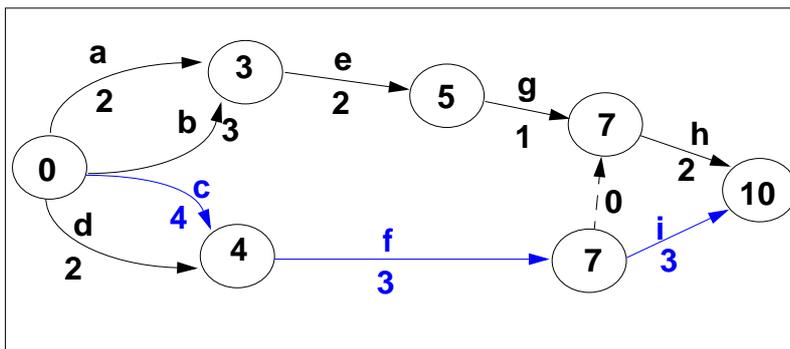
Kritischer Pfad gemäß maximaler
 Summe der Dauer der Operationen



Duale Modellierung:

Kante: Operation
 mit **Dauer** als Marke
 Mehrfachkanten, Multigraph

Knoten: Ereignis
 „vorangehende Operationen
 sind abgeschlossen“
 mit frühestem **Abschlusstermin**
 als Marke



Ablaufgraphen

Gerichteter Graph (auch zyklisch) modelliert Abläufe.

Knoten: Verzweigungsstelle, Zustand

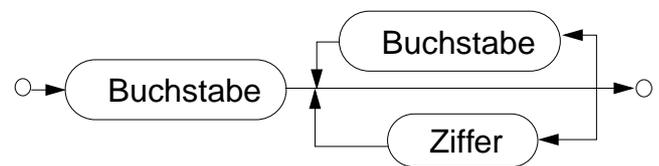
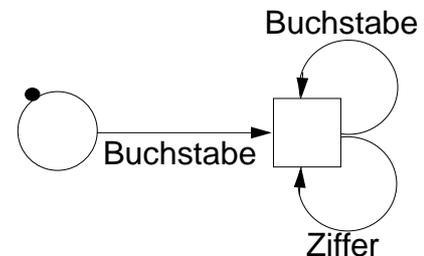
Kanten: Fortsetzungsmöglichkeit

Jeder **Weg durch den Graphen** beschreibt einen **potenziellen Ablauf**

Die **Folge der Markierungen eines Weges** kann einen **Satz einer Sprache** modellieren.

Anwendungen:

- **Endlicher Automat** (siehe Kapitel 6)
modelliert **Folgen von Zeichen**, Symbolen, ...
Knoten: Zustand
Kante: Übergang markiert mit Zeichen
- **Syntaxdiagramm**
modelliert **Folgen von Zeichen**, Symbolen, ...
Knoten: markiert mit Zeichen
Kante $a \rightarrow b$: „auf a kann b folgen“
dual zum endlichen Automaten
- **Aufrufgraphen** (siehe Mod-5.30)
- **Ablaufgraphen** (siehe Mod-5.31)



Aufrufgraphen

Gerichteter Aufrufgraph: Aufrufbeziehung zwischen Funktionen in einem Programm;
wird benutzt in **Übersetzern** und in **Analysewerkzeugen** zur Software-Entwicklung.

Knoten: Funktion im Programm

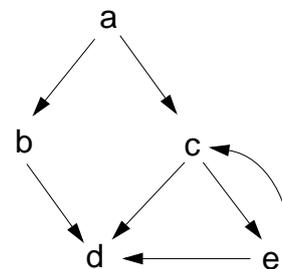
Kante $a \rightarrow b$: Rumpf der Funktion a enthält einen Aufruf der Funktion b; **a könnte b aufrufen**

Zyklus im Aufrufgraph:

Funktionen, die sich **wechselweise rekursiv** aufrufen, z. B. (c, e, c)

Fragestellungen z. B.

- Welche Funktionen sind nicht rekursiv?
- Welche Funktionen sind nicht (mehr) erreichbar?
- Indirekte Wirkung von Aufrufen,
z. B. nur e verändere eine globale Variable x;
welche Aufrufe lassen x garantiert unverändert? b, d



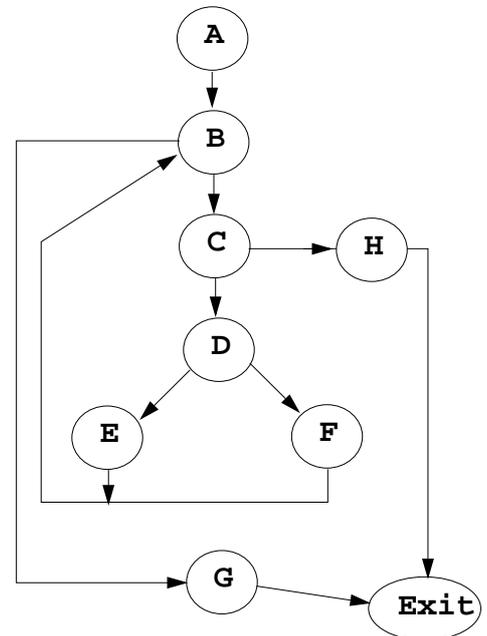
Programmablaufgraphen

Gerichteter Graph, modelliert **Abläufe** durch ein **verzweigtes Programm** (bzw. Funktion); wird benutzt in **Übersetzern** und in **Analysewerkzeugen** zur Software-Entwicklung.

Knoten: unverzweigte Anweisungsfolge (Grundblock), mit Verzweigung (Sprung) am Ende

Kante: potenzieller Nachfolger im Ablauf

<code>ug = 0;</code>	A
<code>og = obereGrenze;</code>	
<code>while (ug <= og)</code>	B
<code>{ mitte = (ug + og) / 2;</code>	C
<code> if (a[mitte] == x)</code>	
<code> return mitte;</code>	H
<code> else if (a[mitte] < x)</code>	D
<code> ug = mitte + 1;</code>	E
<code> else og = mitte - 1;</code>	F
<code>}</code>	
<code>return nichtGefunden;</code>	G



Fragestellungen, z. B.

- Menge von Wegen, die den **Graph überdecken**, **Software-Testen**
- Wege mit bestimmten Eigenschaften, **Datenflussanalyse**

Zusammenfassung zu Graphen

Problemklassen:

- Wegeprobleme
- Verbindungsprobleme
- Entscheidungsbäume
- hierarchische Strukturen
- Zuordnungsprobleme
- Abhängigkeitsprobleme
- Anordnungen in Folgen
- verzweigte Abläufe

Kanten- und Knotenbedeutung:

- verbunden, benachbart, ...
- Entscheidung, Alternative, Verzweigung
- Vorbedingung, Abhängigkeit
- (Un-)Verträglichkeit
- allgem. symmetrische Relation
- besteht aus, enthält, ist-ein
- (Halb-)Ordnungsrelation

Kanten-, Knotenmarkierungen:

- Entfernung, Kosten, Gewinn, ... bei Optimierungsproblemen
- „Färbung“, disjunkte Knotenmengen bei Zuordnungsproblemen
- Symbole einer Sprache