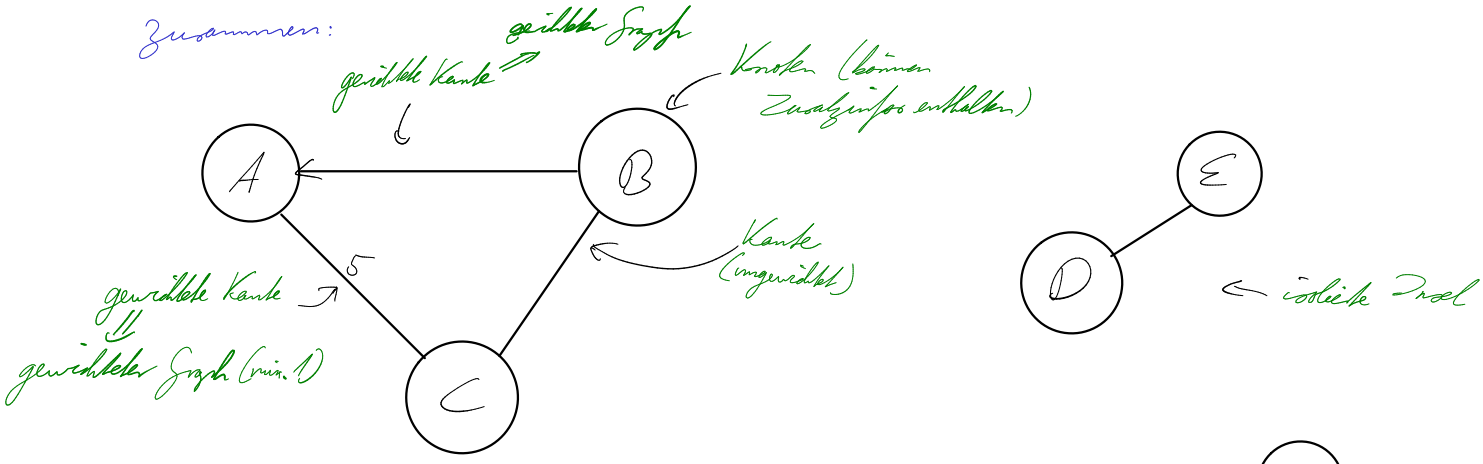
Graphen

Def.: Ein Graph besteht aus einer endlichen Anzahl von Knoten und Kanten zusammen:



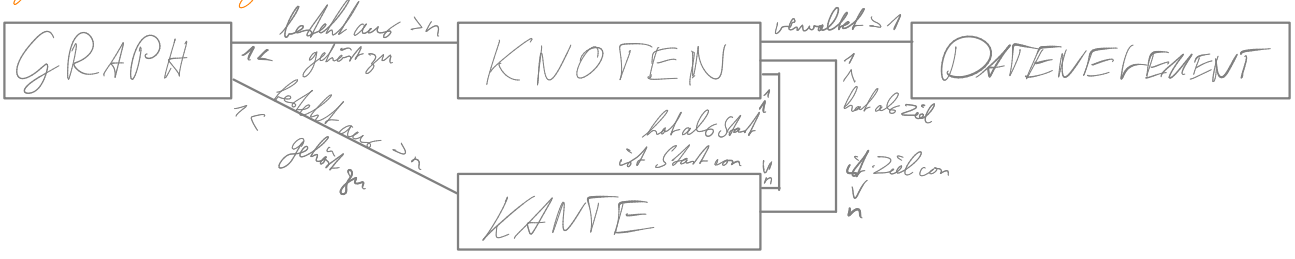
- zusammenhängender Graph: alle Knoten sind durch Kanten verbunden
- vollständiger Graph: „vollvermaschter" Graph
- Pfad / Weg: Folge von durch Kanten verbundenen Knoten (Startknoten zu Zielknoten)
  - einfacher Pfad: Kein Knoten wird doppelt besucht
  - Zyklus: Pfad, bei dem Startknoten = Endknoten

Einsatzbereiche:

- → Navigation (Kürzeste Strecke, Schnellste Strecke, ...) ⇒ Traveling Salesman Problem
- → Internet
- → Chemie
- → Stromnetz
- → Klassendiagramm mit Beziehungen

...

Implementierung:



Problem: Wie stelle ich einen Graphen mit n Knoten in Programmcode dar?

Lösung: Adjazenzmatrix = Tabelle, die Kantenverbindungen zwischen Knoten zeigt.

Adjazenzmatrix:

von \ nach	A	B	C	D	E	F	G	H
A	0	3	5	-1	-1	2	3	2
B	3	0	-1	-1	1	-1	5	-1
C	5	-1	0	8	7	6	-1	2
D	-1	-1	8	0	2	-1	2	-1
E	-1	1	7	2	0	5	5	-1
F	2	-1	6	-1	5	0	1	3
G	3	5	-1	2	5	1	0	6
H	2	-1	2	-1	-1	3	6	0

Spalten = nach-Knoten

keine Verbindung zwischen diesen Knoten

Namen der Knoten

Kantengewichte

Zeilen = von-Knoten

Symmetrieachse: nur bei ungerichteten Graphen

In Java: Kanten: 2D-Array int [Zeilen][Spalten]

Knoten: 1D-Array

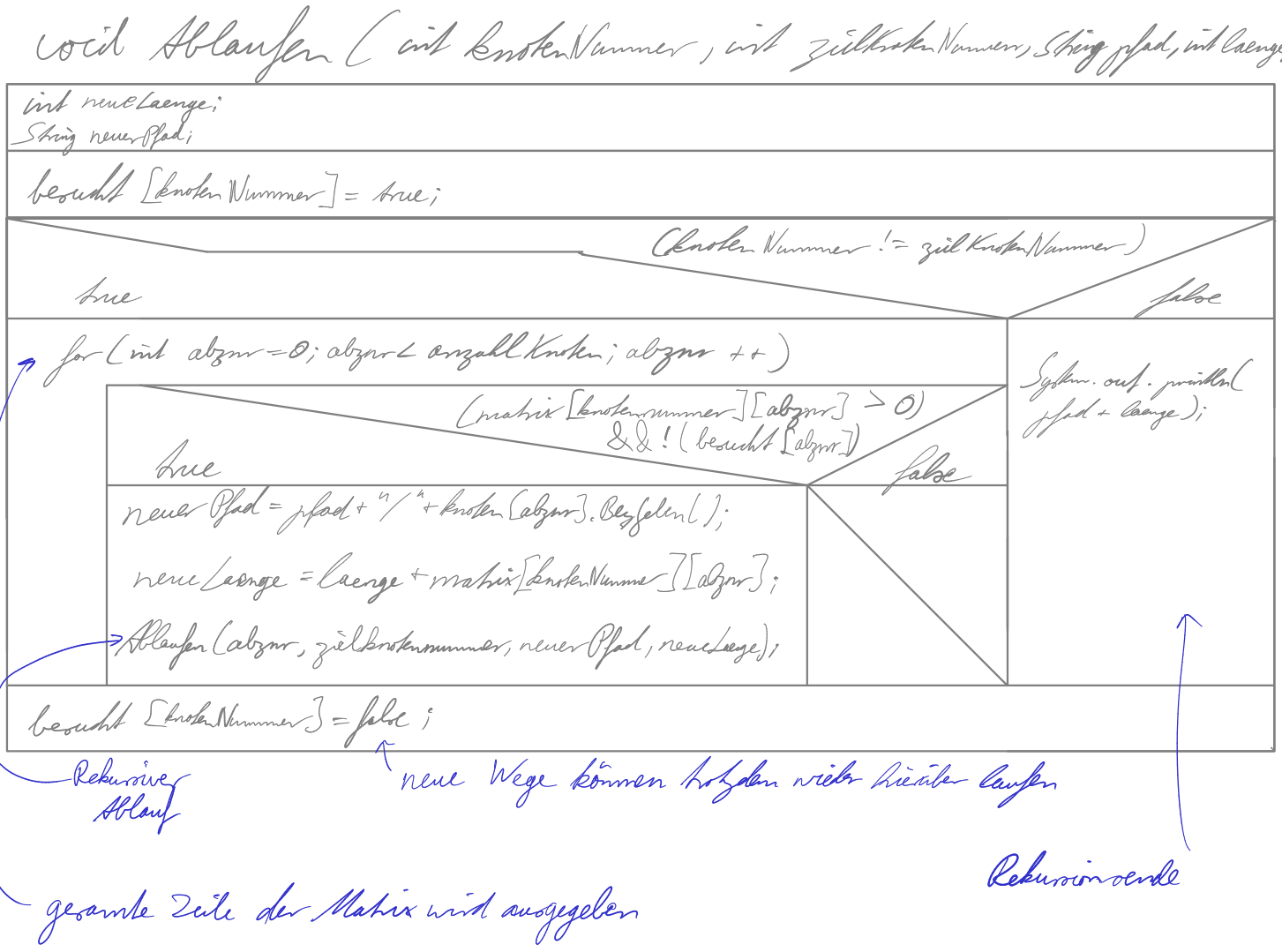
Achtung: Formel für maximale Anzahl an Kanten: $k_{max} = \frac{1}{2} \cdot n \cdot (n-1)$

Tiefensuche:

```
void WegeSuchen (String startKnoten, String zielKnoten
int startNummer = KnotenNummerGeben (startKnoten);
int zielNummer = KnotenNummerGeben (zielKnoten);
(startNummer != -1) && (zielNummer != -1) && (startNummer != zielNummer)
true:
    for (int i = 0; i < anzahlKnoten; i++)
        besucht[i] = false;
    Ablaufen (startNummer, zielNummer, startKnoten, 0);
false:
```

Statt der eigentlichen Suche

Davor: Initialisierungsphase


void Ablaufen (int KnotenNummer , int zielKnotenNummer, String pfad, int laenge)
int neueLaenge;
String neuerPfad;
besucht [knotenNummer] = true;
(knotenNummer != zielKnotenNummer)
true
false
for (int abgnr=0; abgnr< anzahlKnoten; abgnr ++)
System.out.println(pfad + laenge);
(matrix [knotennummer] [abgnr] > 0)
&& ! (besucht [abgnr])
true
false
neuer Pfad = pfad + "/" + knoten [abgnr].Bezeichner( );
neueLaenge = laenge + matrix [knotenNummer] [abgnr];
Ablaufen (abgnr, zielknotennummer, neuerPfad, neueLaenge);
besucht [knotenNummer] = false ;
Rekursiver Ablauf
neue Wege können Knoten wieder überlaufen
Rekursionsende
gesamte Zeile der Matrix wird ausgegeben