# Einführung in die Informatik 2

## – Bäume & Graphen –

Sven Kosub

AG Algorithmik/Theorie komplexer Systeme
Universität Konstanz

E 202 | Sven.Kosub@uni-konstanz.de | Sprechstunde: nach Vereinbarung

Sommersemester 2010

# Grundbegriffe: Graphen

Bisher: Bäume als Verallgemeinerung von Listen

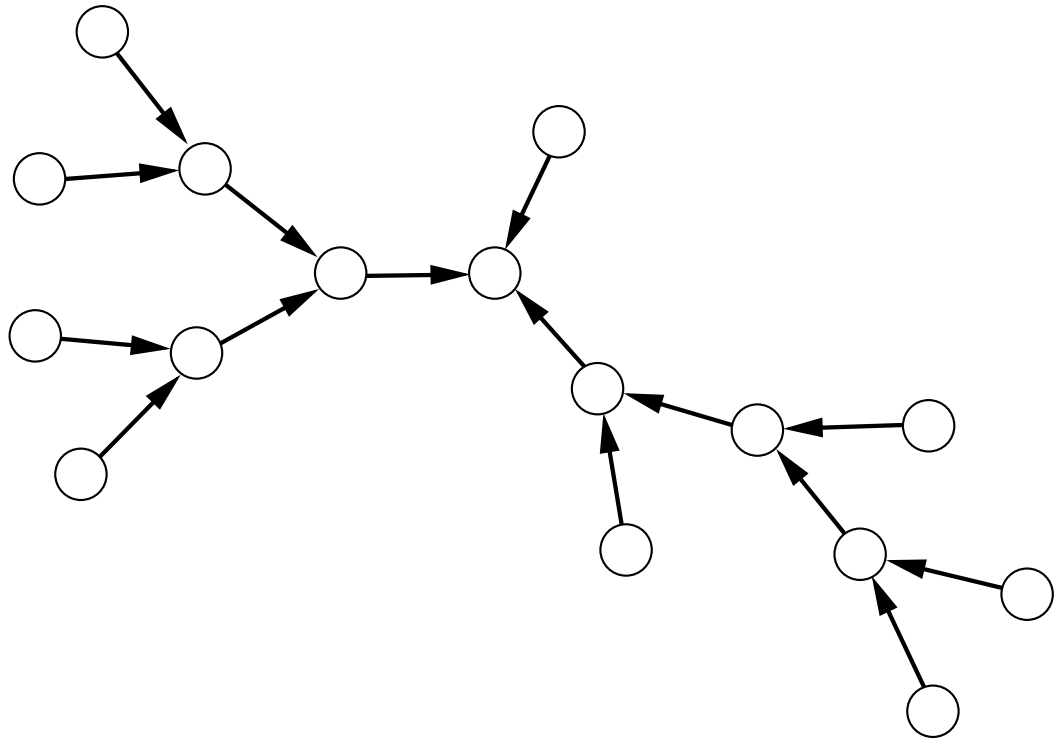
Jetzt: Bäume als Spezialfall von Graphen

Ein gerichteter Graph ist ein Paar $G = (V, E)$ bestehend aus

- einer (nicht-leeren) Menge $V$ von Knoten oder Ecken (engl. *vertices, nodes*)
- einer Menge $E$ von gerichteten Kanten (engl. *edges, arcs*):
  - Kantenmenge $E$ ist eine binäre Relation auf $V$, d.h. $E \subseteq V \times V$
  - Kante $(u, v) \in E$ ist von $u$ nach $v$ gerichtet, symbolisch $u \longrightarrow v$; $u$ heißt Kopf (engl. *head*), $v$ heißt Ende (engl. *tail*)
- in Graphen als Datenstruktur dürfen sowohl Knoten als auch Kanten Etikette (engl. *labels*) besitzen

# Grundbegriffe: Graphen

Baum mit Wurzel als gerichteter Graph:



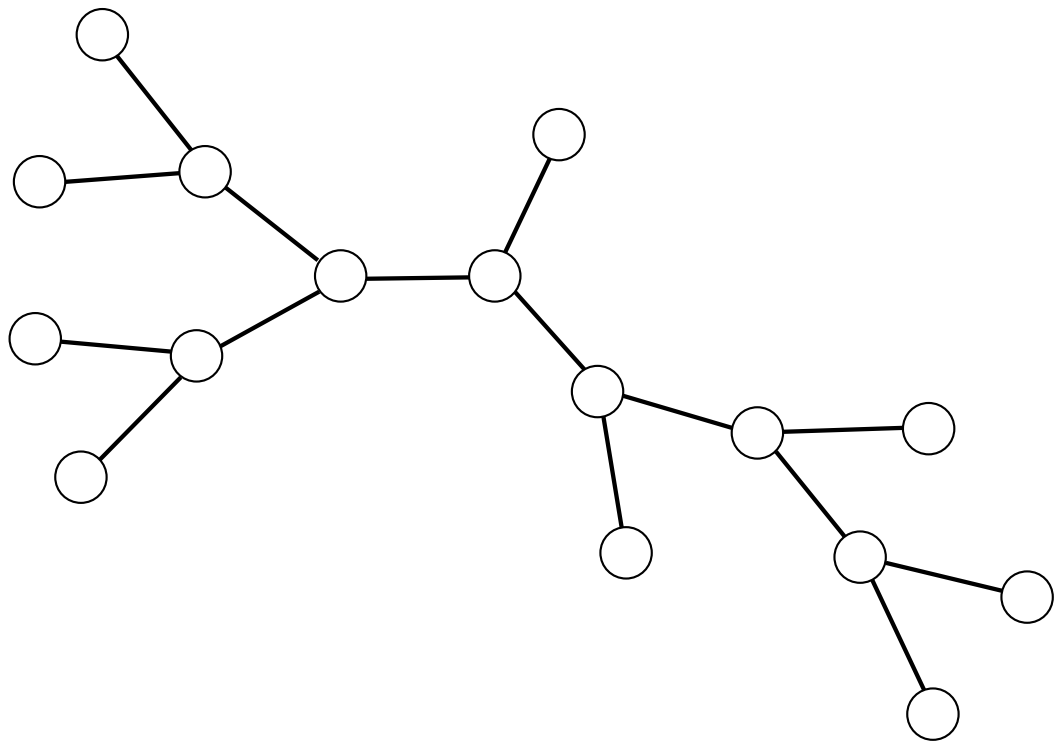
# Grundbegriffe: Graphen

Ein ungerichteter Graph ist ein Paar $G = (V, E)$ bestehend aus

- einer (nicht-leeren) Menge $V$ von Knoten oder Ecken
- einer Menge $E$ von ungerichteten Kanten:
  - Kantenmenge $E$ ist eine symmetrische binäre Relation auf $V$, d.h. $(u, v) \in E \iff (v, u) \in E$
  - ungerichtete Kanten werden meist durch Mengen beschrieben, also $\{u, v\} \in E$ (in Mengen spielt die Reihenfolge der Elemente keine Rolle)
  - alternativ: nur ein Paar angeben und festlegen, dass Graph ungerichtet sein soll

# Grundbegriffe: Graphen

Baum als ungerichteter Graph:



Wo ist die Wurzel?

# Grundbegriffe: Graphen

Betrachten ungerichteten Graphen $G = (V, E)$

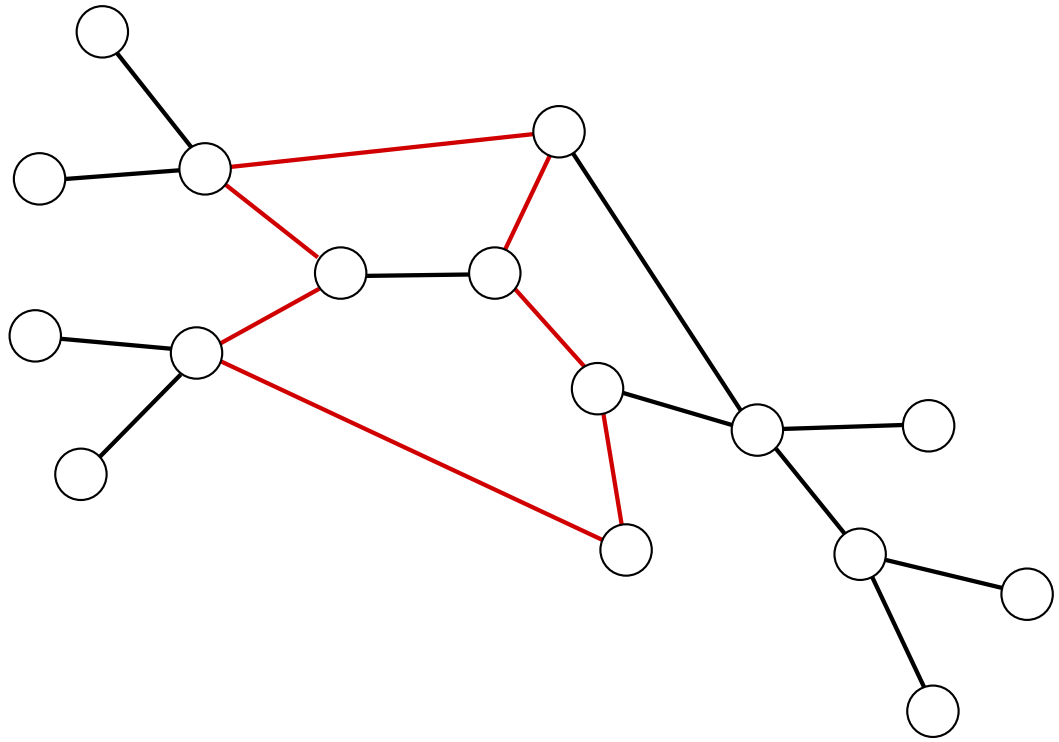
- Ein Weg (der Länge $k$) in $G$ ist Folge von Knoten $(v_0, v_1, \ldots, v_k)$, so dass jeweils $\{v_{i-1}, v_i\}$ eine Kante in $E$ ist
  (Beachte: Weg der Länge 0 besteht nur aus einem einzigen Knoten)
- Ein Pfad (der Länge $k$) in $G$ ist eine Weg $(v_0, v_1, \ldots, v_k)$, wobei jeder Knoten nur einmal vorkommt
- Ein Kreis in $G$ ist ein Weg, bei dem der erste und der letzte Knoten übereinstimmen und bei dem keine Kante mehrfach benutzt wird

Eigenschaften eines ungerichteten Graphen $G = (V, E)$:

- $G$ ist zusammenhängend $\iff_{\text{def}}$ für alle Knoten $u, v \in V$ gibt es einen Weg mit $u$ als erstem und $v$ als letztem Knoten
- $G$ ist ein Baum $\iff_{\text{def}}$ $G$ ist zusammenhängend und kreisfrei
- $G$ ist ein Wald $\iff_{\text{def}}$ $G$ ist kreisfrei

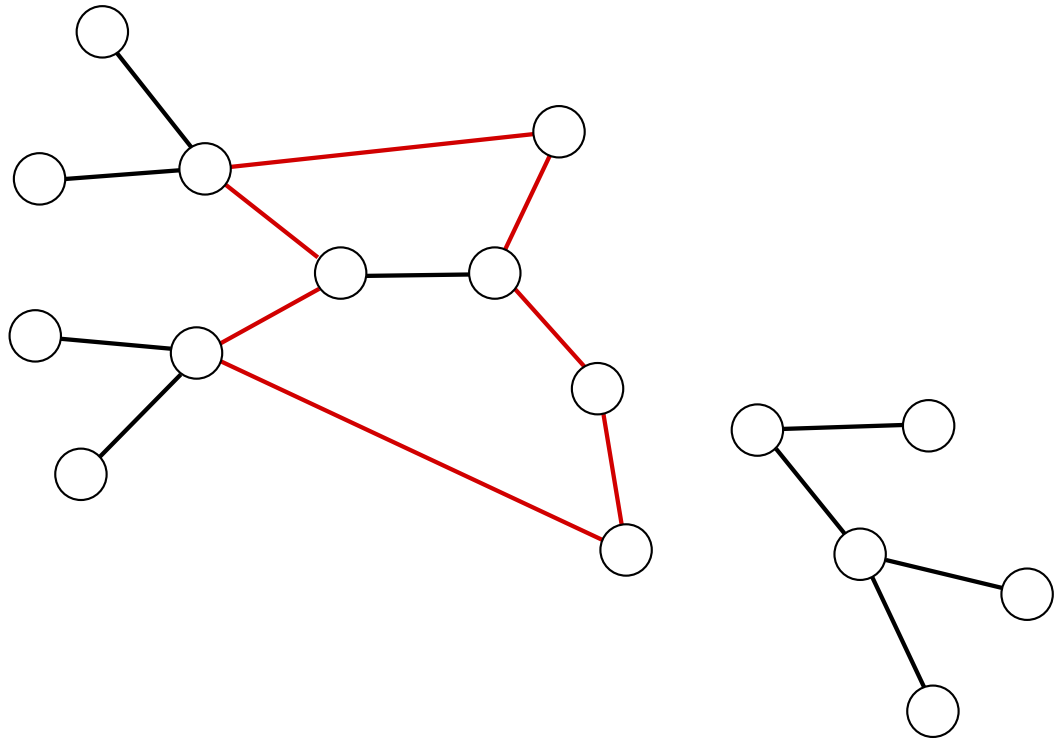
# Grundbegriffe: Graphen

ungerichteter Graph mit Kreis, zusammenhängend (kein Baum):



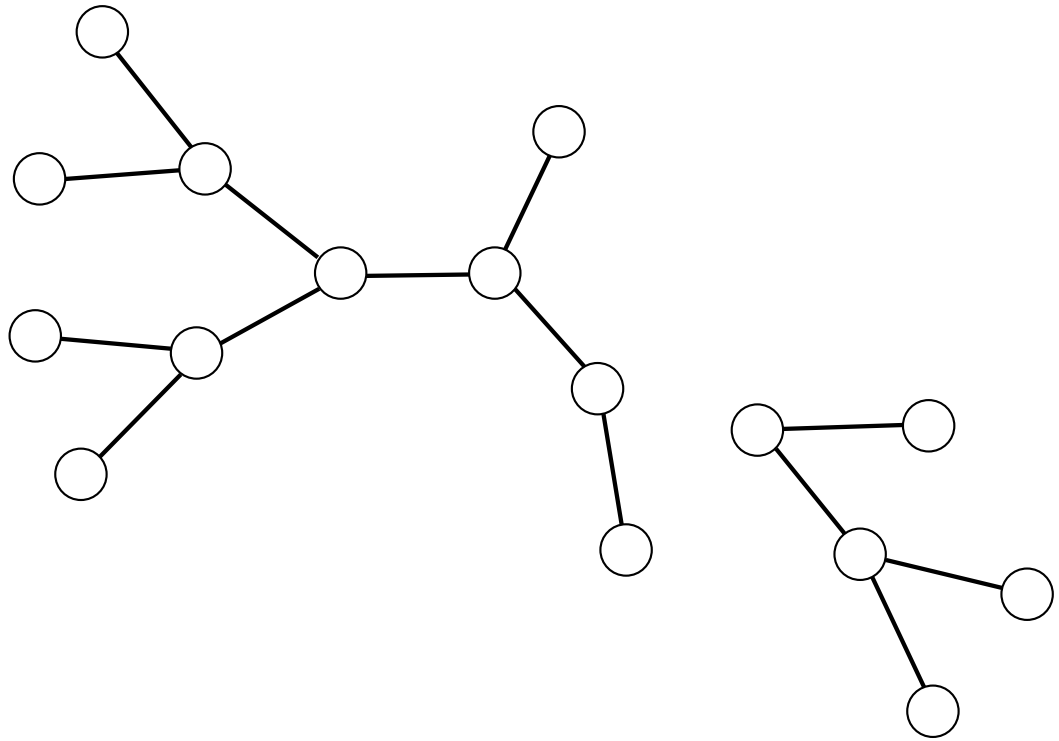
# Grundbegriffe: Graphen

ungerichteter Graph mit Kreis, nicht zusammenhängend (kein Wald):



# Grundbegriffe: Graphen

Wald (kein Baum):



# Grundbegriffe: Graphen

Es sei $G = (V, E)$ ein ungerichteter Graph

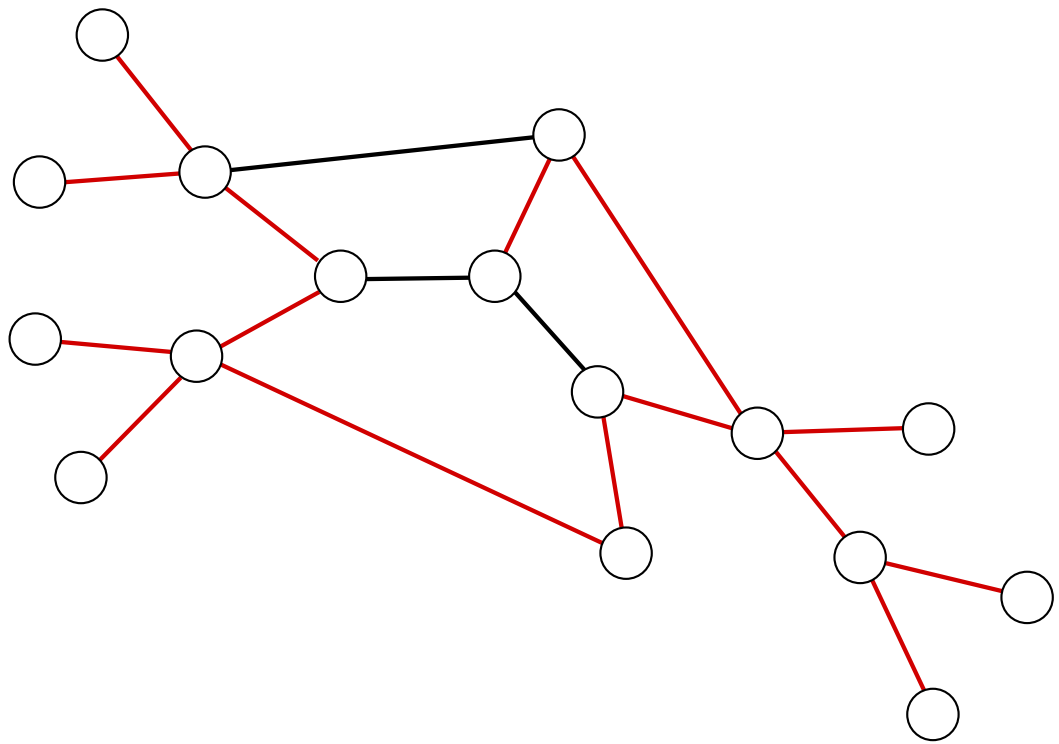
- $G' = (V', E')$ ist Teilgraph von $G \iff_{\text{def}} V' \subseteq V$ und $E' \subseteq E$
- Ein Spannbaum von $G$ ist ein Teilgraph $G' = (V', E')$ von $G$ mit folgenden Eigenschaften:
  - $G'$ ist ein Baum
  - $V' = V$

Fakten:

- Jeder zusammenhängende, ungerichtete Graph enthält einen Spannbaum
- $G$ ist Baum $\iff$ $G$ ist zusammenhängend und $m = n - 1$
- $G$ ist Baum $\iff$ zwischen jeweils zwei Knoten von $G$ gibt es genau einen Pfad

# Grundbegriffe: Graphen

Graph mit Spannbaum:



Wie finden wir einen Spannbaum?

# Implementierung von Graphen

Konvention:

- $n$ ist die Anzahl der Knoten eines Graphen
- $m$ ist die Anzahl der Kanten eines Graphen

Gebräuchliche Implementierungen:

- Kantenlisten
- Adjazenzlisten
- Adjazenzmatrix

wichtige Operationen auf Graphen:

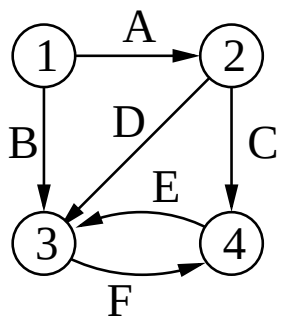
- Adjazenztest: Sind zwei Knoten durch eine Kante verbunden?
- Bestimmung inzidenter Kanten: Welche Kanten berühren einen gegebenen Knoten?

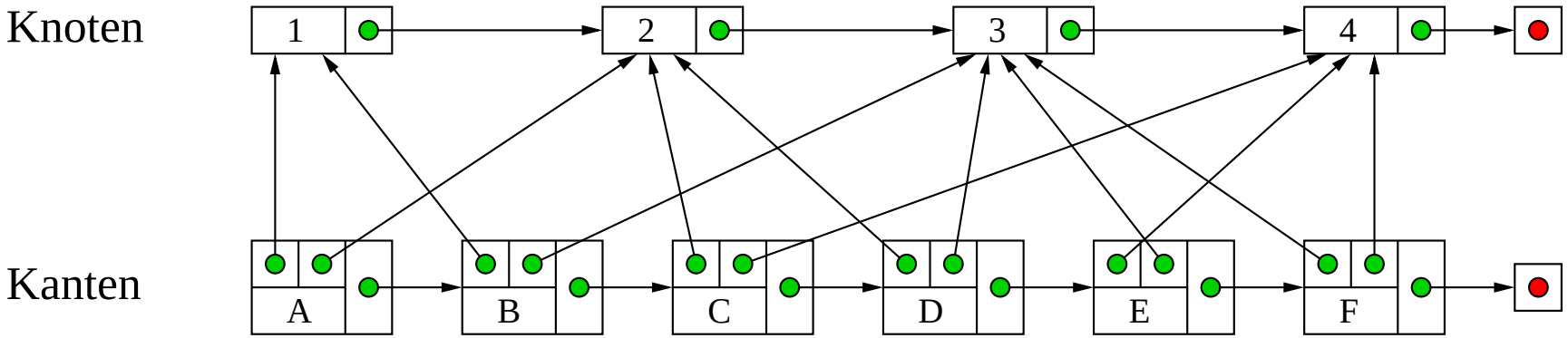
# Implementierung von Graphen: Kantenlisten

- Kantenlisten bestehen aus Knoten- und Kantenobjekten
- Knotenobjekt für einen Knoten $v \in V$:
  - Zeiger auf Datenobjekt (Label bzw. Etikett)
  - Zeiger auf nächstes Knotenobjekt
- Kantenobjekt für eine Kante $e \in E$:
  - Zeiger auf Datenobjekt (Label bzw. Etikett)
  - Zeiger auf die beteiligten Knoten (`head` und `tail`)
  - Markierung für „gerichtet“ oder „ungerichtet“
  - Zeiger auf nächstes Kantenobjekt

# Implementierung von Graphen: Kantenlisten

Kantenliste für Beispielgraph:



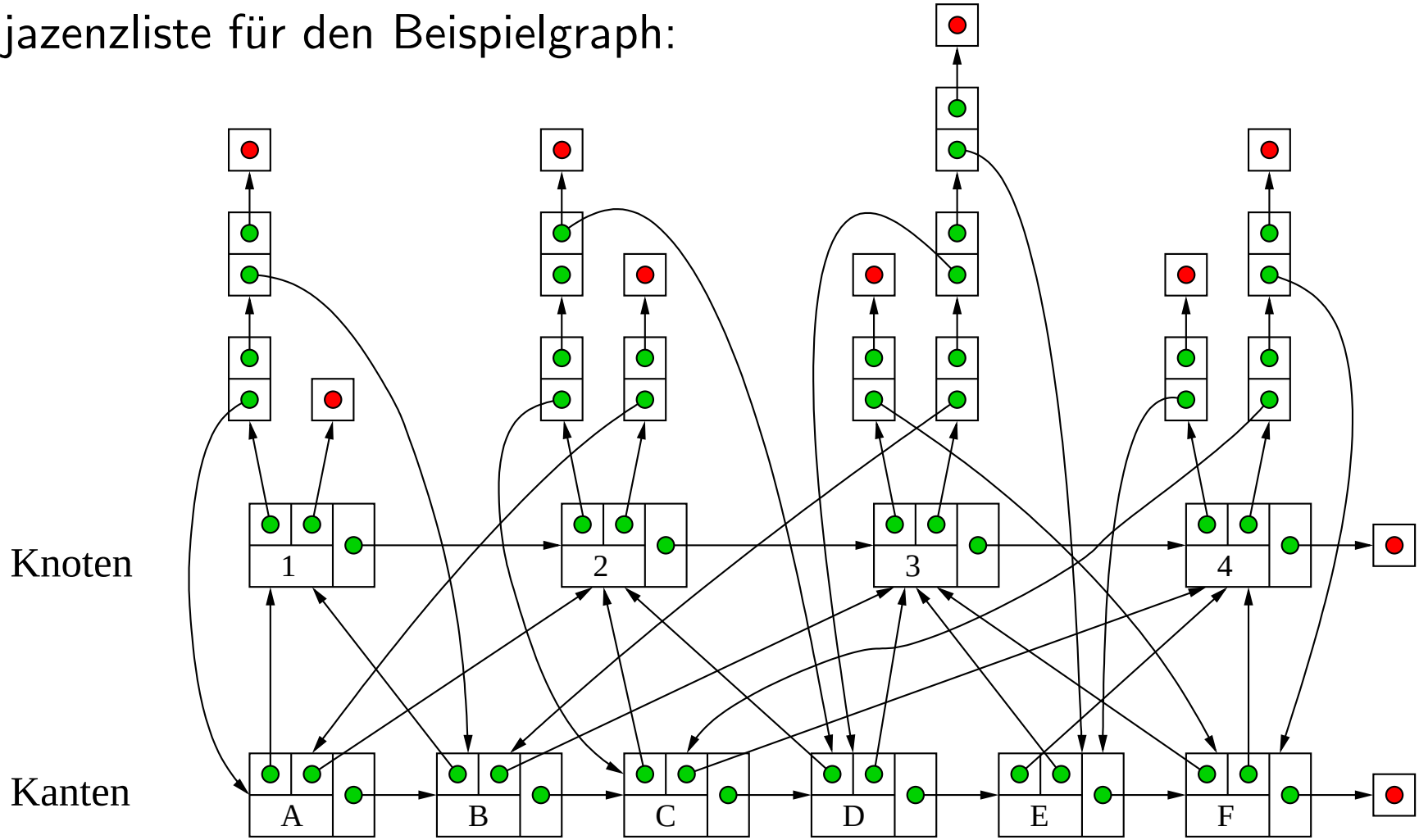

- Bestimmung inzidenter Kanten an einem Knoten in $O(m)$
- Adjazenztest in $O(m)$

# Implementierung von Graphen: Adjazenzlisten

- Adjazenzlisten bestehen aus Knoten- und Kantenobjekten
- Knotenobjekt für einen Knoten $v \in V$:
  - Zeiger auf Datenobjekt (Label bzw. Etikett)
  - Zeiger auf Listen für eingehende (`inbound`) und ausgehende Kanten (`outbound`)
  - Zeiger auf nächstes Knotenobjekt
- Kantenobjekt für eine Kante $e \in E$:
  - Zeiger auf Datenobjekt (Label bzw. Etikett)
  - Zeiger auf die beteiligten Knoten (`head` und `tail`)
  - Markierung für „gerichtet“ oder „ungerichtet“
  - Zeiger auf nächstes Kantenobjekt

# Implementierung von Graphen: Adjazenzliste

Adjazenzliste für den Beispielgraph:




- Bestimmung inzidenter Kanten an einem Knoten in $O(\deg(v))$
- Adjazenztest in $O(\min\{\deg(u), \deg(v)\})$

# Implementierung von Graphen: Adjazenzmatrix

Adjazenzmatrix $A(G)$ für Graphen $G = (V, E)$ mit $V = \{0, \ldots, n-1\}$ ist eine $n \times n$-Matrix mit

$$a_{ij} =_{\text{def}} \begin{cases} 1 & \text{wenn } (i,j) \in E \\ 0 & \text{wenn } (i,j) \notin E \end{cases}$$

Beispielgraph $G = (V, E)$ hat folgende Adjazenzmatrix:

$$A(G) = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

# Implementierung von Graphen: Adjazenzmatrix

- Adjazenzmatrix (als Datenstruktur) besteht aus Knotenobjekten, Kantenobjekten und einem Matrixobjekt
- Knotenobjekt für einen Knoten $v \in V$:
  - Zeiger auf Datenobjekt (Label bzw. Etikett)
  - Index von $v$, d.h. seine Nummer in $V$
  - Zeiger auf nächstes Knotenobjekt
- Kantenobjekt wie bei Kantenlisten
- Matrixobjekt als $n \times n$-Reihung $A$ mit `A[i][j]` enthält Zeiger auf Kantenobjekt für Kante $e = (i, j)$ (für ungerichtete Graphen: `A[i][j]=A[j][i]`), sonst `null`

- Bestimmung inzidenter Kanten an einem Knoten in $O(\min\{n, m\})$
- Adjazenztest in $O(1)$
- Problem: stets $O(n^2)$ Speicherplatz unabhängig von Anzahl tatsächlich vorhandener Kanten (nur für dichte Graphen sinnvoll)