Outline

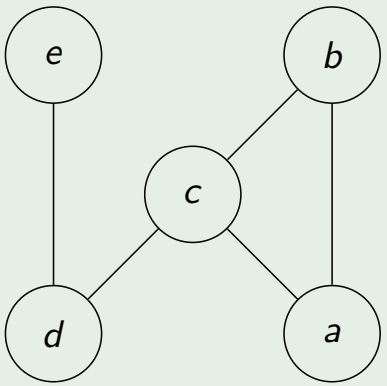
Zusammenhängend

Zusammenhängend

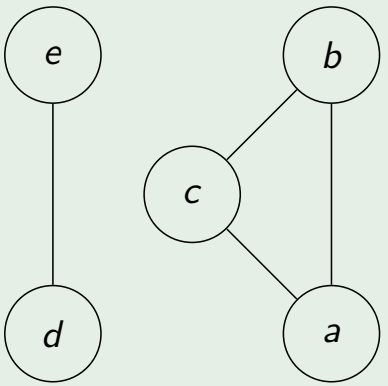
Ein Graph $G = (V, E)$ heißt *zusammenhängend*, wenn gilt:

$$\forall u, v \in V \text{ existiert ein Pfad } P = (u \dots v)$$

Beispiel



Zusammenhängend



Unzusammenhängend

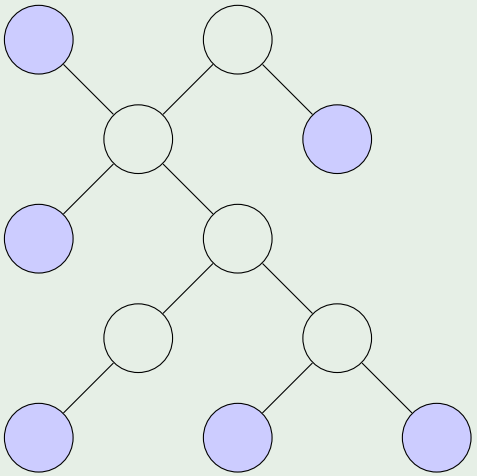
Kreisfreiheit

Baum

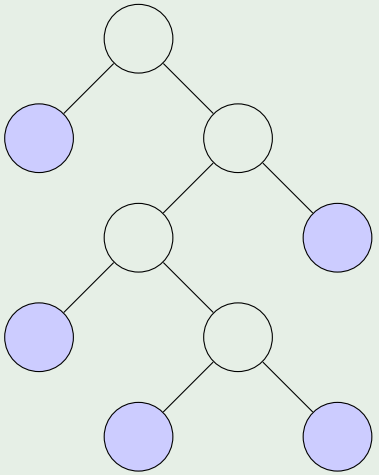
Ein Baum ist ein zusammenhängender, kreisfreier Graph.

Ein Knoten v eines Baumes mit Grad $deg(v) = 1$ heißt Blatt.

Beispiel



Baum



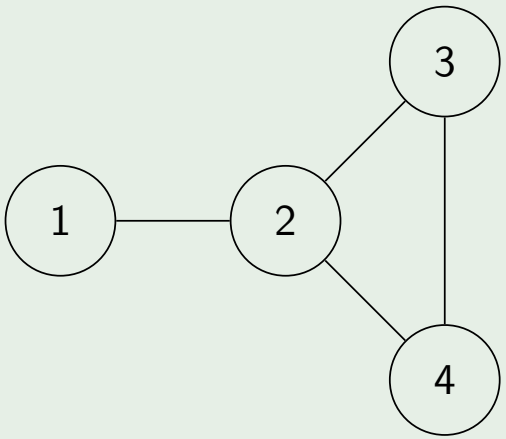
Binärbaum

Speicherung von Graphen

Adjazenzliste

- Darstellung des Graphen $G = (V, E)$ durch Listen.
- Pro Knoten eine Liste der adjazenten Knoten.

Beispiel



$$
\begin{array}{lcl}
1 & \rightarrow & 2 \\
2 & \rightarrow & 1,\quad 3,\quad 4 \\
3 & \rightarrow & 2,\quad 4 \\
4 & \rightarrow & 2,\quad 3
\end{array}
$$

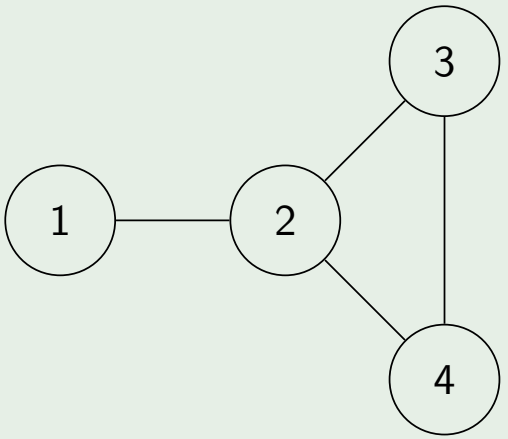
Speicherung von Graphen

Adjazenzmatrix

- Darstellung des Graphen $G = (V, E)$ als Matrix A.
- Eine Zeile und eine Spalte pro Knoten $v_i \in V$.
- Eintrag in Spalte i und Zeile j:

$$a_{i,j} = \begin{cases} 1 & , \text{falls } \{v_i, v_j\} \in E \\ 0 & , \text{falls } \{v_i, v_j\} \notin E \end{cases}$$

Beispiel



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Speicherung von Graphen

Vergleich

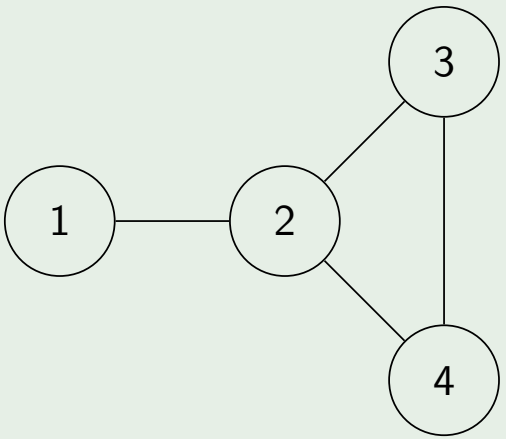
Adjazenzliste

- \+ Speicherverbrauch für $2|E|$ Einträge
- – Überprüfen ob eine Kante enthalten ist
- \+ Erstellen der Liste adjazenter Knoten

Adjazenzmatrix

- – Speicherverbrauch für $|V|^2$ Einträge
- \+ Überprüfen ob eine Kante enthalten ist
- – Erstellen der Liste adjazenter Knoten

Beispiel



$$
\begin{array}{lcl}
1 & \rightarrow & 2 \\
2 & \rightarrow & 1,\ 3,\ 4 \\
3 & \rightarrow & 2,\ 4 \\
4 & \rightarrow & 2,\ 3
\end{array}
$$

$$
A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}
$$

Breitensuche

Warteschlange (Queue)

- Liste von Objekten
- Hinzufügen von Objekte am Ende der Liste (*Q.Insert(u)*)
- Entfernen von Objekte am Anfang der Liste (*Q.Dequeue()*)
- Testen auf Leerheit der Liste (*Q.IsEmpty()*)
- Auch als *FIFO-Warteschlange* bezeichnet.

Breitensuche

- Verfahren zum durchlaufen aller Knoten
- Nutz eine FIFO-Warteschlange

Algorithmus

Eingabe: Graph $G = (V, E)$, Startknoten $s \in V$.
Ausgabe: Felder $d[v], pred[v]$ mit $v \in V$.

```
for all v ∈ V do begin
    if v = s then d[v] ← 0 else d[v] ← ∞;
    pred[v] ← nil;
end
Q ← new Queue;
Q.Insert(s);
while not Q.IsEmpty() do begin
    v ← Q.DeQueue();
    for all u ∈ Γ(v) do
        if d[u] = ∞ then begin
            d[u] ← d[v] + 1;
            pred[u] ← v;
            Q.Insert(u);
        end
end
```

Beispiel