35. Algorithmus der Woche

Zyklensuche

Prof. Dr. H. Schlingloff

Humboldt-Universität zu Berlin

Szenario 1

- Flugzeugabsturz im Urwald
- verschlungene Trampelpfade
- Verzweigungen, Kreuzungen, Sackgassen
- keine Orientierungs-möglichkeit (durch Sonne, Kompass oder ähnliches)
- Suche nach Ausweg

Wie kann man es vermeiden, im Kreis zu laufen?

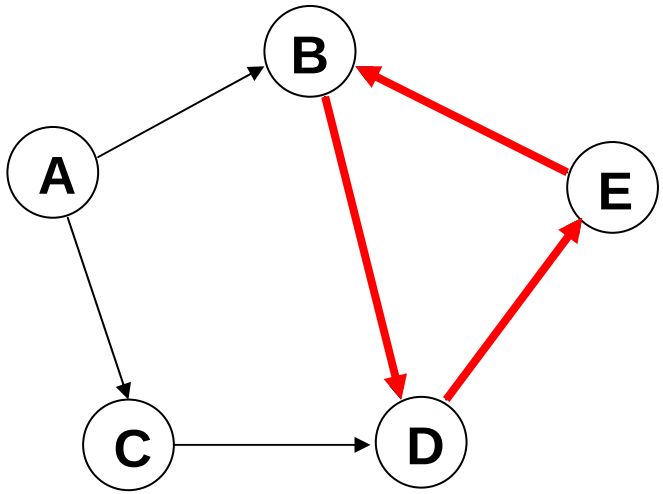
Szenario 2

- **Andy:** möchte mit Benny und Charly ins Kino
- **Benny:** braucht Hausaufgabenhilfe von Dany
- **Charly:** muss Babysitten, wartet auf Dany als Ablösung
- **Dany:** hat Eddy Heft geliehen, wartet auf Rückgabe
- **Eddy:** grübelt an Aufgaben, wartet auf Mail von Benny

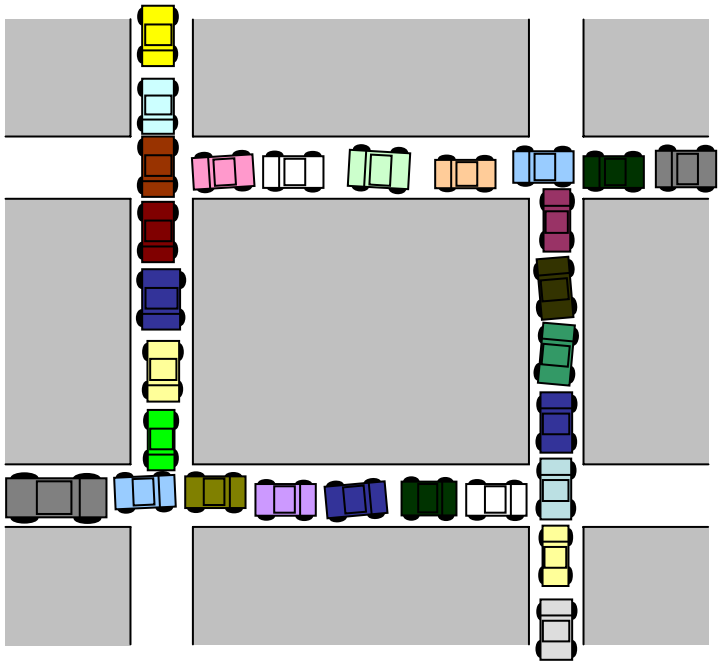
Warum kommt Andy wahrscheinlich nicht ins Kino?

Szenario 2

- **Andy:** möchte mit Benny und Charly ins Kino
- **Benny:** braucht Hausaufgabenhilfe von Dany
- **Charly:** muss Babysitten, wartet auf Dany als Ablösung
- **Dany:** hat Eddy Heft geliehen, wartet auf Rückgabe
- **Eddy:** grübelt an Aufgaben, wartet auf Mail von Benny



zyklische Wartesituation

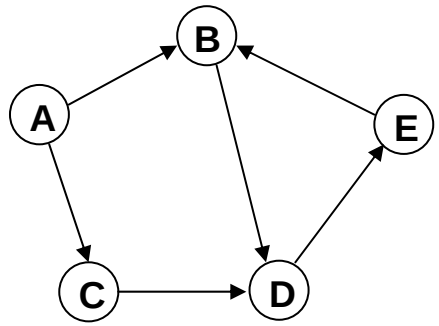


- Endlosschleifen („livelock“)
- Verklemmungen („deadlock“)

Erkennung / Vermeidung von Zyklen?

Modellierung: Graphen

- Graph G = (N, E)
  - nichtleere endliche Menge N von *Knoten (nodes)*
  - Menge E $\subseteq$ N$\times$N von *Kanten (edges)*



Beispiel: N={A,B,C,D,E}

E={(A,B), (A,C), (B,D), (C,D), (D,E), (E,B)}

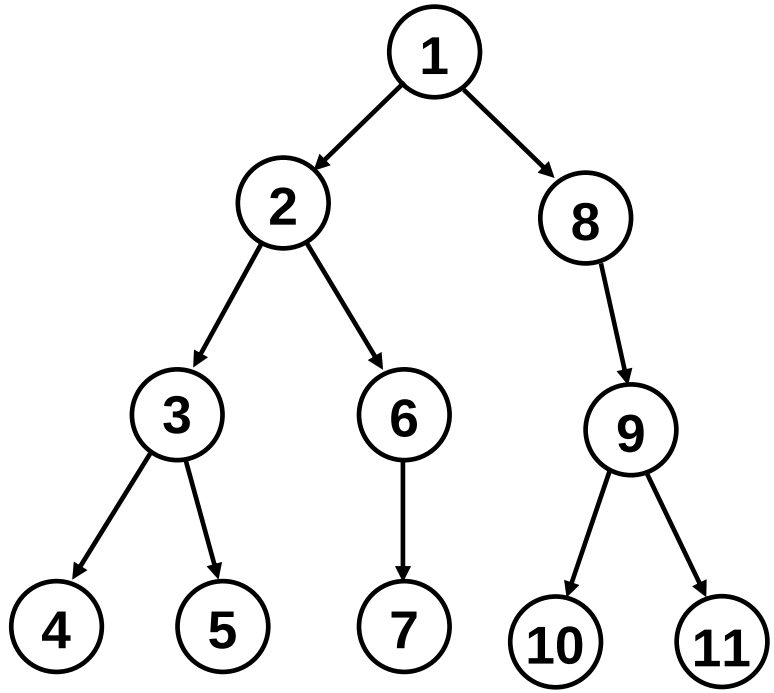
graphische Darstellung: Knoten als Kreise, Kanten als Pfeile zwischen Kreisen (von-nach)

Tiefensuche

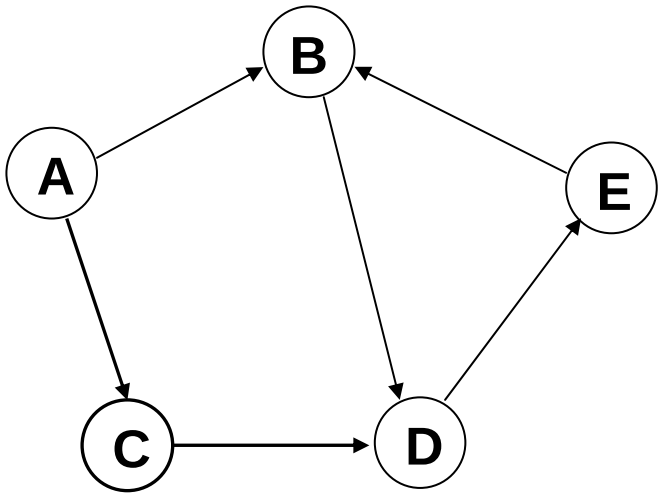
Gegeben ist ein Graph sowie ein Anfangsknoten im Graphen, gesucht ist irgendein Knoten mit einer bestimmten Eigenschaft.

- Besuche der Reihe nach alle Knoten, die über Kanten erreichbar sind
  - Also: zunächst den ersten vom Anfangsknoten erreichbaren Knoten
  - Dann: den ersten von diesem Knoten aus erreichbaren, und so weiter
  - Wenn kein Knoten mehr erreichbar ist, kehre zum letzten Punkt zurück, an dem es noch weitere Kanten gab
  - Wenn vom Anfangsknoten alle Kanten abgesucht wurden, ist der gesamte erreichbare Teil des Graphen abgesucht

Beispiel:



Algorithmus

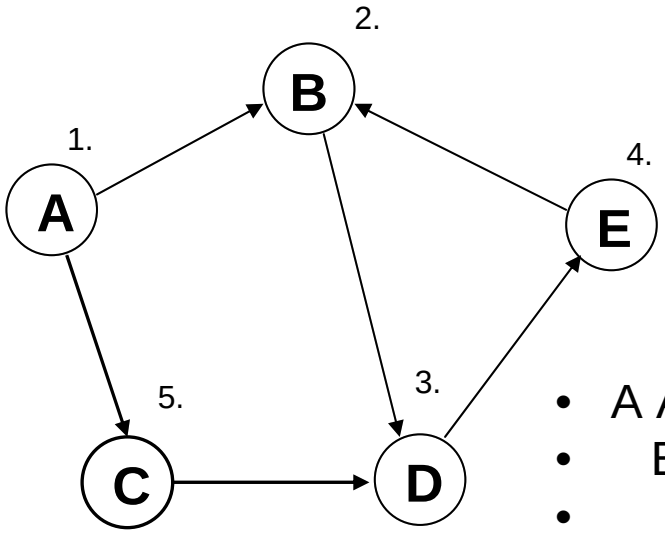


„Markieren“ kann zum Beispiel dadurch geschehen, dass ein Sternchen oder eine fortlaufende Nummer dazugeschrieben wird.

rekursiv!

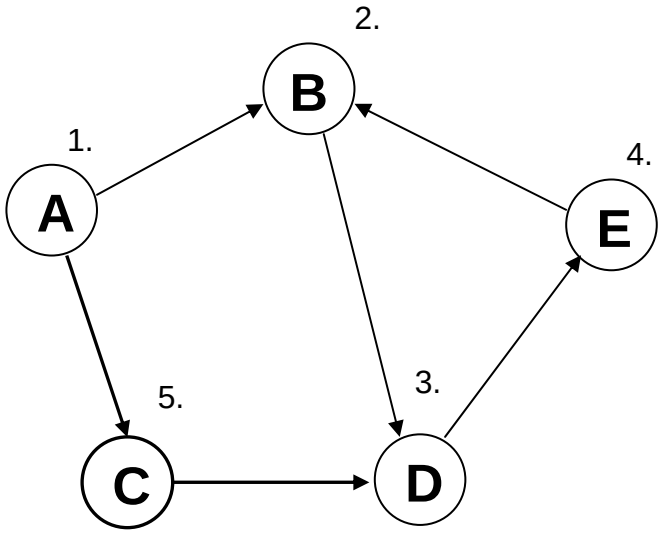
```
1 prozedur Tiefensuche (Knoten x)
2    falls Ausgang erreicht dann stopp
3    sonst falls x unmarkiert dann
4       markiere x;
5       für alle Nachfolgerknoten y von x tue Tiefensuche(y)
```

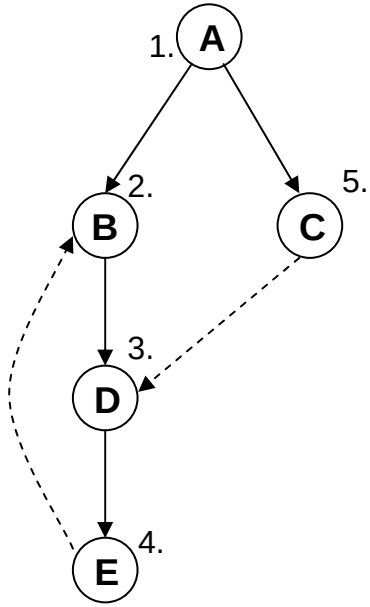
Ausführung



- A Anfangsknoten; Kanten zu B und C
- B ist erster Nachfolger von A; Kante zu D
- D ist Nachfolger von B, Kante zu E
- E ist Nachfolger von D, Kante zu B
- B wurde schon markiert, Rückkehr zu E
- E hat keine weiteren Nachfolger, Rückkehr zu D
- D hat keine weiteren Nachfolger, Rückkehr zu B
- B hat keine weiteren Nachfolger, Rückkehr zu A
- C ist zweiter Nachfolger von A, Kante zu D
- D wurde schon markiert, Rückkehr zu C
- C hat keine weiteren Nachfolger, Rückkehr zu A
- A hat keine weiteren Nachfolger, Stopp

Tiefensuch-Baum

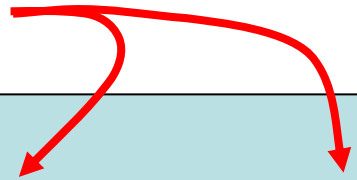




Vorwärtskanten, z.B. von A nach C
Querkanten, z.B. von C nach D
Rückwärtskanten, z.B. von E nach B

Zyklensuche

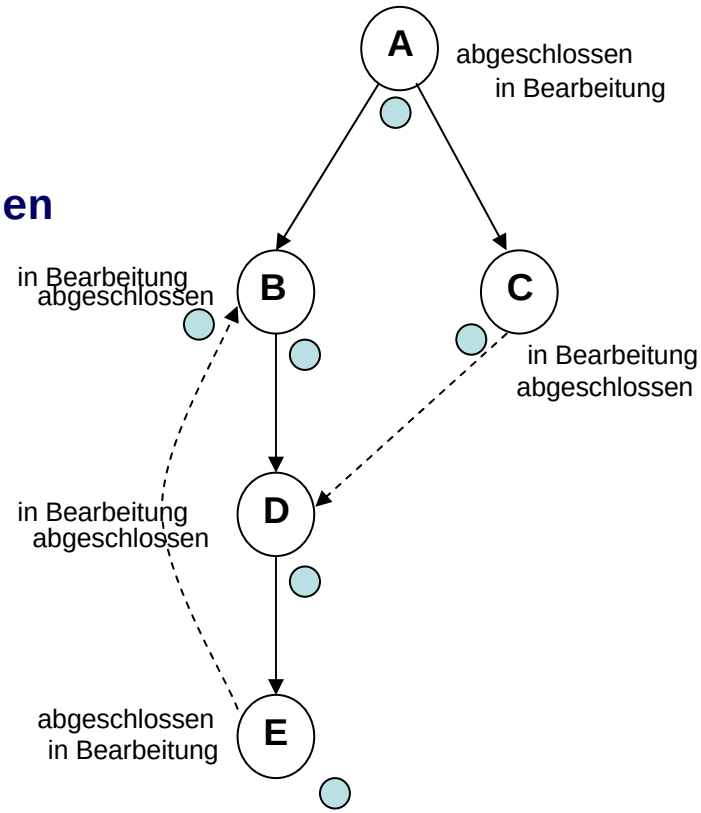
- Rückwärtskanten verweisen auf einen Knoten, der noch „in Bearbeitung“ ist
- Erweiterte Markierung
  - „noch nicht begonnen“ (oder „unmarkiert“, am Anfang alle Knoten)
  - „in Bearbeitung“
  - „abgeschlossen“
- Wenn eine Kante zu einem Knoten „in Bearbeitung“ führt, wurde ein Zyklus gefunden



```
1 prozedur Zyklensuche (Knoten x)
2    falls Markierung(x) = „in Bearbeitung“ dann Zyklus gefunden
3    sonst falls Markierung(x) = „noch nicht begonnen“ dann
4       Markierung(x) := „in Bearbeitung“;
5       für alle Nachfolgerknoten y von x tue Zyklensuche(y);
6       Markierung(x) := „abgeschlossen“;
```

Aufrufbeispiel

```
Zyklensuche (A) // A noch nicht begonnen
  A in Bearbeitung
  Zyklensuche (B) // B noch nicht begonnen
    B in Bearbeitung
    Zyklensuche (D) // D noch nicht begonnen
      D in Bearbeitung
      Zyklensuche (E) // E noch nicht begonnen
        E in Bearbeitung
        Zyklensuche (B) // B in Bearbeitung
          Zyklus gefunden!
        E abgeschlossen
      D abgeschlossen
    B abgeschlossen
  Zyklensuche (C) // C noch nicht begonnen
    C in Bearbeitung
    Zyklensuche (D) // D abgeschlossen
    C abgeschlossen
  A abgeschlossen
```



Fertig!

Zyklen finden

Was tun, wenn ein Zyklus entdeckt wird?

Um alle Knoten auf dem Zyklus auszugeben, muss man sich „merken“, welche gerade in Bearbeitung sind

- Zusätzliche Datenstruktur zum Speichern der Knoten „in Bearbeitung“ nötig
- Geeignet: Liste bzw. Stapel von Knoten

```
1 prozedur Zyklenfinden (Knoten x)
2   falls Markierung(x) = „in Bearbeitung“ dann
3     Zyklus gefunden;
      alle Knoten auf dem aktuellen Pfad ab x liegen auf dem Zyklus
4   sonst falls Markierung(x) = „noch nicht begonnen“ dann
5     Markierung(x) := „in Bearbeitung“;
6     Verlängere den aktuellen Suchpfad um x;
7     für alle Nachfolgerknoten y von x tue Zyklenfinden(y);
8     Markierung(x) := „abgeschlossen“;
9     Entferne x (das letzte Element) aus dem aktuellen Pfad;
```

Ausgabe des Ergebnisses

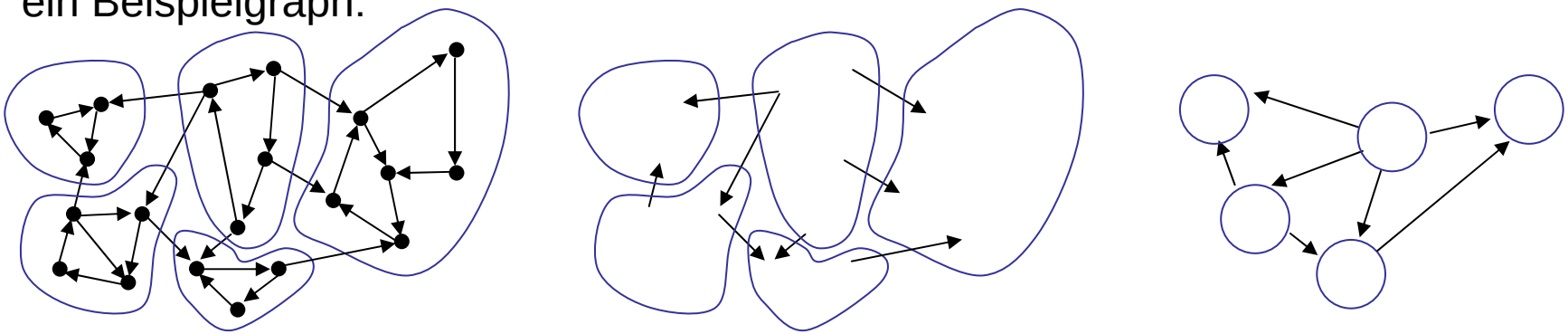
- aktueller Pfad: A→B→D→E
- Kante von E führt zu B

} B, D, E gehören zum Zyklus

- **Programmieraufgabe:** für ein gegebene Liste L von Knoten und eine Knotenbezeichnung k alle Elemente bis zum ersten Vorkommnis von k rückwärts ausgeben.
- Beispiel: Eingabe „ZYKLENSUCH“ und „E“, Ausgabe „HCUSNE“

Ausblick: Starke Zusammenhangskomponenten

ein Beispielgraph:



- Starke Zusammenhangskomponente: Verbundene Zyklen (jeder Knoten ist von jedem erreichbar)
- Alle Knoten in einer Zusammenhangskomponente sind sozusagen „äquivalent“
- Quotientengraph: ersetze jede starke Zusammenhangskomponente durch einen eigenen Knoten
- Tarjan‘s Algorithmus zur Berechnung des Quotientengraphen ist eine kleine Erweiterung von **`prozedur`** `Zyklenfinden`