Wege, Pfade und Kreise

- Def.: Ein Weg ist eine Folge von Knoten $(v_1, v_2, ..., v_k)$, so dass $\{v_i,v_{i+1}\}\in E$ für alle $1 \leq i<k$.
  - Der Abstand zwischen zwei Knoten ist die Länge des kürzesten Weges zwischen diesen Knoten.
  - Ein Weg heißt einfach, wenn $\{v_i,v_{i+1}\}\neq\{v_j,v_{j+1}\}$ für alle $1 \leq i<j<k$
  - Analog für gerichtete Graphen
- Def.: Ein Pfad ist ein Weg $(v_1, v_2, ..., v_k)$, so dass $v_i\neq v_j$ für alle $1 \leq i<j\leq k$.
- Def.: Ein Kreis ist ein einfacher Weg $(v_1, v_2, ..., v_k)$, so dass $v_i\neq v_j$ für alle $1<i<j<k$ und $v_1=v_k$.

Zusammenhang

- Def.: Ein ungerichteter Graph ist zusammenhängend, wenn es für alle $u,v \in V$ einen Pfad von u nach v gibt.
- Def.: Ein gerichteter Graph ist stark zusammenhängend, wenn es für alle $u,v \in V$ einen Pfad von u nach v und einen Pfad von v nach u gibt.
  - Ein gerichteter Graph ist schwach zusammenhängend wenn sein ungerichteter Graph zusammenhängend ist.
- Die (Zusammenhangs-)Komponenten eines Graphen sind seine maximalen zusammenhängenden Teilgraphen.

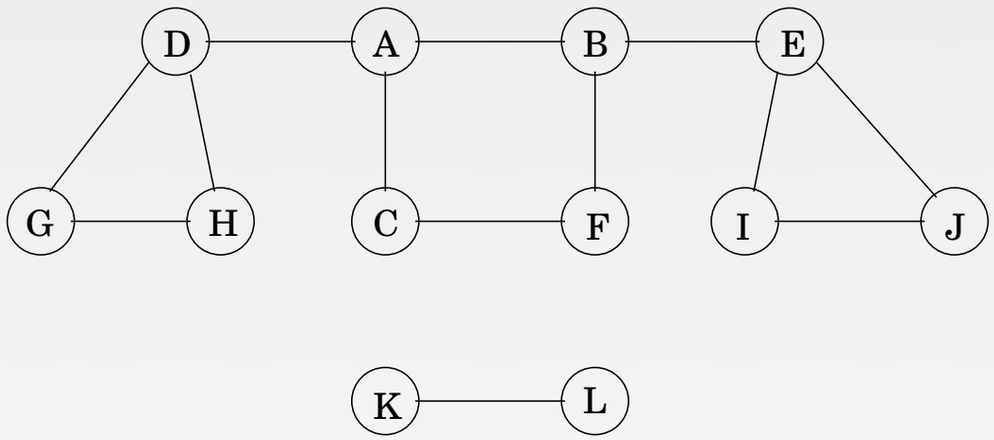
Bäume

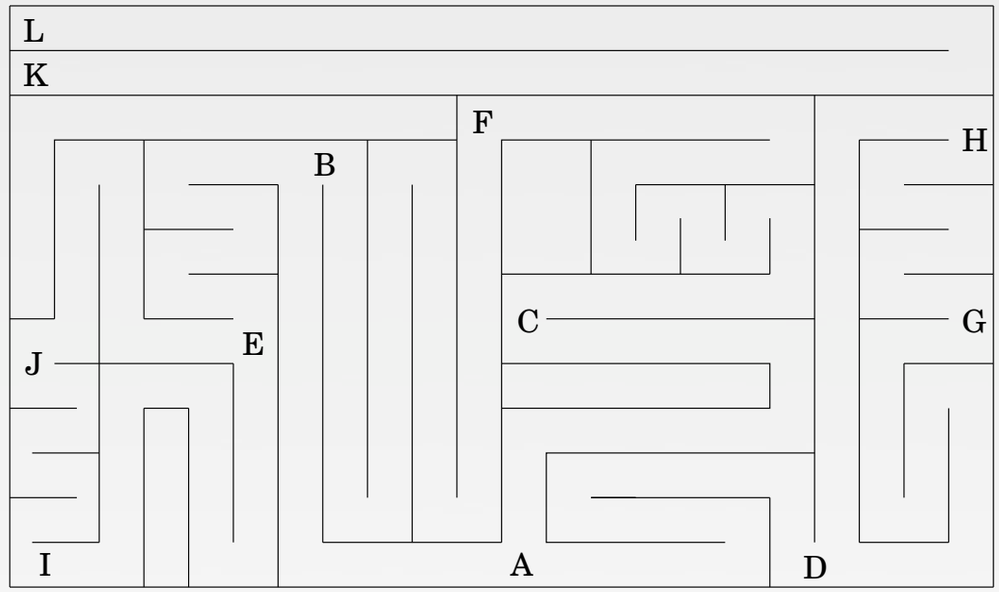
- Def.: Ein Baum ist ein zusammenhängender kreisfreier Graph.
  - Ein Knoten eines Baums v heißt Blatt, wenn deg(v)=1.
  - Ein Wald ist ein Graph, dessen Komponenten Bäume sind.

- Satz: In jedem Baum gilt |E|=|V|-1.
  - Beweis: Widerspruch (durch Annahme es gäbe ein kleinstes Gegenbeispiel)
    - Jeder Baum mit mehr als zwei Knoten enthält mindestens zwei Blätter
    - Durch Entfernen eines Blatts in einem Baum mit mehr als zwei Knoten erhält man wieder einen Baum

- Wenn man eine Kante aus einem Baum entfernt, ist der resultierende Graph nicht zusammenhängend.
  - Bäume sind "gerade noch" zusammenhängend.

Graphtraversierung



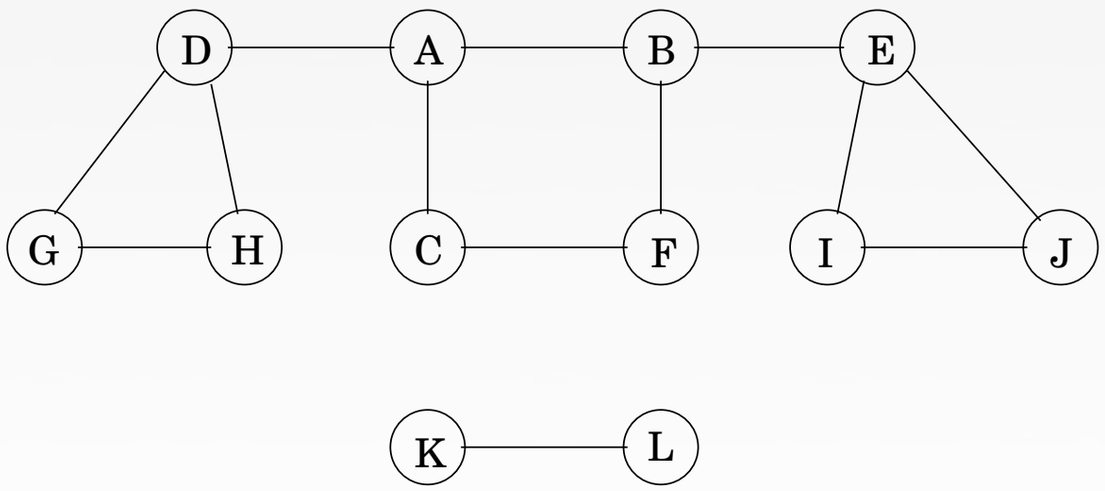


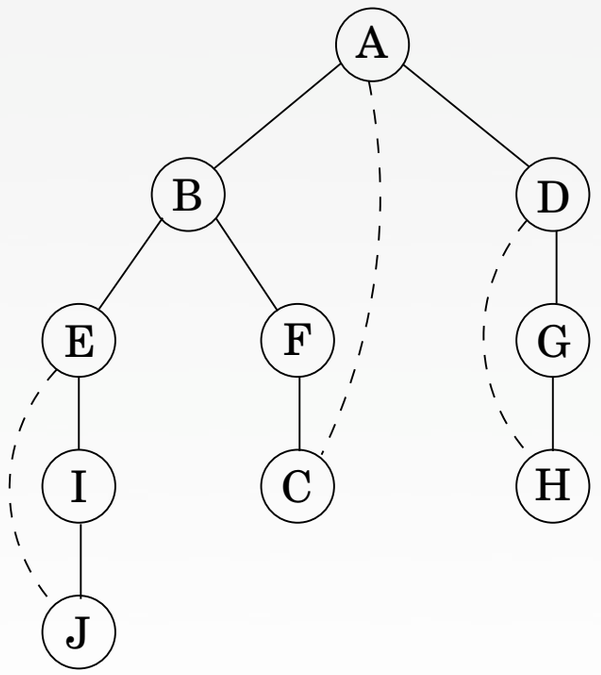
[Dasgupta, Papadimitriou, & Vazirani, 2006]

- Allgemeiner Traversierungsalgorithmus (Startknoten s)
  - Z={s}
  - While ∃u∈Z, v∈V\Z: {u,v}∈E
  - Z=Z∪{v}
  - Endwhile

Tiefensuche

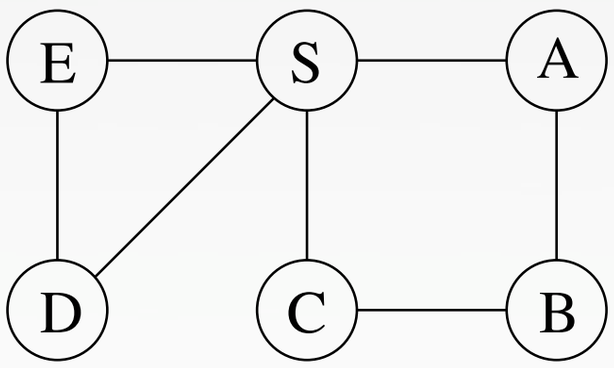
- Exploriere die Nachbarschaft der jüngsten Knoten in Z
  - Implementierung mit Hilfe eines Stapelspeichers (*stack*): LIFO
    - Push: nicht betrachtete Knoten in der Nachbarschaft des obersten Elements
    - Pop: falls alle Knoten in der Nachbarschaft des obersten Elements bereits betrachtet wurden
  - Lineare Laufzeit: O(|V|+|E|) bei Verwendung einer Adjazenzliste

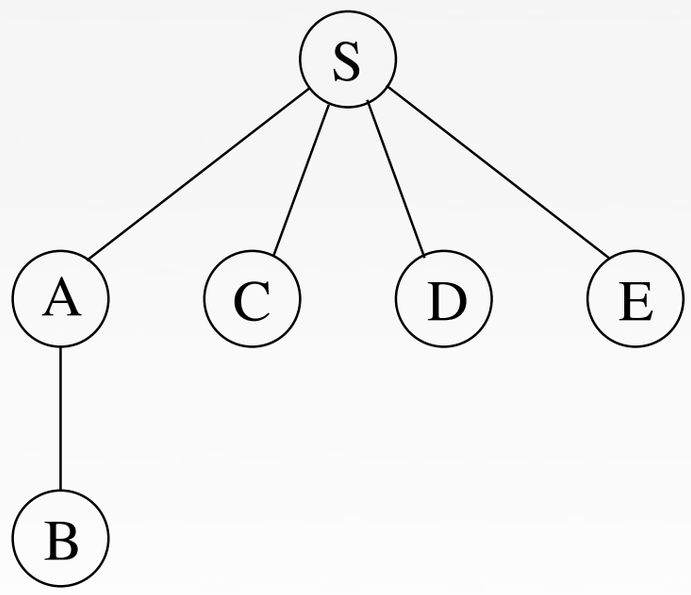




Breitensuche

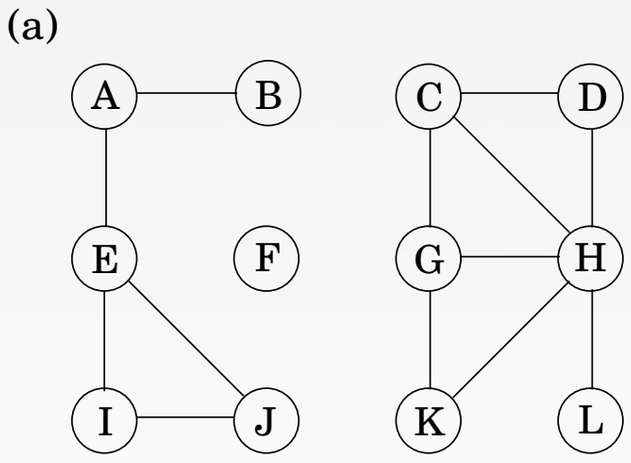
- Exploriere die Nachbarschaft der ältesten Knoten in Z
  - Implementierung mit Hilfe einer Warteschlange (*queue*): FIFO
  - Vordringen in Ebenen
    - Die k. Ebene besteht aus allen Knoten, deren kürzester Abstand zu u k beträgt.
    - Spannbaum aus kürzesten Wegen
  - Lineare Laufzeit

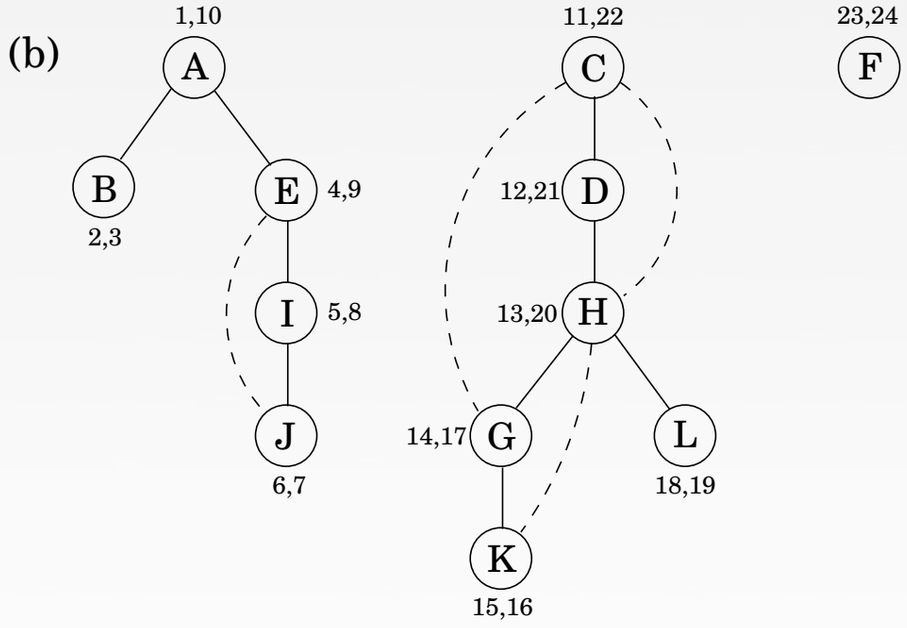




Traversierung: Anwendungen

- Finden aller von s erreichbaren Knoten
- Finden aller Zusammenhangskomponenten
  - Knotenbezeichnung (*pre*,*post*)
    - *pre*: push-Zeitpunkt, *post*: pop-Zeipunkt





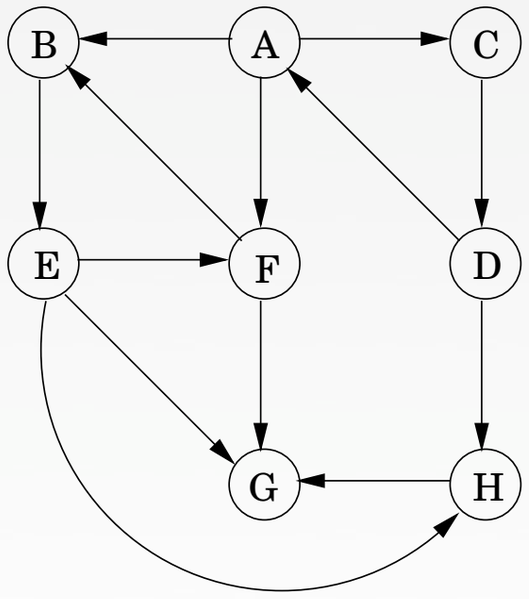
- Test auf Kreisfreiheit
  - G enthält einen Kreis ⇔ Traversierung liefert eine Rückkante

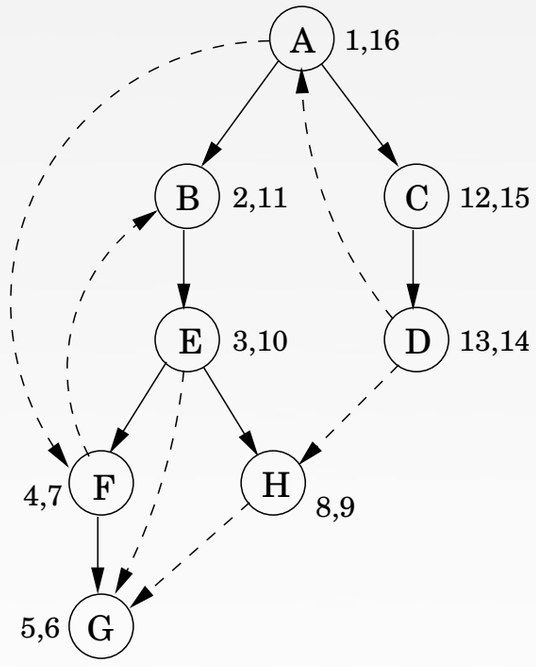
Breitensuche: Anwendungen

- Finden des kürzesten Weges zwischen zwei Knoten
- Test auf Bipartitheit
  - Def.: Ein Graph (V,E) ist bipartit, wenn es zwei disjunkte Mengen V_1 und V_2 gibt, so dass $V=V_1 \cup V_2$ und $e \cap V_1 \neq \emptyset$ und $e \cap V_2 \neq \emptyset$ für alle $e \in E$.
  - G ist bipartit $\Leftrightarrow$ G enthält keinen Kreis ungerader Länge
    - von links nach rechts: indirekter Beweis durch 2-Färbung der Knoten (adjazente Knoten dürfen nicht dieselbe Farbe haben)
    - von rechts nach links: konstruktiv durch 2-Färbung der Knoten mit Hilfe einer Breitensuche

Gerichtete Graphen (1)

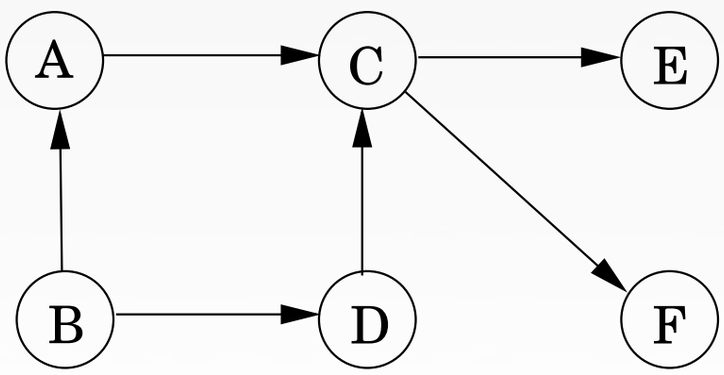
- Terminologie
  - Ein Knoten v heißt Quelle, wenn sein Eingangsgrad indeg(v)=0.
  - Ein Knoten v heißt Senke, wenn sein Ausgangsgrad outdeg(v)=0.
- Finden des kürzesten Weges zwischen zwei Knoten
  - Breitensuche
- Test auf Kreisfreiheit
  - Breiten- oder Tiefensuche





Gerichtete Graphen (2)

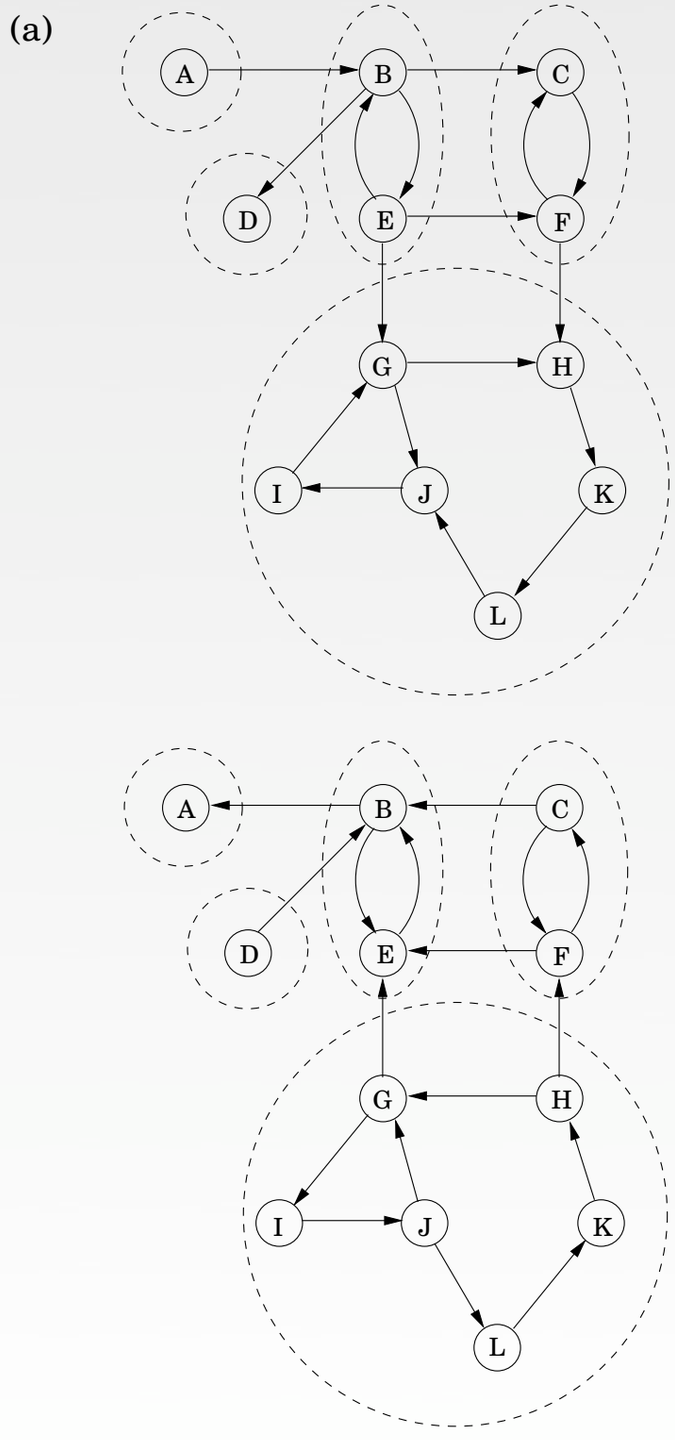
- Topologische Sortierung (Linearisierung)
  - Def: Eine topologische Sortierung eines gerichteten Graphen G=(V,A) ist eine bijektive Abbildung f:V→{1,2,...,|V|}, so dass f(u)<f(v) für alle (u,v)∈A gilt.
  - G besitzt eine topologische Sortierung ⇔ G ist kreisfrei
    - von links nach rechts: indirekter Beweis
    - von rechts nach links: konstruktiv mit Hilfe einer Tiefensuche, topologische Sortierung in umgekehrter *post*-Reihenfolge
      - Jede Kante im Tiefensuchebaum führt zu einem Knoten mit geringerem *post*-Index

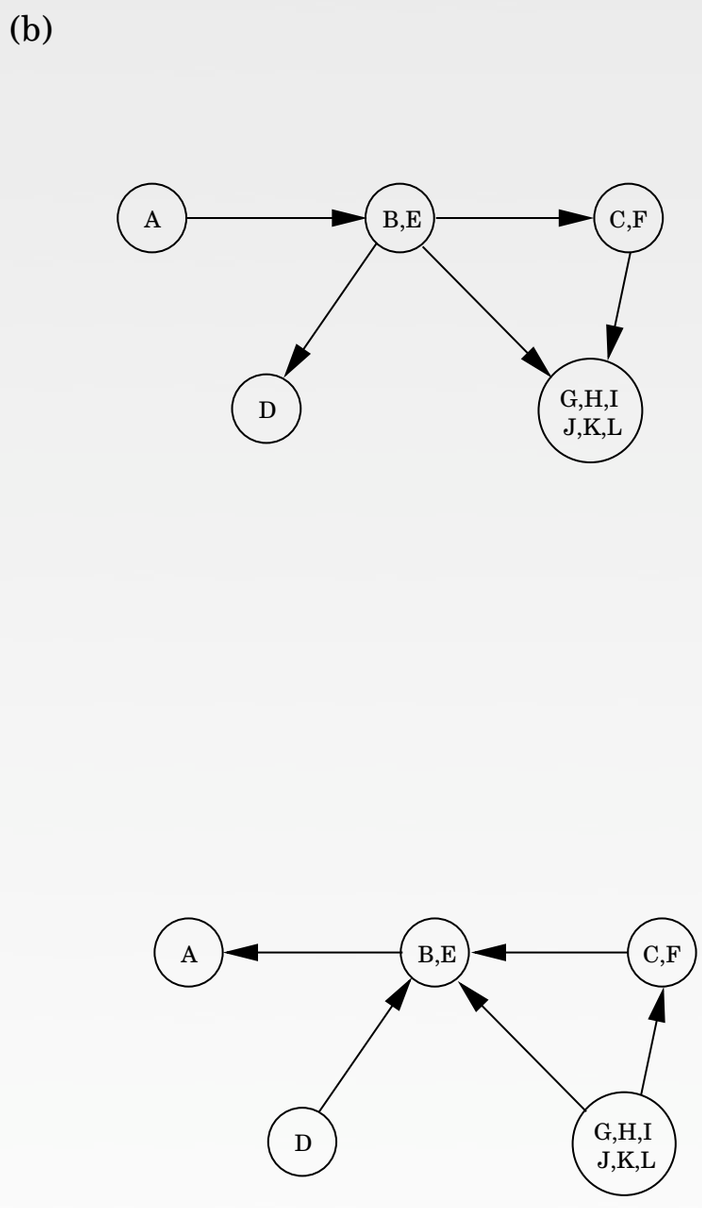


Gerichtete Graphen (3)

- Test auf starken Zusammenhang
  - Def.: Den transponierten Graph G^T eines gerichteten Graphen G erhält man durch Transponierung der Adjazenzmatrix.
    - Umkehrung aller Kanten
  - Breiten- oder Tiefensuche ausgehend von beliebigem Knoten s in G und G^T
    - G ist stark zusammenhängend ⇔ Beide Traversierungen liefern alle Knoten
- Finden aller starken Zusammenhangskomponenten
  - Die starken Zusammenhangskomponenten eines Graphen G bilden einen kreisfreien gerichteten Graphen
  - Tiefensuche ausgehend vom Knoten einer Senkenkomponente liefert alle Knoten dieser Komponente
  - Ein solcher Knoten hat maximalen *post*-Index nach Tiefensuche in G^T
  - Algorithmus von Kosaraju (1978)

Algorithmus von Kosaraju





- Tiefensuche in G^T einschl. *post*-Markierung
- While $V \neq \emptyset$
- Tiefensuche in G ausgehend vom Knoten mit höchstem *post*-Index
- Resultierende Knotenmenge stellt eine Komponente dar und kann entfernt werden
- Endwhile

Zusammenfassung

- Folgende Probleme sind sowohl in ungerichteten als auch in gerichteten Graphen in linearer Zeit lösbar (O(|V|+|E|) bei Verwendung von Adjazenzlisten):
  - Testen auf Kreisfreiheit
  - Testen auf Bipartitheit
  - Finden aller Zusammenhangskomponenten
  - Finden kürzester Wege
  - Finden einer topologischen Sortierung eines gerichteten Graphen
- Erweiterung von Breitensuche für gewichtete Graphen
  - Dijkstras Algorithmus (für positive Kantengewichte)
  - Bellman & Ford Algorithmus