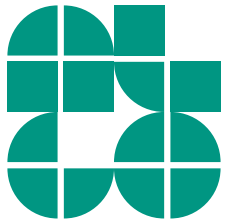
Algorithmen zur Visualisierung von Graphen

Übung 4: Lagenlayouts

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Tamara Mchedlidze · **Benjamin Niedermann**

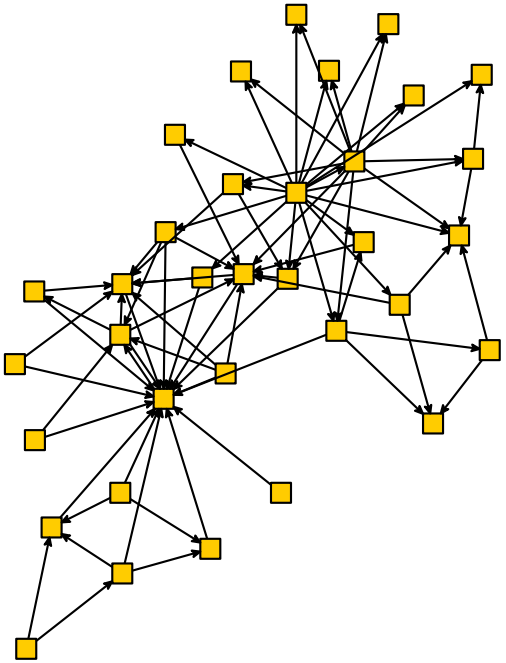
11.01.2017

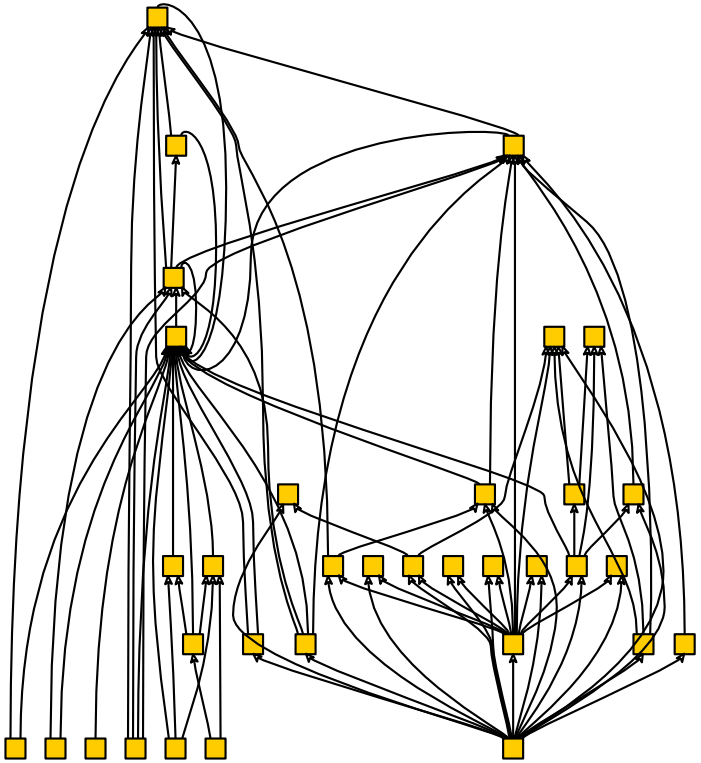


Layered Layout

Given: directed graph $D = (V, A)$

Find: drawing of D that emphasized the hierarchy





Layered Layout

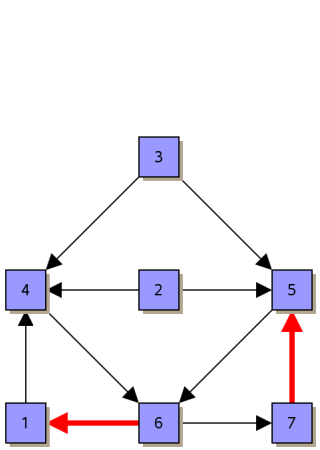
Given: directed graph $D = (V, A)$

Find: drawing of D that emphasized the hierarchy

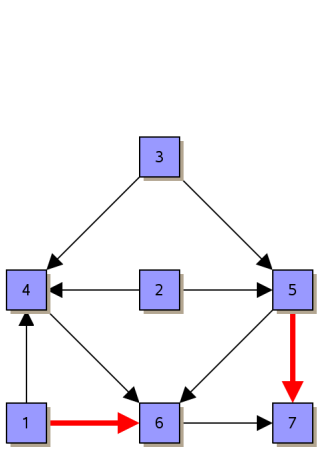
Criteria:

- many edges pointing to the same direction
- edges preferably straght and short
- position nodes on (few) horizontal lines
- preferably few edge crossings
- nodes distributed evenly

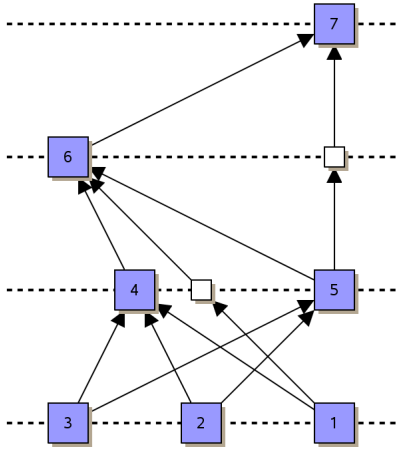
Sugiyama Framework (Sugiyama, Tagawa, Toda 1981)



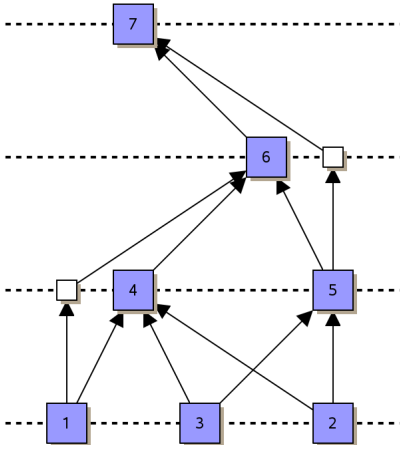
given



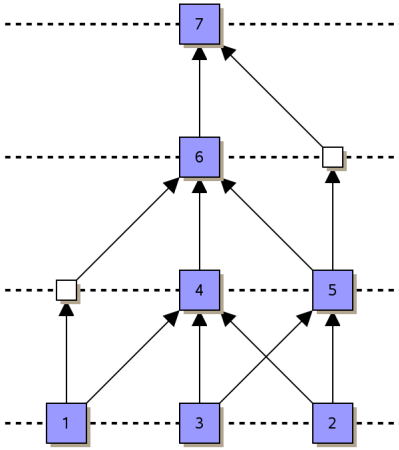
resolve cycles



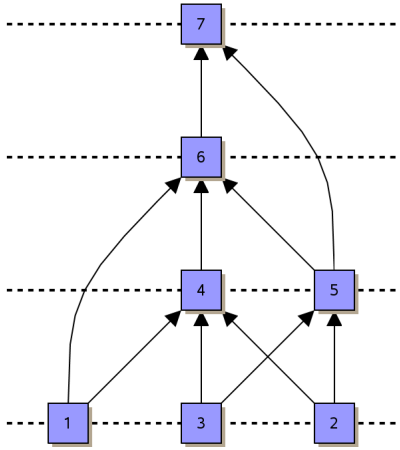
layer assignement



crossing minimization



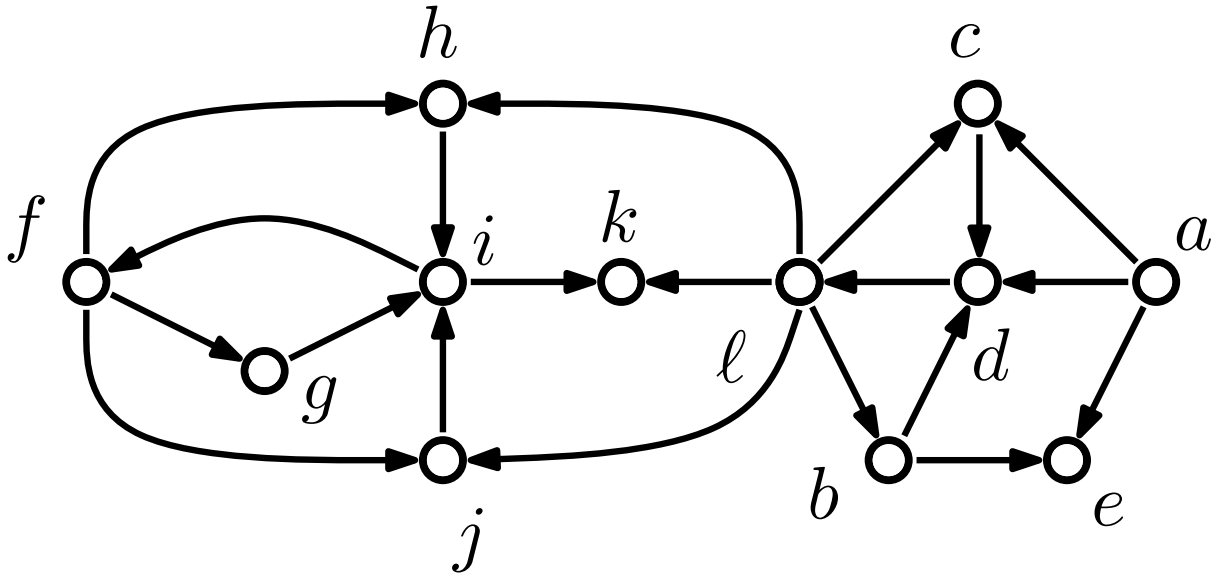
node positioning



edge drawing

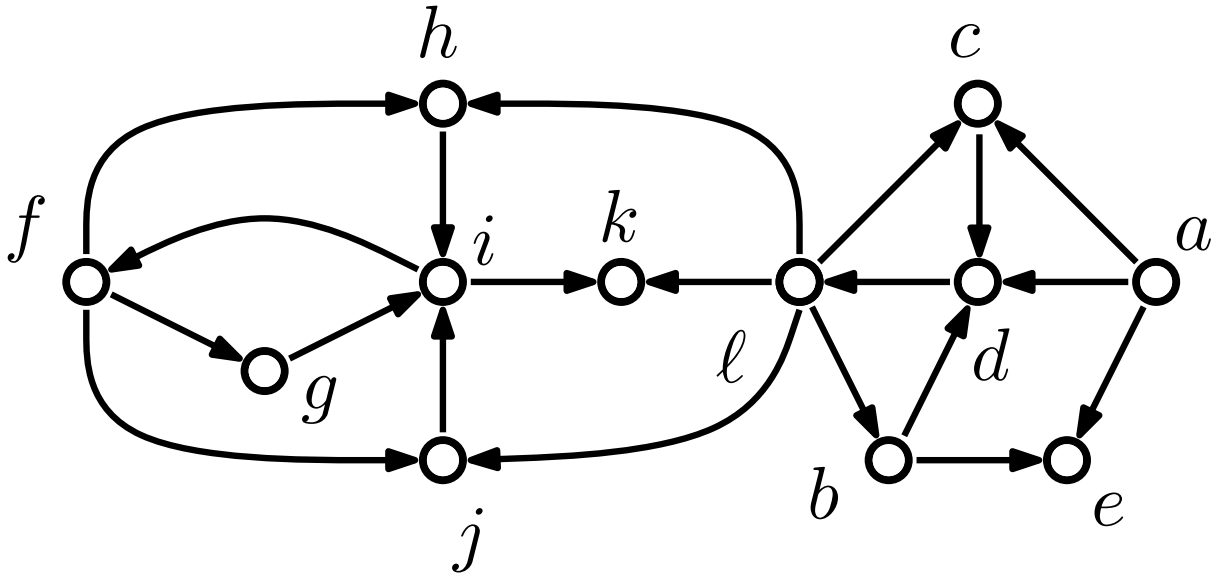
Aufgabe 2 – Lagenlayout

Führen Sie den in der Vorlesung vorgestellten Algorithmus zur Generierung von Lagenlayouts schrittweise (manuell und optimal) für den untenstehenden Graphen aus. Entfernen Sie dazu gerichtete Kreise indem Sie für eine möglichst kleine Menge an Kanten die Richtung umkehren, finden Sie eine Lagenzuordnung minimaler Höhe bei maximaler Breite 4 und ordnen sie die Knoten innerhalb der Lagen so an, dass die Anzahl an Kreuzungen minimal ist.



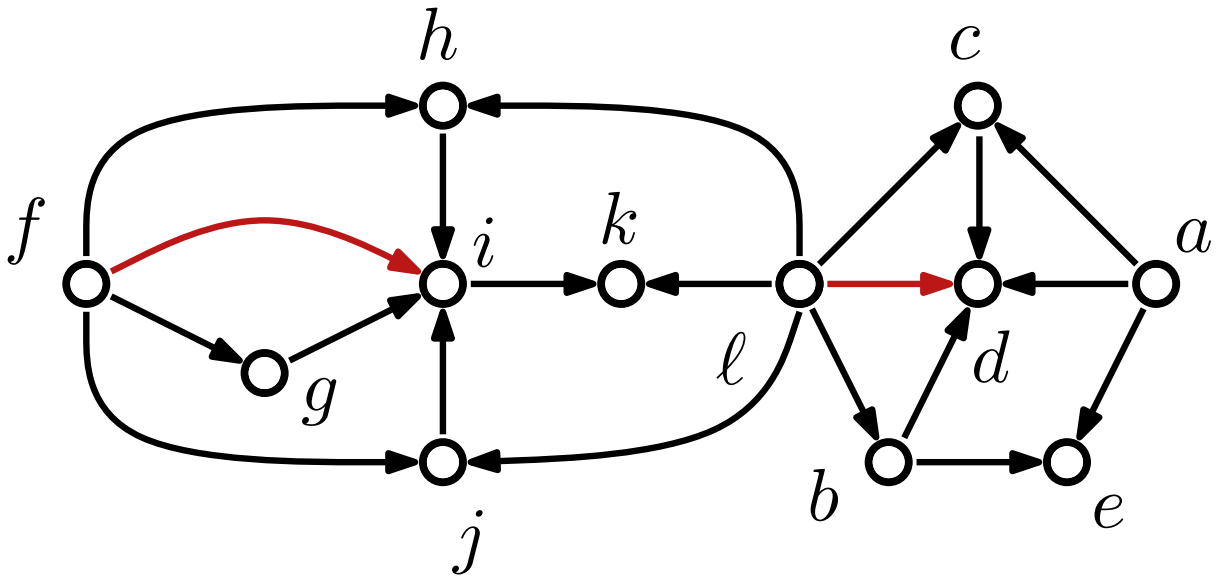
Lagenlayout – 1. FAS

Finde $E' \subseteq E$ sodass $G - E'$ azyklisch und $|E'|$ minimal (FAS)



Lagenlayout – 1. FAS

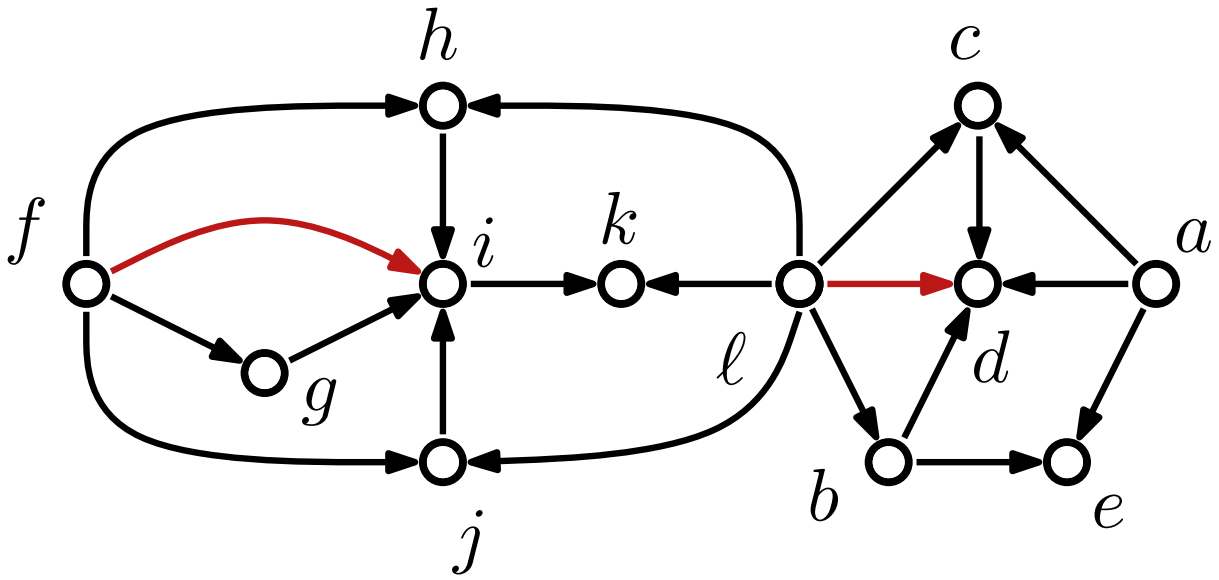
Finde $E' \subseteq E$ sodass $G - E'$ azyklisch und $|E'|$ minimal (FAS)



Lagenlayout – 1. FAS

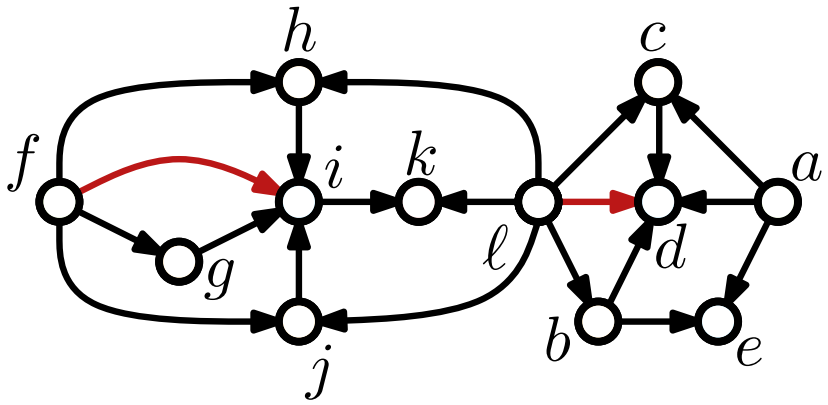
Finde $E' \subseteq E$ sodass $G - E'$ azyklisch und $|E'|$ minimal (FAS)

Kehre die Orientierung der Kanten in E' um



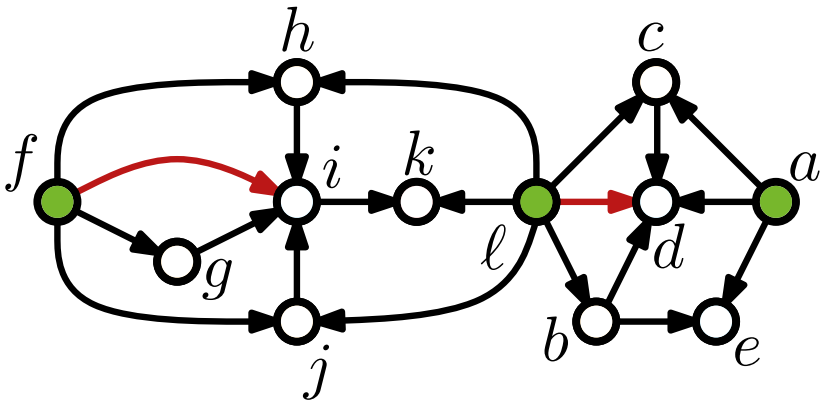
Lagenlayout – 2. Lagenzuordnung

Lagenzuordnung min. Höhe: iteratives Löschen von Quellen



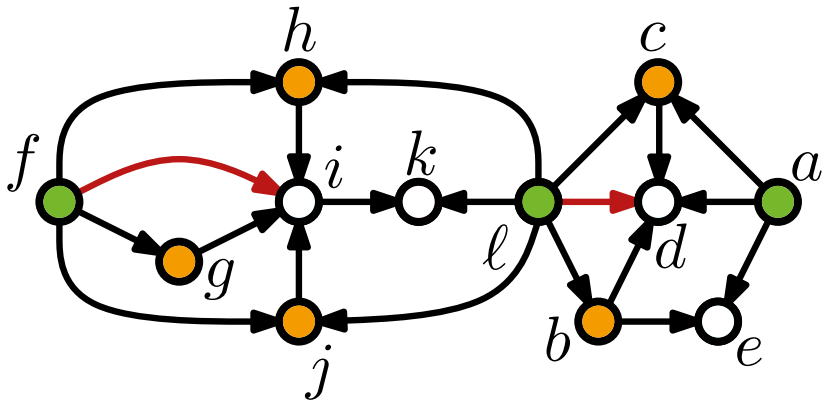
Lagenlayout – 2. Lagenzuordnung

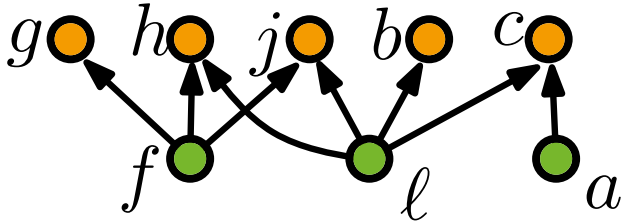
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen



Lagenlayout – 2. Lagenzuordnung

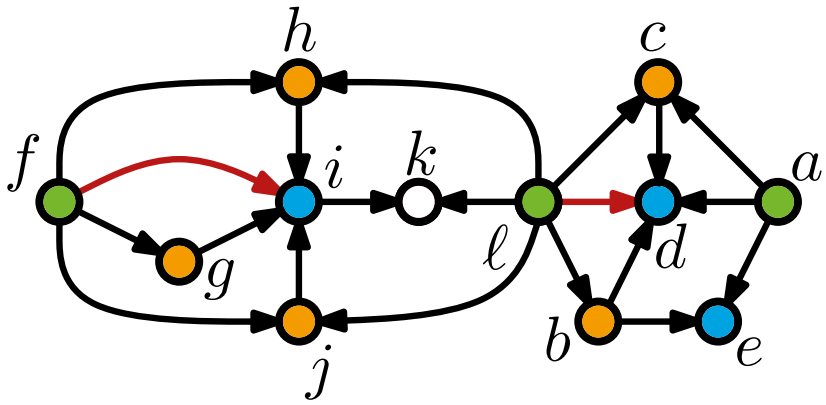
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen

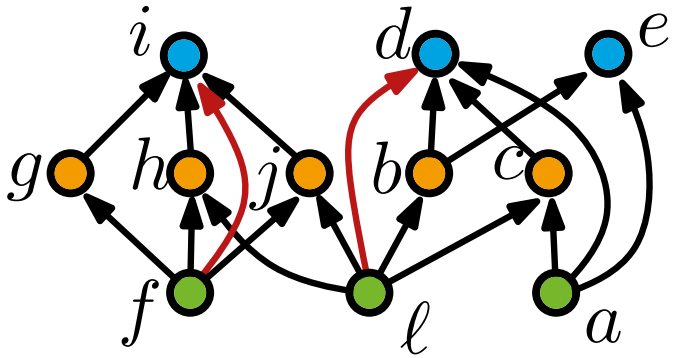




Lagenlayout – 2. Lagenzuordnung

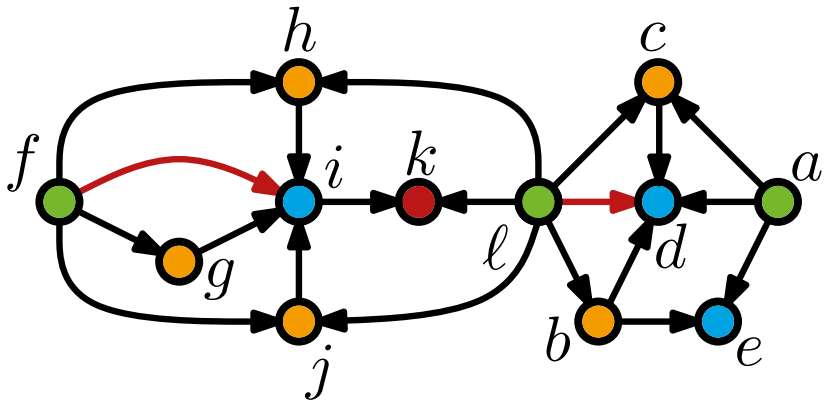
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen

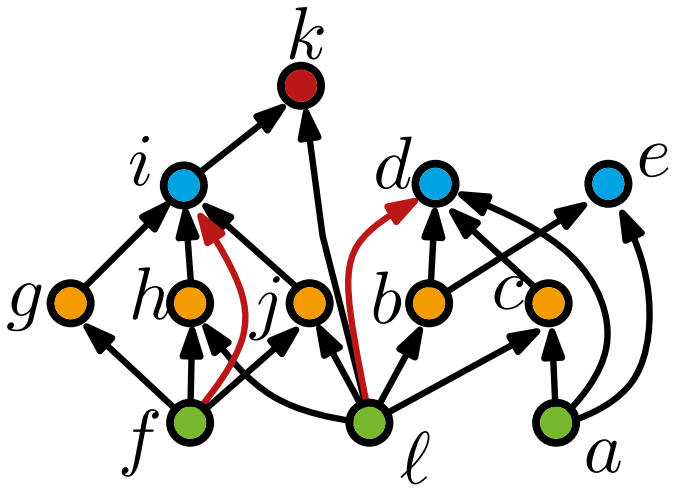




Lagenlayout – 2. Lagenzuordnung

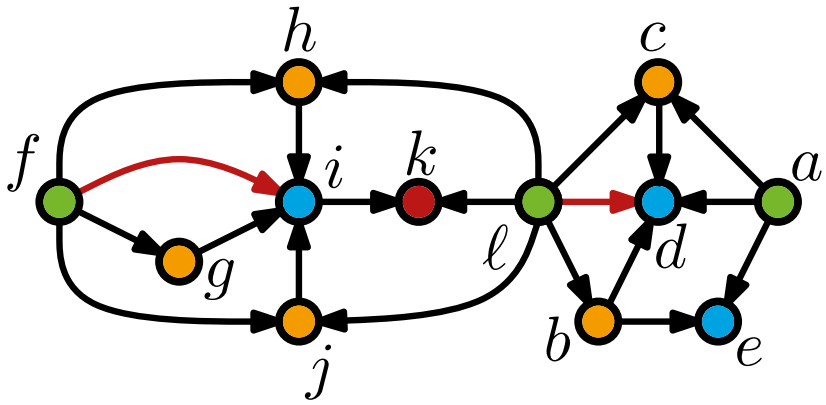
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen

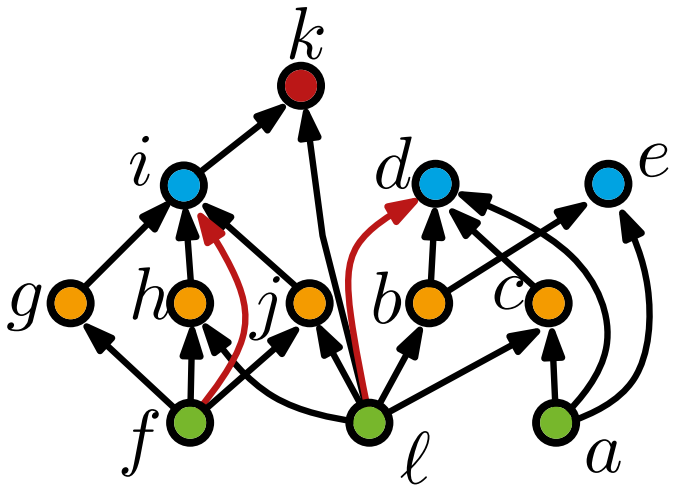




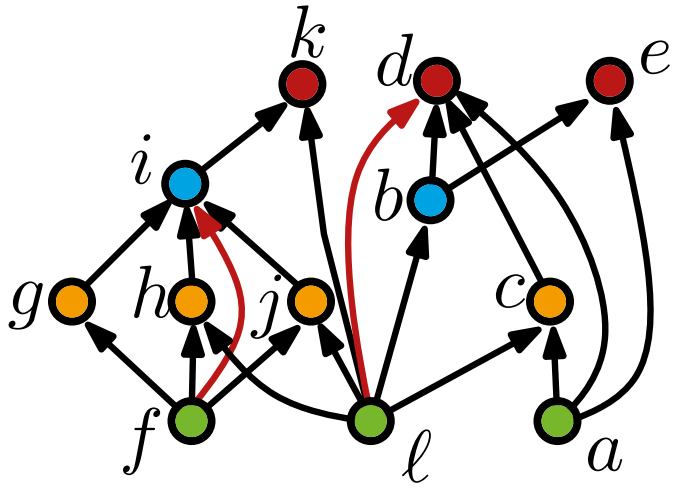
Lagenlayout – 2. Lagenzuordnung

Lagenzuordnung min. Höhe: iteratives Löschen von Quellen



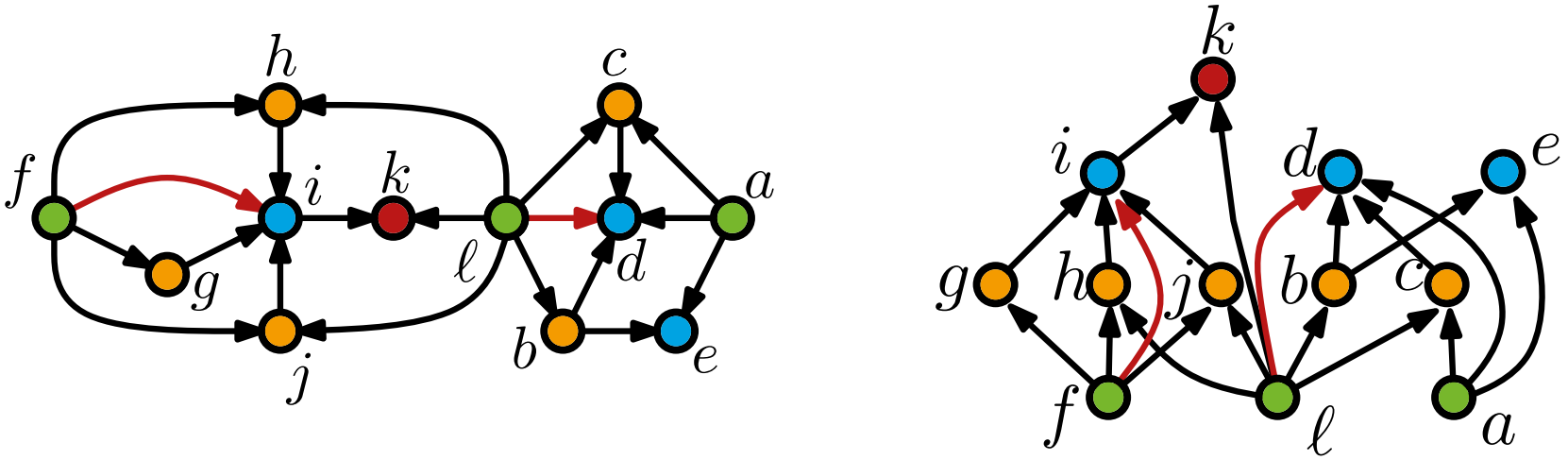


Min Höhe bei auf 4 beschränkter Breite: genaues hinschauen

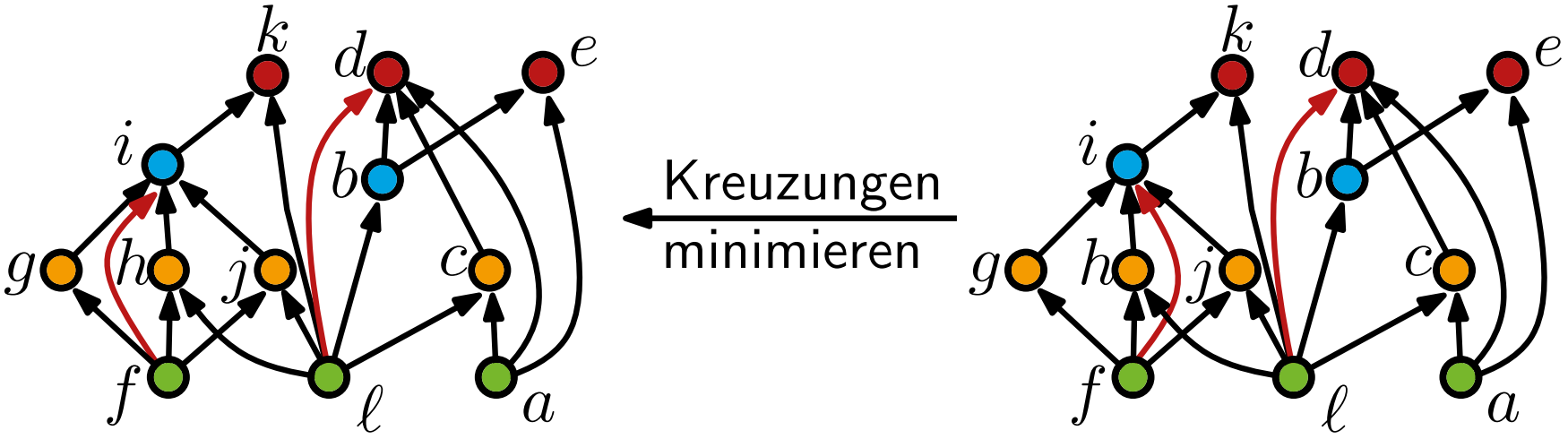


Lagenlayout – 2. Lagenzuordnung

Lagenzuordnung min. Höhe: iteratives Löschen von Quellen



Min Höhe bei auf 4 beschränkter Breite: genaues hinschauen



Aufgabe 1 – Feedback Arc Set

Minimum Feedback Arc Set: Sei $D = (V, A)$ ein gerichteter Graph. Bestimme eine Menge $A_f \subset A$ minimaler Kardinalität, so dass $D_f = (V, A \setminus A_f)$ azyklisch ist.

Minimum Feedback Set: Sei $D = (V, A)$ ein gerichteter Graph. Bestimme eine Menge $A_r \subset A$ minimaler Kardinalität, so dass $D_r = (V, A \setminus A_r \cup \mathrm{rev}(A_r))$ azyklisch ist. Dabei bezeichne $\mathrm{rev}(A)$ die Kantenmenge, die man erhält wenn die Richtung jeder Kante der Menge A invertiert wird.

Für minimale Mengen gilt die Äquivalenz: Zeigen Sie, dass die Lösungsmengen der beiden Probleme identisch sind.

Aufgabe 4 – Kreuzungen bei Lagenlayouts

Zeigen Sie, dass die Baryzenter-Heuristik zur einseitigen Kreuzungsreduktion die optimale Lösung liefert, falls diese keine Kreuzung enthält.

Aufgabe 4 – Kreuzungen bei Lagenlayouts

Zeigen Sie, dass die Baryzenter-Heuristik zur einseitigen Kreuzungsreduktion die optimale Lösung liefert, falls diese keine Kreuzung enthält.

Aufgabe 4 – Kreuzungen bei Lagenlayouts

Zeigen Sie, dass die Baryzenter-Heuristik zur einseitigen Kreuzungsreduktion die optimale Lösung liefert, falls diese keine Kreuzung enthält.

Lösungsskizze

- kreuzungsfrei gdw. alle Nachbarn auf fester Lage konsekutiv sind
- Nachbarn bilden im Inneren disjunkte Intervalle, also getrennte Schwerpunkte
- Schwerpunkte nur identisch bei Grad-1 Knoten $\rightarrow$ Reihenfolge beliebig

Aufgabe 3 – Fehlstände Zählen

(a) Sei $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ eine Permutation. Ein Paar (i, j) mit $1 \leq i < j \leq n$ heißt Inversion, wenn $\pi(i) > \pi(j)$. Entwerfen Sie einen Algorithmus, der die Anzahl der Inversionen einer Permutation von n Elementen in $O(n \log n)$ Zeit berechnet.
Hinweis: Denken Sie an Sortieralgorithmen wie Mergesort.

Aufgabe 3 – Fehlstände Zählen

(a) Sei $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ eine Permutation. Ein Paar (i, j) mit $1 \leq i < j \leq n$ heißt Inversion, wenn $\pi(i) > \pi(j)$. Entwerfen Sie einen Algorithmus, der die Anzahl der Inversionen einer Permutation von n Elementen in $O(n \log n)$ Zeit berechnet.
Hinweis: Denken Sie an Sortieralgorithmen wie Mergesort.

(b) Gegeben sei ein einfacher, bipartiter Graph $G = (V, E)$, dessen Knoten gemäß der Bipartition auf zwei parallele Geraden verteilt sind. Die Knoten seien disjunkt und die Kanten geradlinig gezeichnet. Entwerfen Sie einen Algorithmus, der die Anzahl der Kreuzungen in $O(|E| \log |V|)$ Zeit bestimmt. Begründen Sie, warum es nicht möglich ist, in dieser worst-case-Laufzeit alle Kreuzungen (Paare betroffener Kanten) *auszugeben*.

Aufgabe 3 – Lösungsskizze

- modifiziere divide & conquer mergesort-Algorithmus
- zähle beim Sortieren die übersprungenen Elemente mit – genau das sind die Inversionen

- zwei Kanten (u, v) und (w, z) schneiden sich gdw. $u < w$ und $v > z$
- bezeichne Knoten auf Lage 1 als $a, b, c, \ldots$ und Knoten auf Lage 2 als $1, 2, 3, \ldots$
- bilde Folge der Kanten aus Sicht von Lage 2, d.h. $a1, c1, d1, a2, b2, b3, d3, e3, \ldots$
- wende Algorithmus aus (a) an und sortiere lexikographisch
- leicht zu sehen, dass es Graphen mit $O(|E|^2)$ Kreuzungen gibt