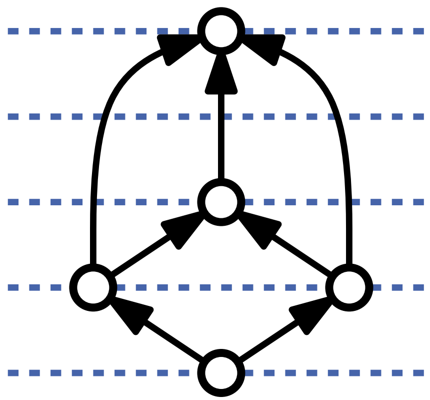
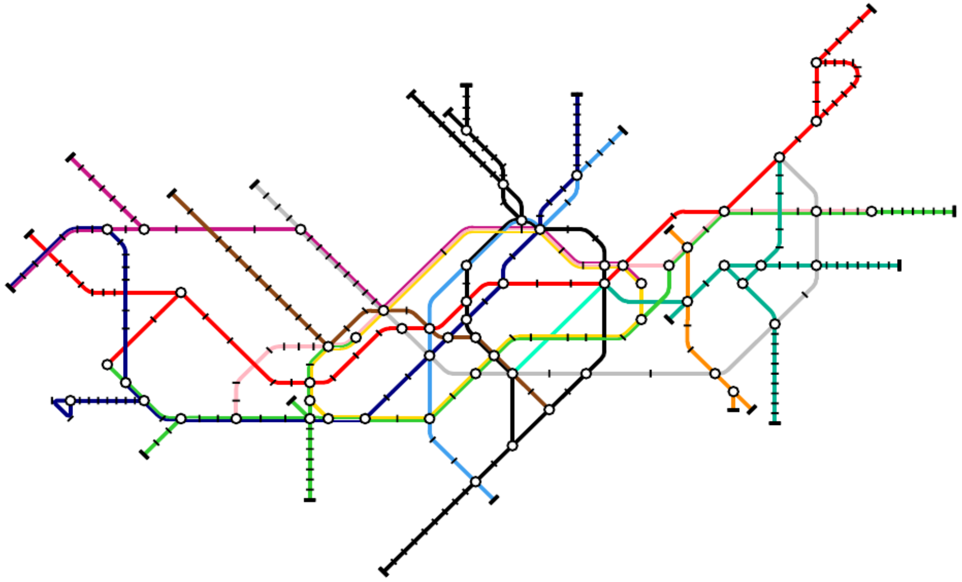
Vierte Übung

Algorithmen zu Visualisierung von Graphen
Lagenlayouts und Metro Maps

Thomas Bläsius

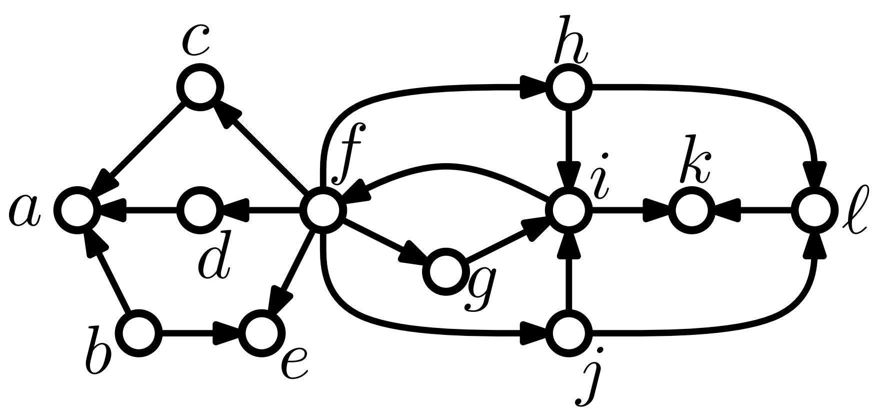
INSTITUTE OF THEORETICAL INFORMATICS
KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT)





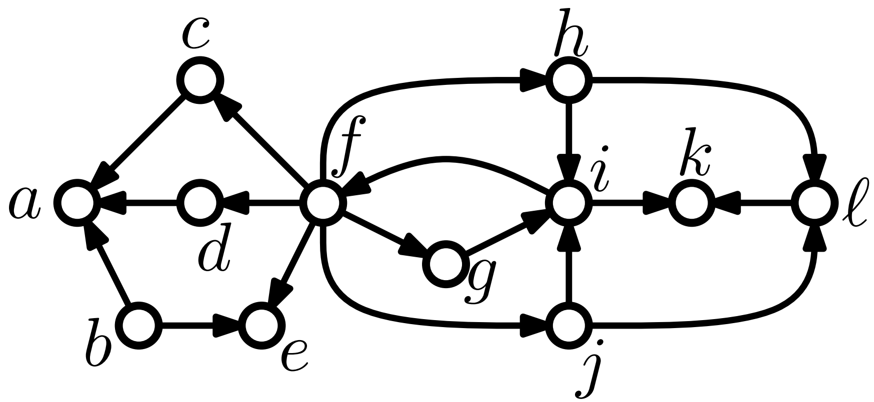
Aufgabe 1 – Lagenlayout

Führen Sie den in der Vorlesung vorgestellten Algorithmus zur Generierung von Lagenlayouts schrittweise für den nebenstehenden 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

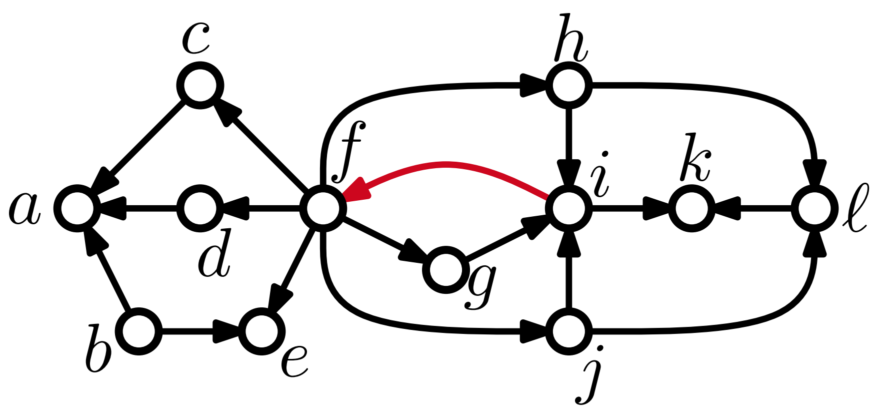
Finde $E' \subseteq E$ sodass $G - E'$ azyklisch und $|E'|$ minimal (FAS)



Institute for Theoretical Informatics
Algorithmics Group I

Lagenlayout – 1. FAS

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

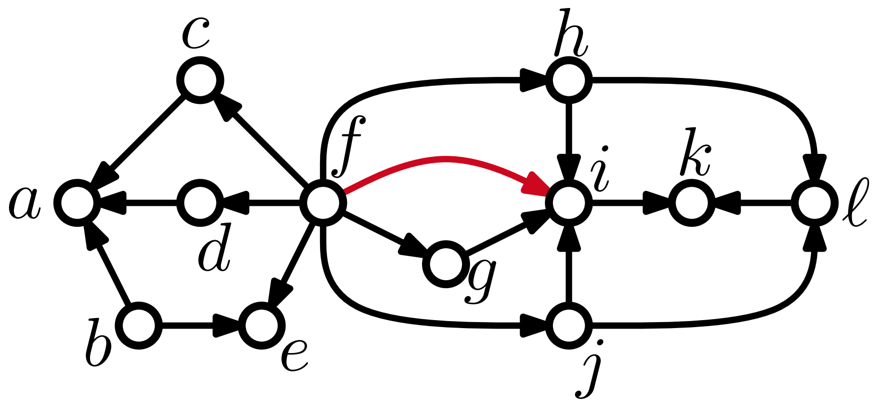


Institute for Theoretical Informatics
Algorithmics Group I

Lagenlayout – 1. FAS

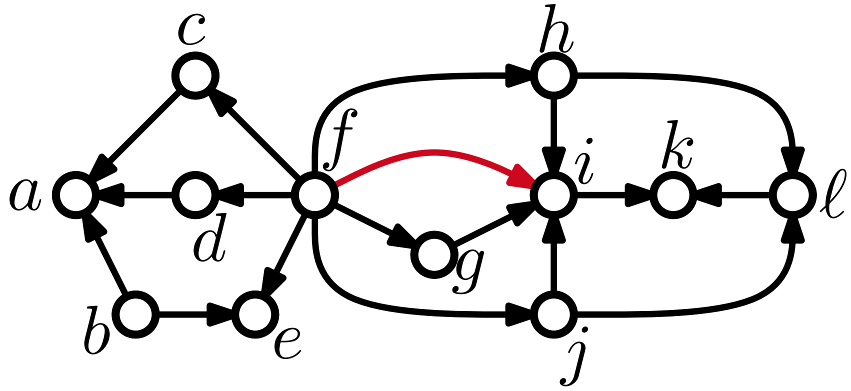
Finde $E' \subseteq E$ sodass $G - E'$ azyklisch und $|E'|$ minimal (FAS)

Kehre die Orientierung der Kanten in E' um



Lagenlayout – 2. Lagenzuordnung

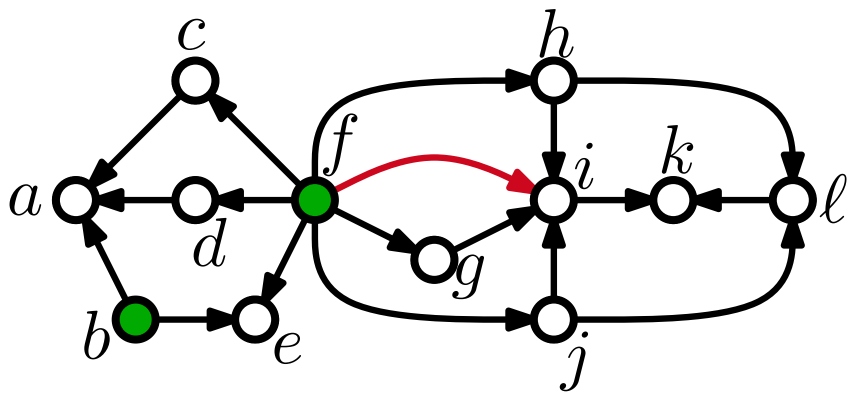
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen



Institute for Theoretical Informatics
Algorithmics Group I

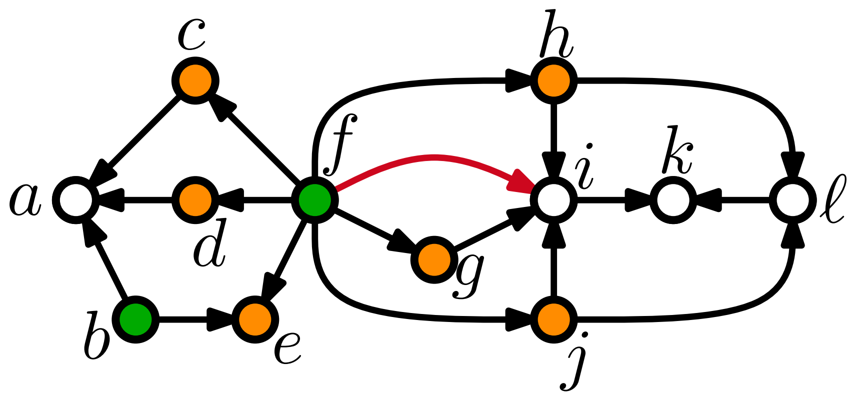
Lagenlayout – 2. Lagenzuordnung

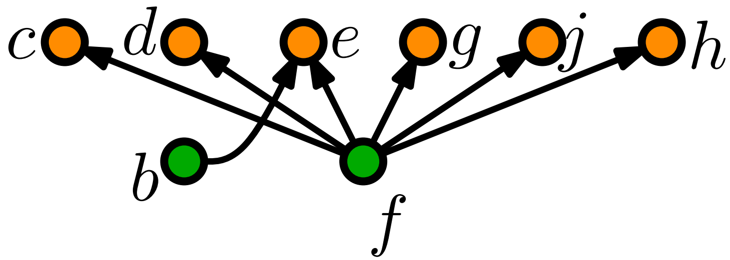
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen



Lagenlayout – 2. Lagenzuordnung

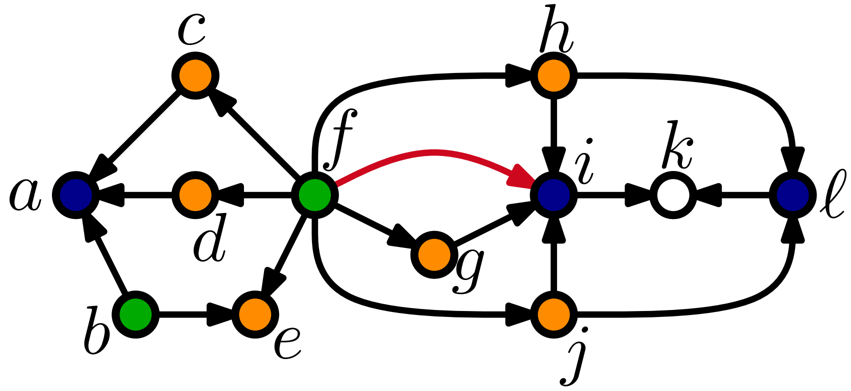
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen

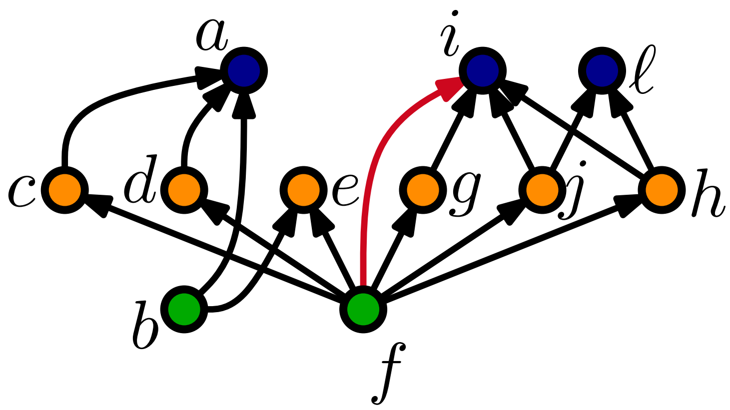




Lagenlayout – 2. Lagenzuordnung

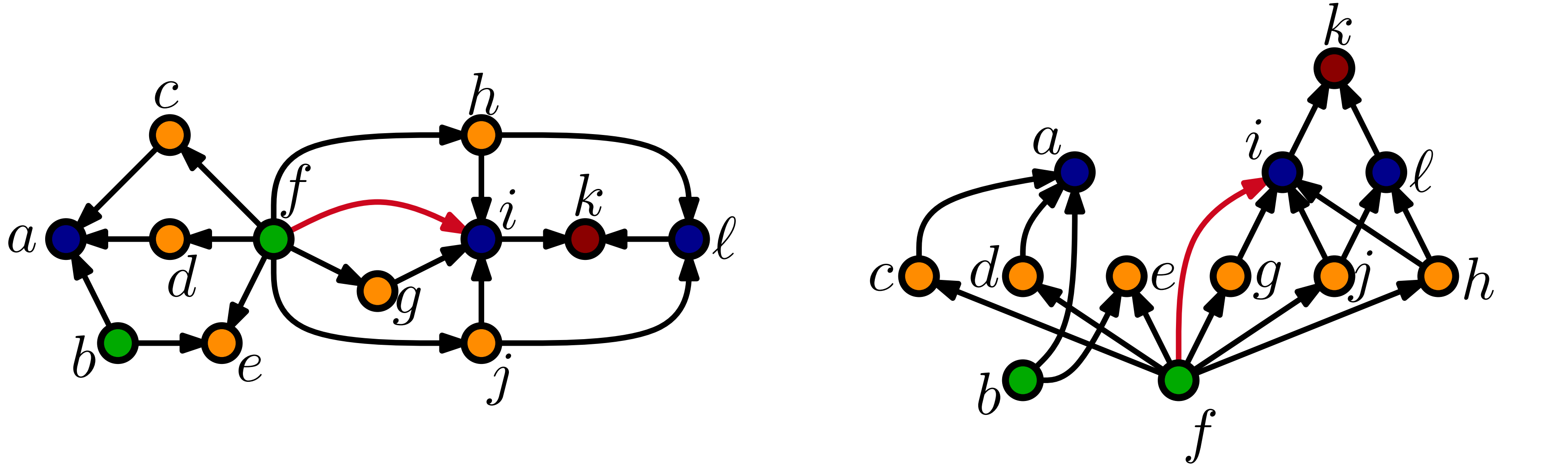
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen





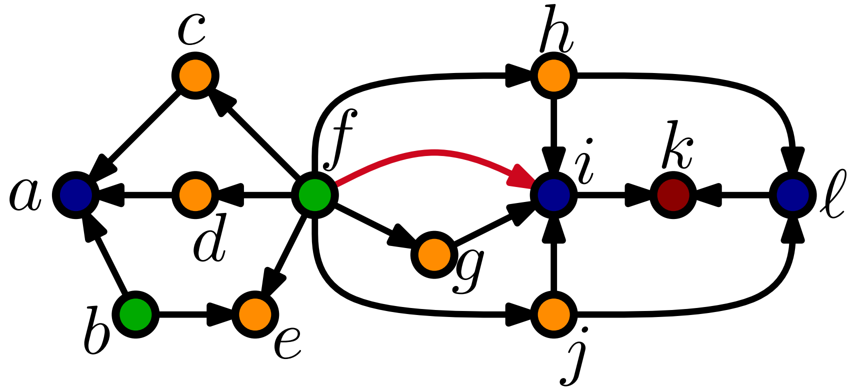
Lagenlayout – 2. Lagenzuordnung

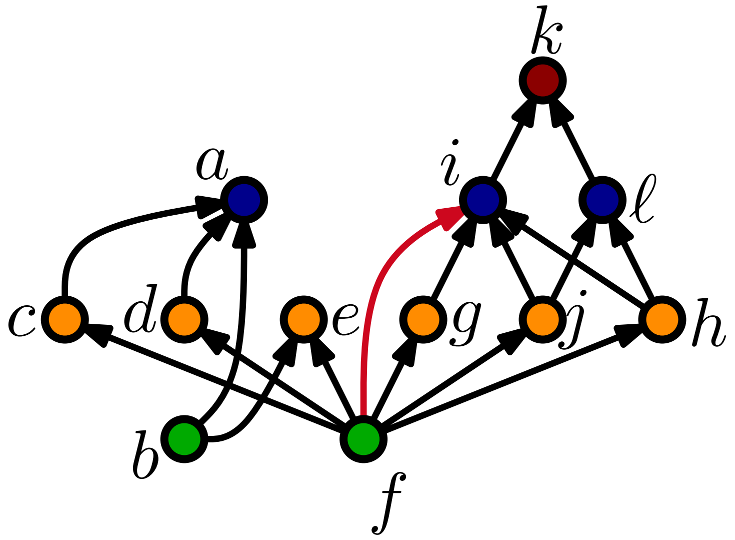
Lagenzuordnung min. Höhe: iteratives Löschen von Quellen



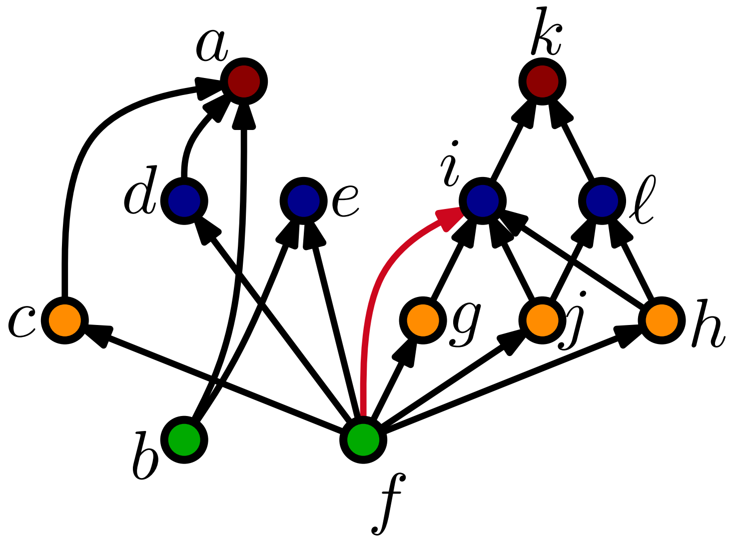
Lagenlayout – 2. Lagenzuordnung

Lagenzuordnung min. Höhe: iteratives Löschen von Quellen





Min Höhe bei auf 4 beschränkter Breite: genaues hinschauen

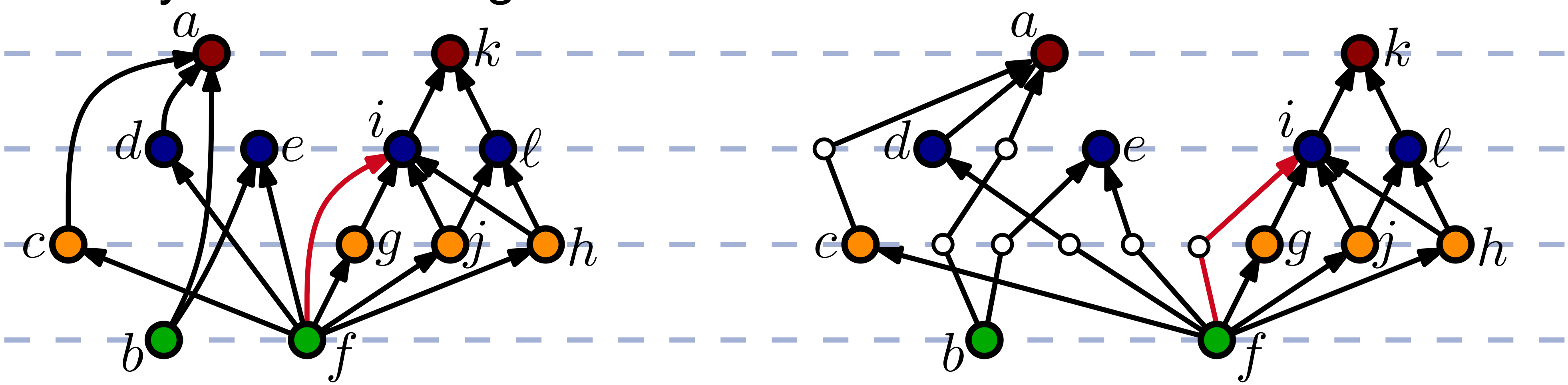


Lagenlayout – 3. Kreuzungsminimierung

Dummyknoten einfügen

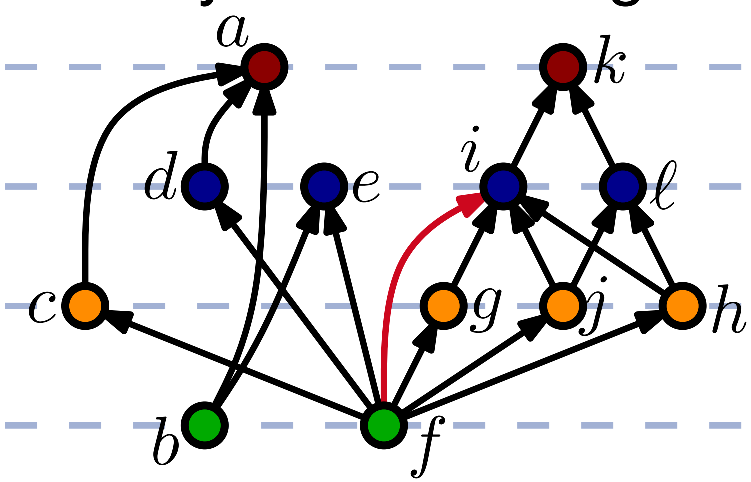
Lagenlayout – 3. Kreuzungsminimierung

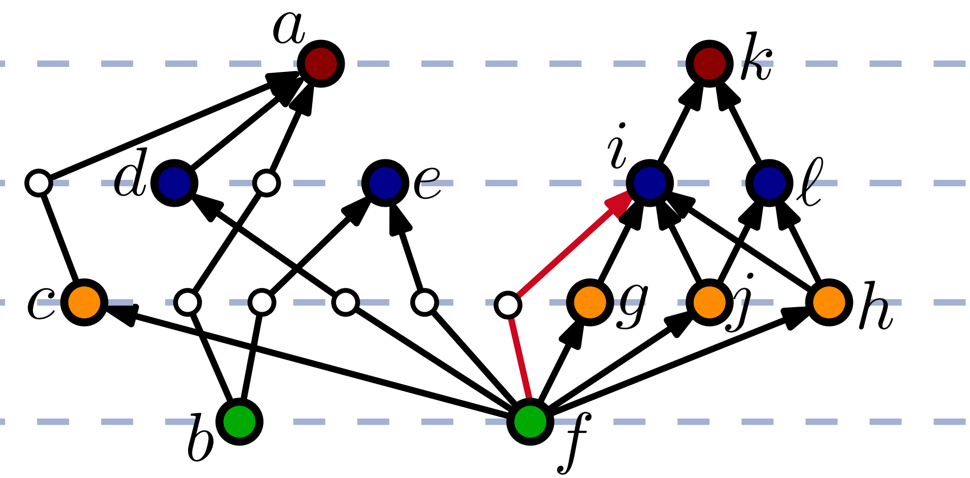
Dummyknoten einfügen



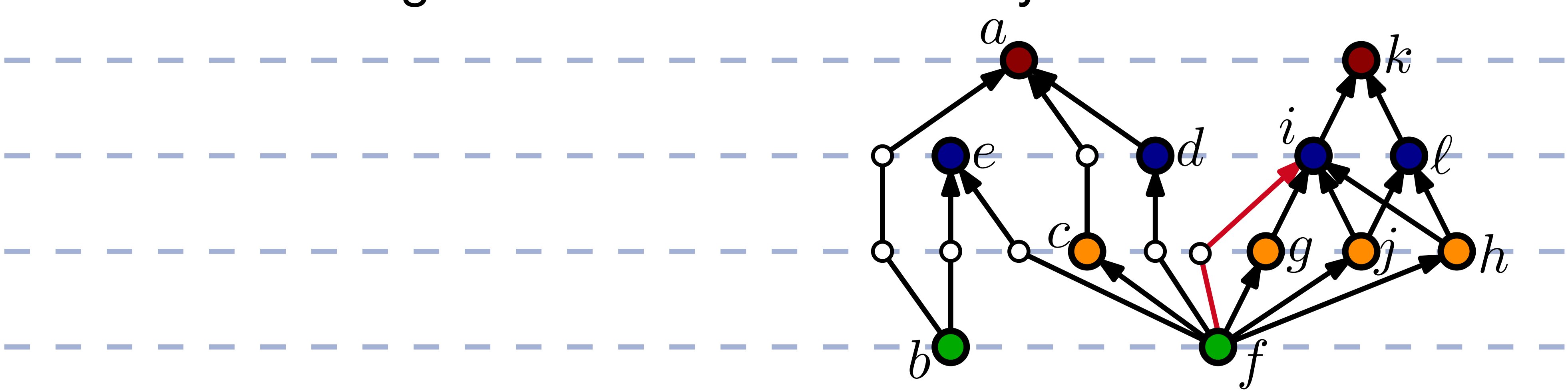
Lagenlayout – 3. Kreuzungsminimierung

Dummyknoten einfügen



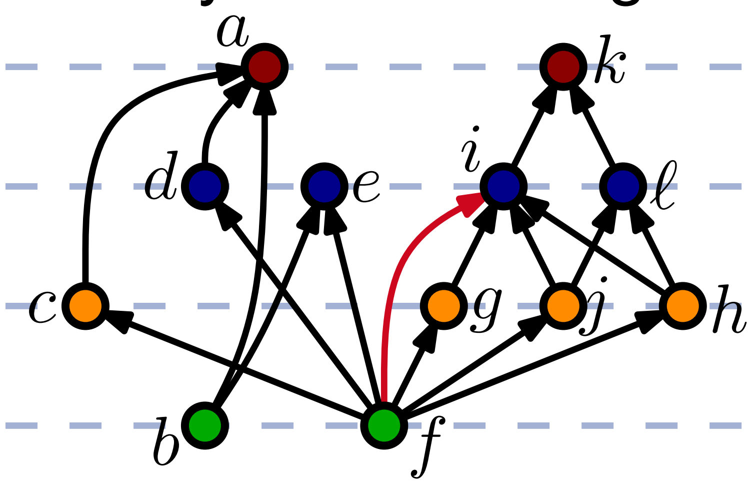


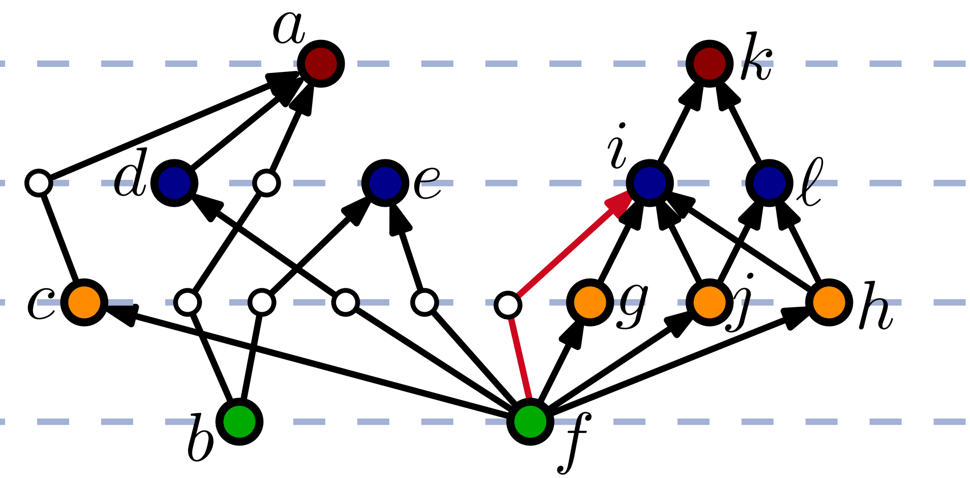
Knickminimierung und löschen der Dummyknoten



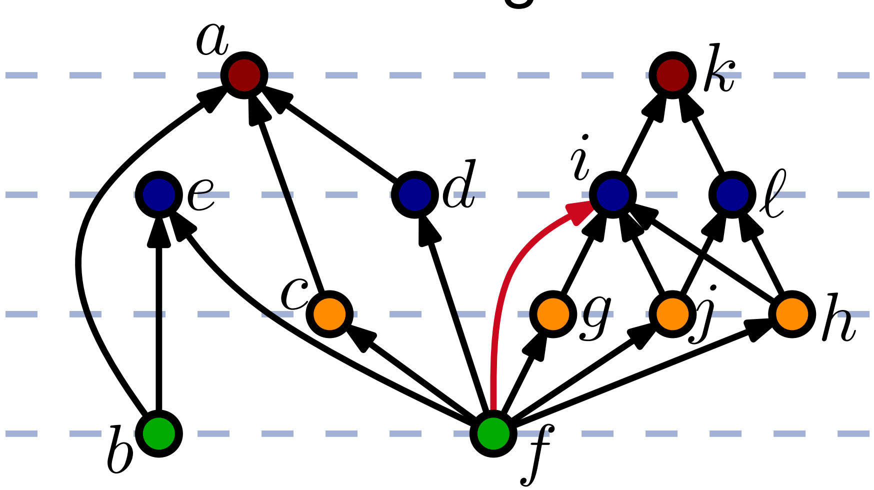
Lagenlayout – 3. Kreuzungsminimierung

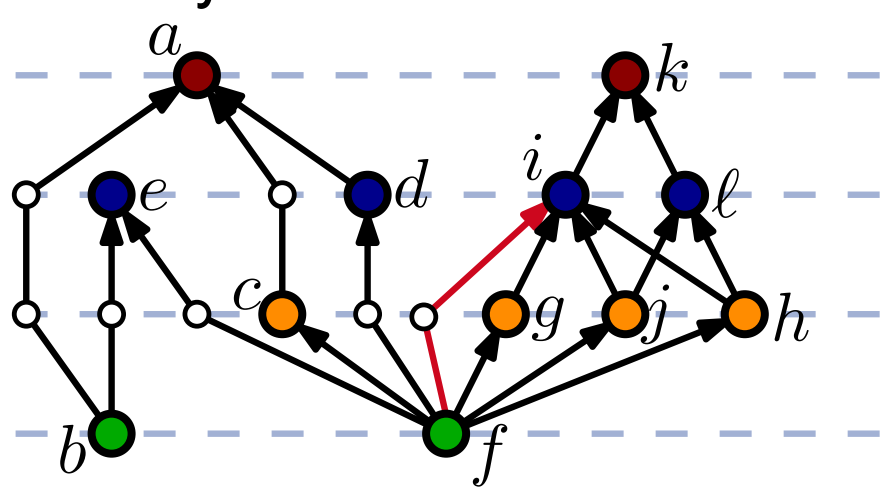
Dummyknoten einfügen





Knickminimierung und löschen der Dummyknoten





Aufgabe 2 – Greedy-Heuristik (FAS)

Wie kann die Heuristik von Eades, Lin und Smyth zur Berechnung eines möglichst großen azyklischen Teilgraphen in linearer Zeit implementiert werden?

Algorithmus 1: Greedy-Algorithmus (Eades, Lin & Smyth)

$A' := \emptyset$;
while $V \neq \emptyset$ **do**
    while *in V existiert eine Senke v* **do**
        $A' \leftarrow A' \cup N^{\leftarrow}(v)$
        entferne v und $N^{\leftarrow}(v)$: $\{V, n, m\}_{\text{sink}}$
    Entferne alle isolierten Knoten aus V: $\{V, n, m\}_{\text{iso}}$
    while *in V existiert eine Quelle v* **do**
        $A' \leftarrow A' \cup N^{\rightarrow}(v)$
        entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\text{source}}$
    if $V \neq \emptyset$ **then**
        sei $v \in V$ mit $|N^{\rightarrow}(v)| - |N^{\leftarrow}(v)|$ maximal;
        $A' \leftarrow A' \cup N^{\rightarrow}(v)$
        entferne v und $N^{\rightarrow}(v)$: $\{V, n, m\}_{\{=,<\}}$

Aufgabe 3 – Fehlstände Zählen

Karlsruhe Institute of Technology

(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 einer Laufzeit $O(|E| \log |V|)$ bestimmt. Begründen Sie, warum es nicht möglich ist, in dieser worst-case-Laufzeit alle Kreuzungen (d.h. die Paare betroffener Kanten) *auszugeben*.

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 einer Laufzeit $O(|E| \log |V|)$ bestimmt. Begründen Sie, warum es nicht möglich ist, in dieser worst-case-Laufzeit alle Kreuzungen (d.h. die Paare betroffener Kanten) *auszugeben*.

Aufgabe 4 – Baryzenter-Heuristik

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

Institute for Theoretical Informatics
Algorithmics Group I

Aufgabe 5 – Metro Maps

(a) Beim Zeichnen von Metro Maps können neben Knickzahl, Kantenlänge und Erhaltung der relativen Lage andere Optimierungskriterien relevant sein. Modifizieren Sie das in der Vorlesung vorgestellte ILP, sodass zusätzlich der Umfang der Bounding-Box bzw. die Breite bei festgelegter Höhe minimiert wird. Warum ist die Fläche der Zeichnung kein geeignetes Optimierungskriterium?

Aufgabe 5 – Metro Maps

(b) Zur Minimierung der Kantenlänge wurde in der Vorlesung die folgende Kostenfunktion definiert.

$$\mathsf{cost}_{\mathsf{length}} = \sum_{\{u,v\} \in E} \mathsf{length}(\{u, v\})$$

Geben Sie lineare Nebenbedingungen an, die sicherstellen, dass die Variable $\mathsf{length}(\{u, v\})$ der Länge der Kante $\{u, v\}$ bezüglich der Normen L_1, L_∞ und L_2 entspricht. Verwenden sie dazu nur die Positionen $(x(u), y(u))$ und $(x(v), y(v))$ von u bzw. v.