

3. Projekt: Lagenlayout

Ausgabe: 17. Juni 2009 **Abgabe:** 10. Juli 2009, 8 Uhr
Die Bearbeitung in Zweiergruppen ist ausdrücklich erwünscht.

Ziel ist das Zeichnen von gerichteten Graphen $G = (V, E)$ mit möglichst vielen aufwärts zeigenden Kanten. Sie werden dazu den am häufigsten verwendeten Ansatz implementieren, nämlich das *Sugiyama-Framework*, und darin in jeder der vier Phasen eine Heuristik anwenden. Sie können neben den angegebenen auch noch eigene weitere Heuristiken implementieren und Erweiterungen und Verbesserungen vornehmen (bitte entsprechend dokumentieren).

Das Paket `<gruppeXX>.p03` ist als zip-Datei bis zum Abgabetermin an `gd_u_S09@inf.uni-konstanz.de` zu schicken (Quelldateien genügen). Achten Sie bitte auf die Kommentierung der Klassen und Methoden mittels javadoc und Inline-Kommentaren, insbesondere auf die `author`-tags im Klassenkommentar.

Implementieren Sie eine Klasse `<gruppeXX>.p03.SugiyamaLayouter`, welche die Klassen der einzelnen Phasen benutzt, als ein entsprechendes yEd-Modul.

Aufgabe 1: Entfernen von Kreisen

7 Punkte

Bestimmen Sie eine Teilmenge von Kanten, die Sie temporär umdrehen, damit der Graph azyklisch wird. Implementieren Sie dazu den verbesserten Greedy-Algorithmus von Eades, Lin und Smyth (1993) im Skriptum.

Implementieren Sie eine Klasse `<gruppeXX>.p03.Cycles`.

Aufgabe 2: Lagenzuordnung

7 Punkte

Ordnen Sie jedem Knoten eine Lage zu. Verwenden Sie dazu die Methode aus dem Skriptum, mit der die Höhe minimiert wird: Quellen (Knoten ohne eingehende Kante) werden dabei ganz unten in V_0 platziert; alle anderen Knoten werden auf der Lage V_i platziert, die der Länge i eines längsten Weges von irgendeiner der Quellen entspricht.

Berechnen Sie dazu zunächst eine topologische Sortierung und berechnen Sie dann in dieser Reihenfolge für jeden Knoten v seine Lage $y(v) = \max_{u:(u,v) \in E} y(u) + 1$. Am Ende ist die Lagenzuordnung eine Partition $V = V_0 \uplus \dots \uplus V_k$, wobei k die Länge des längsten Weges von einer Quelle zu irgendeinem Knoten bezeichnet und $v \in V_i \Leftrightarrow y(v) = i$ für alle $v \in V$.

Implementieren Sie eine Klasse `<gruppeXX>.p03.Layers`. Sie dürfen die Klasse `y.algo.NodeOrders` für die topologische Sortierung verwenden.

[Bitte wenden]

Aufgabe 3: Kreuzungsminimierung

8 Punkte

Fügen Sie für alle Kanten $(u, v) \in E$ mit $|y(u) - y(v)| > 1$, die also mehr als eine Lage überspannen, Dummyknoten auf dieser Kante in den entsprechenden überspannten Lagen ein und behandeln Sie diese Dummys zunächst wie normale Knoten.

Bei gegebener Lagenzuordnung wird nun in jeder Lage V_i eine Permutation der darin enthaltenen Knoten gesucht, die man auch als ganzzahlige x -Koordinaten $1, \dots, |V_i|$ auffassen kann.

Implementieren Sie eine *Pendelheuristik*: In einem Einzelschritt werden nur zwei benachbarte Lagen V_i, V_{i+1} betrachtet ($0 \leq i \leq k - 1$), von denen eine festgehalten wird und in der anderen die Anordnung der Knoten verbessert werden soll. Beim einfachen Verfahren *greedy switch* werden zwei direkt nebeneinander liegende Knoten vertauscht, wenn sich dadurch die Anzahl der Kantenkreuzungen verringert; diese Vertauschungen werden von links nach rechts ausgeführt.

Dieser Einzelschritt wird pendelnd von oben nach unten und zurück einige Male wiederholt, bis keine Verringerung der Kreuzungszahl im ganzen Graphen mehr möglich ist.

Implementieren Sie eine Klasse `<gruppeXX>.p03.Crossings`.

Aufgabe 4: Koordinatenzuweisung

8 Punkte

Weisen Sie den Knoten bei gegebener horizontaler Reihenfolge innerhalb der Lagen die endgültigen x -Koordinaten zu. Legen Sie iterativ, über alle Lagen von oben nach unten, jeden Knoten aus der darunter liegenden, nicht festgehaltenen Lage in den Median-Durchschnitt seiner p Nachbarn in der benachbarten festgehaltenen Ebene: Wenn $x_1 < \dots < x_p$ die Koordinaten dieser Nachbarn sind, wird der Knoten auf $x_{(p+1)/2}$ gelegt, falls p ungerade ist, und sonst auf $(x_{p/2} + x_{p/2+1})/2$.

Falls für einen Knoten die so bestimmten Koordinaten dazu führen, dass die Reihenfolge innerhalb der Lage nicht erhalten bleibt, erzwingen Sie die Erhaltung der Reihenfolge. Beachten Sie außerdem, dass Sie evtl. die absoluten Werte der beiden Lagen für die Berechnung noch aneinander ausrichten müssen (verschieben und strecken).

Stellen Sie die Dummyknoten als Knicke der Kanten dar.

Implementieren Sie eine Klasse `<gruppeXX>.p03.Coordinates`.