Universität Karlsruhe (TH)
Institut für Theoretische Informatik

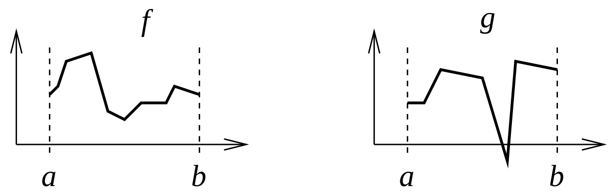
Prof. Dr. Peter Sanders, Dr. Rob van Stee
G. Veit Batz, Johannes Singler

7. Übungsblatt zur Algorithmentechnik WS 2007/08

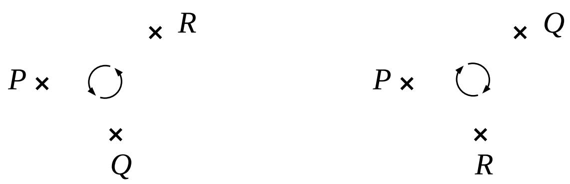
http://algo2.iti.uni-karlsruhe.de/algotech.php
{sanders|vanstee|batz|singler}@ira.uka.de

Aufgabe 1 (*Stückweise lineare Funktionen, Schwierigkeit* $3 + 2 + 4$)

Sei $[a, b] \subset \mathbb{R}$ ein abgeschlossenes Interval. Weiter seien jeweils k Stützstellen $a = x_1 < x_2 < \cdots < x_k = b$ mit Funktionswerten $y_1, \ldots, y_k$ gegeben. Eine *stückweise lineare Funktion* ist dann ein stetiges $f : [a, b] \to \mathbb{R}$ mit $f(x_i) = y_i$ für alle $i \in \{1, \ldots, k\}$, dessen Graph jeweils die Punkte (x_i, y_i) und (x_{i+1}, y_{i+1}) mit einer geraden Strecke verbindet, für $1 \leq i < k$ (siehe Abbildung).



a) Seien drei Punkte $P, Q, R \in \mathbb{R}^2$ gegeben. Für die Reihenfolge (P, Q, R) sind diese Punkte in der Ebene dann entweder *entgegen* dem Urzeigersinn (Abbildung links) oder *mit* dem Urzeigersinn (Abbildung rechts) orientiert.



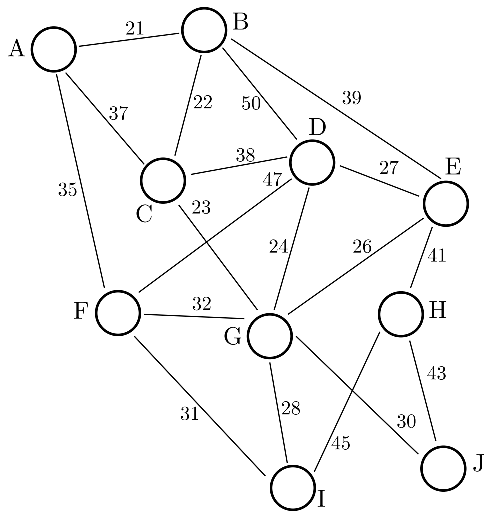
Schreiben Sie in Pseudocode eine Funktion *clockwise*$(P, Q, R : \mathbb{R}^2) : \{-1, 0, 1\}$, die 1 zurückgibt, wenn (P, Q, R) *im* Urzreigersinn orientiert ist, -1, wenn *entgegen* dem Urzeigersinn, und 0, wenn die drei Punkte auf einer Geraden liegen. Dabei darf nicht dividiert werden, die Laufzeit soll konstant sein.

b) Wie kann man mit Hilfe von *clockwise* feststellen, ob sich zwei Liniensegmente schneiden?

c) Schreiben Sie in Pseudocode einen Algorithmus, der ermittelt, ob für zwei stückweise lineare Funktionen $f, g : [a, b] \to \mathbb{R}$ stets $f(x) \leq g(x)$ gilt (für $x \in [a, b]$) oder nicht. Verwenden Sie dabei die Funktion *clockwise* aus Teilaufgabe a). Die Laufzeit darf höchstens $\mathcal{O}(k + \ell)$ betragen (k und ℓ seien jeweils die Anzahlen der Stützstellen von f und g), dividieren ist nach wie vor nicht erlaubt. Hinweis: Der Algorithmus sollte ähnlich wie *vergleichsbasiertes Mischen* arbeiten.

Aufgabe 2 (*Minimale Spannbäume, Schwierigkeit* $1 + 1$)

Berechnen Sie für den umseitig oben stehenden Graphen einen minimalen Spannbaum. Verwenden Sie dazu einerseits den Algorithmus von Kruskal, andererseits den von Jarník-Prim (startend bei G). Geben Sie jeweils den Zustand nach dem 7. Schritt sowie den Endzustand an. In welcher Reihenfolge werden die Kanten beschritten?



Aufgabe 3 (*Hashing, Schwierigkeit* $1 + 2 + 1 + 1$)

a) Gegeben sei eine Hashtabelle mit 10 Einträgen. Dabei sei die Hashfunktion h definiert über $h(x) = x \mod 10$. Verwenden Sie einerseits Hashing mit verketteten Listen, andererseits Hashing mit linearer Suche (Torus), um folgende Operationen durchzuführen.

   Einfügen von 27, 41, 63, 78, 23, 10, 66, 71. Löschen von 66.

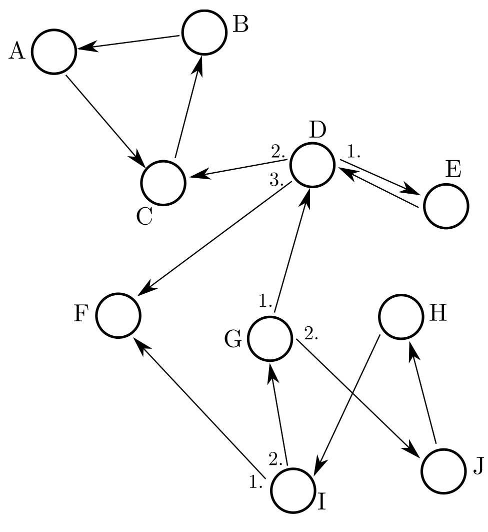
b) Wie viele Stellen müssen im Durchschnitt abgesucht werden, wenn nach jedem enthaltenen Element x gesucht wird, und zusätzlich jeweils nach $x + 1$?

c) Wie groß ist der Speicherverbrauch, wenn sowohl Zeiger als auch Element ein Maschinenwort benötigen, und mit einem Nullzeiger das Ende einer Liste markiert werden kann?

d) Hätte eine andere Reihenfolge das Ergebnis verändert?

Aufgabe 4 (*Starker Zusammenhang, Schwierigkeit* 2)

Finde sie im unten stehenden Graphen alle stark zusammenhängenden Komponenten. Benutzen Sie dazu den DFS-basierten Algorithmus aus der Vorlesung, beginnen Sie bei H. In den mehrdeutigen Fällen ist explizit angeschrieben, in welcher Reihenfolge die Kanten abgearbeitet werden sollen. In welchem Zustand befindet sich der Algorithmus, nachdem C zum zweiten Mal erreicht wurde (Markierung der Knoten, Inhalt der Stapel)?



Geben Sie ebenso den Endzustand des Graphen an.