Ludwig-Maximilians-Universität München
Department "Institut für Informatik"
PD Dr. Peer Kröger
Gregor Jossé

München, 08.01.2014

Geo-Informationssysteme

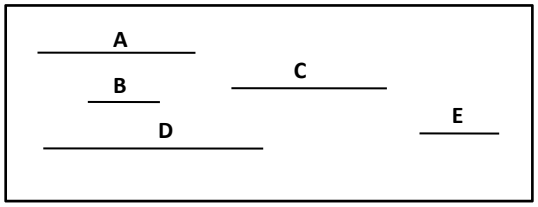
WS 20013/14

Übungsblatt 6: Algorithmen der Computer-Geometrie

Besprechung: 10.01.2014

Aufgabe 6-1 (Streckensichtbarkeit)

Gegeben sei eine Menge S von n horizontalen Strecken in der Ebene, bei denen die x-Koordinaten aller Anfangs- und Endpunkte paarweise verschieden sind. Gesucht sind alle Paare von Strecken, die sich gegenseitig *sehen* können. Zwei Strecken s und t in S sind gegenseitig sichtbar, wenn es eine vertikale Gerade gibt, die s und t, aber keine weitere Strecke der Menge S zwischen s und t schneidet. Gegeben sei folgendes Beispiel, in dem sich (A,B), (A,D), (B,D) und (C,D) *sehen*:



Entwerfen Sie einen Plane Sweep Algorithmus zur Lösung des Problems.
Welche Laufzeit besitzt der Algorithmus?

Aufgabe 6-2 (Schnitt von orthogonalen Strecken mit Divide-and-Conquer-Technik)

Entwerfen Sie einen Algorithmus, der die Operation $Y \star V$ in $O(|Y| + |V| + |Y \star V|)$ berechnet (siehe Kap. 6, Folie 29 & 30).

Aufgabe 6-3 (Intervallbaum)

Gegeben sei die Menge A, B, C, D, E, F von Intervallen mit $A = [4, 9], B = [12, 15], C = [10, 12], D = [2, 6], E = [3, 5]$ und $F = [8, 13]$.

(a) Man skizziere den Intervallbaum zur Speicherung dieser Intervallmenge (die jeweiligen Intervall- Listen sind ebenfalls mit einzuzeichnen)

(b) Man führe eine Punktanfrage für den Punkt x = 5 durch. In welcher Reihenfolge werden die gefundenen Intervalle ausgegeben?

(c) Man gebe einen Algorithmus $IntervalQuery(Node, [au, ao])$ für die Bestimmung aller Intervalle in einem Intervallbaum $Node$ an, die ein Anfrageintervall $[au, ao]$ schneiden.

Lösungsvorschlag:

b) Punktanfrage für Punkt $x = 5$: Beginne bei Node = Wurzel, traversiere nach Algorithmus POINTQUERY im Skript auf Seite 182 bis zu dem Knoten mit Label 5. Analog dazu bezeichnen wir darum im folgenden den Punkt x mit y.

POINTQUERY(Node $= 8,\ 5$):

positiver Test auf "$y <$ Node.y" $\quad (5 < 8) \quad \Rightarrow$

- Ausgabe aller Intervalle I_i in u-Liste bis $I_i.u > y$ $\quad$ (bei F.$y = 8 > 5$) $\quad \Rightarrow \quad$ **A**
- POINTQUERY(Node.LEFT $= 4,\ 5$)
  - Fall "Node.$y < y$" $\quad (4 < 5) \quad \Rightarrow$
    - Ausgabe aller Intervalle I_i in o-Liste bis $I_i.o < y$ $\quad$ (tritt nicht auf) $\quad \Rightarrow \quad$ **D, E**
    - POINTQUERY(Node.RIGHT $= 5,\ 5$) $\quad \Rightarrow \quad$ Treffer, aber keine Intervalle mehr übrig zur Ausgabe. $\quad \Rightarrow \quad$ Stop

Ausgabe der Intervalle: (**A, D, E**)

c) siehe Algorithmus 1

Algorithm 1 INTERVALQUERY(Node, $[au, ao]$)

Input: Knoten Node eines Intervallbaumes und Intervall $a = [au, ao]$

```
1:  function IntervalQuery(Node, [au, ao])
2:      if Node = ∅ then
3:          return;
4:      end if
5:      if Node.y ∈ [au, ao] then                          ▷ au ≤ Node.y ≤ ao
6:          gebe alle Intervalle I_i der u-Liste aus;      ▷ und in beiden Teilbäumen weitersuchen
7:      else if ao < Node.y then
8:          gebe I_i der u-Liste aus, bis I_i rechts von ao liegt  (I_i.u > ao).   ▷ dann links weiter
9:      else                                               ▷ au > Node.y
10:         gebe I_i der o-Liste aus, bis I_i links von au liegt  (I_i.o < au).   ▷ dann rechts weiter
11:     end if
12:     if au < Node.y then
13:         IntervalQuery(Node.left, [au, ao])
14:     end if
15:     if ao > Node.y then
16:         IntervalQuery(Node.right, [au, ao])
17:     end if
18: end function
```

Output: alle Intervalle, die Intervall $[au, ao]$ schneiden.