Grundlagen der Algorithmischen Geometrie SS 2013
Übungsblatt 8
Universität Bonn, Institut für Informatik I

Für jede Aufgabe werden bis zu vier Punkte vergeben.

Aufgabe 1:

Sei D die Punktmenge $\{(1,1),(2,4),(3,3),(3,5),(4,6),(5,7),(6,2)\}$ in der Ebene. Geben Sie alle Teilbäume eines 2-dimensionalen Bereichsbaumes für D an, die zur Beantwortung der Bereichsanfrage $q = [x_1, x_2] \times [y_1, y_2] = [1.5, 6.5] \times [4.5, 6.5]$ benötigt werden und skizzieren Sie die Bereichsanfrage.

Aufgabe 2:

Orthogonale Bereichsanfragen auf einer Menge von n orthogonalen Objekten lassen sich — wie bei einer Menge von n Punkten — durch einen Bereichsbaum realisieren.

Sei P eine Menge von n achsenparallelen Rechtecken in der Ebene. Beschreiben Sie eine Datenstruktur, die $O(n(\log n)^3)$ viel Speicherplatz benötigt, und die alle Rechtecke, die ganz in einem achsenparallem Rechteck $R = [x_1, x_2] \times [y_1, y_2]$ liegen, in Zeit $O((\log n)^4 + a)$ berichten kann, wobei a die Anzahl der in R enthaltenen Rechtecke ist.

Hinweis: Transformieren Sie das Problem auf eine 4-dimensionale Bereichsanfrage.

Aufgabe 3:

Betrachten Sie folgende Variation eines zweidimensionalen Bereichsbaums: Das Grundgerüst ist genau wie beim Standard-Bereichsbaum ein balancierter Suchbaum $\mathcal{T}$ bezüglich der x-Koordinaten.

An jeden Knoten v dieses Baumes hängen wir nun aber nicht einen weiteren Suchbaum, sondern ein nach y-Koordinaten aufsteigend sortiertes Array A_v, das die zum entsprechenden Knoten korrespondierende Punktmenge enthält.

Ist nun v ein innerer Knoten von $\mathcal{T}$ und ist u sein linker Sohn, w sein rechter Sohn, so speichern wir zu v zusätzlich zwei Integer-Arrays L_v und R_v die die gleiche Länge haben wie A_v:

Hierbei ist $L_v[i] :=$ der kleinste Index j, sodass die y-Koordinate von $A_u[j]$ größergleich der y-Koordinate von $A_v[i]$ ist, falls so ein j existiert, und $L_v[i] := -1$ sonst.

$R_v[i]$ ist analog der kleinste Index j, sodass die y-Koordinate von $A_w[j]$ größergleich der y-Koordinate von $A_v[i]$ ist, falls so ein j existiert, und $L_v[i] := -1$ sonst.

Geben Sie einen Algorithmus an, der eine Rechteckanfrage mithilfe dieser Datenstruktur in Zeit $O(\log n + a)$ beantwortet, wobei n die Größe der in der Datenstruktur gespeicherten Punktmenge und a die Anzahl der Punkte im Anfragerechteck ist. Beweisen Sie Korrektheit und Laufzeit.