CS 5633 Analysis of Algorithms – Spring 04

3/31/04

9. Homework

Due **4/7/04** before class

1. **Bottom-up BST construction (4 points)**
   Let a *sorted* set of n integers be given. You may assume that n is a power of 2. The task is to compute a balanced binary search tree of those integers, where the integers are stored *only in the leaves*, and the inner nodes contain copies of the integers, as discussed in class.

   Give an $O(n)$ algorithm which constructs this balanced binary search tree bottom-up (i.e., starting with the leaves first). You may assume that the input is given in your favorite way (array, linked list, ...).

   Show why the runtime of your algorithm is $O(n)$, and argue what the height of the tree is.

2. **Red-black trees with keys in the leaves (5 points)**
   Given a list of n *unsorted* integers, we want to construct a red-black tree that stores all those integers *only in the leaves*, and the inner nodes contain copies of the integers, as discussed in class.

   a) (3 points) Assume you construct this tree by incrementally inserting the numbers, using the RB-Insert routine (see page 280). Which changes do you have to make to this routine such that it stores the integers in the leaves?

   b) (2 points) Assume now you construct the tree using problem 1.: First sort the numbers (using some $O(n \log n)$-time algorithm), and then construct a balanced binary search tree bottom-up. How can you make red-black color-assignments to the nodes, such that the tree becomes a red-black tree?

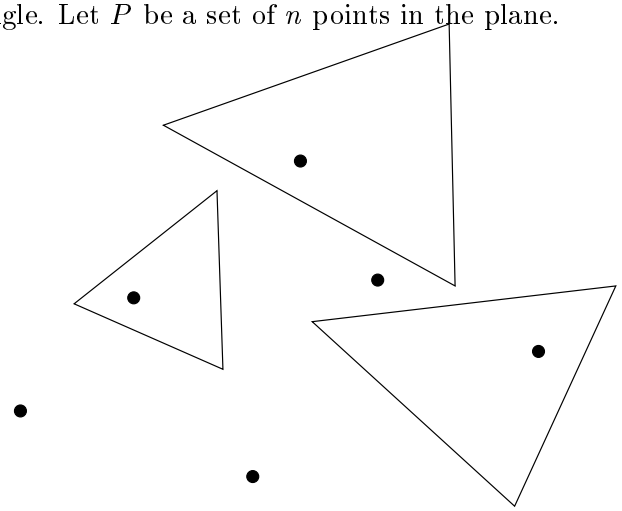
3. **Sweep-line status (4 points)**
   Consider the sweep-line algorithm we had in class for computing if n given line segments intersect or not. We said that the sweep-line status is a "dynamic set" which is implemented using red-black trees. But how exactly is this done?

   Notice that between two consecutive events, the sweep-line status stays the same (because the ordering of the segments stays the same). But if you look at the intersection points of the segments with the sweep line, then the coordinates of those intersection points vary when the sweep-line moves; even if the sweep-line moves just between two consecutive event points. However, the *order* of the segments stays the same.

   How can this functionality be implemented using a balanced binary search tree? You do not have to go into details of the implementation; just state what is stored in the nodes, and which values are used to determine how to insert a new node.

Flip over to back page ⟹

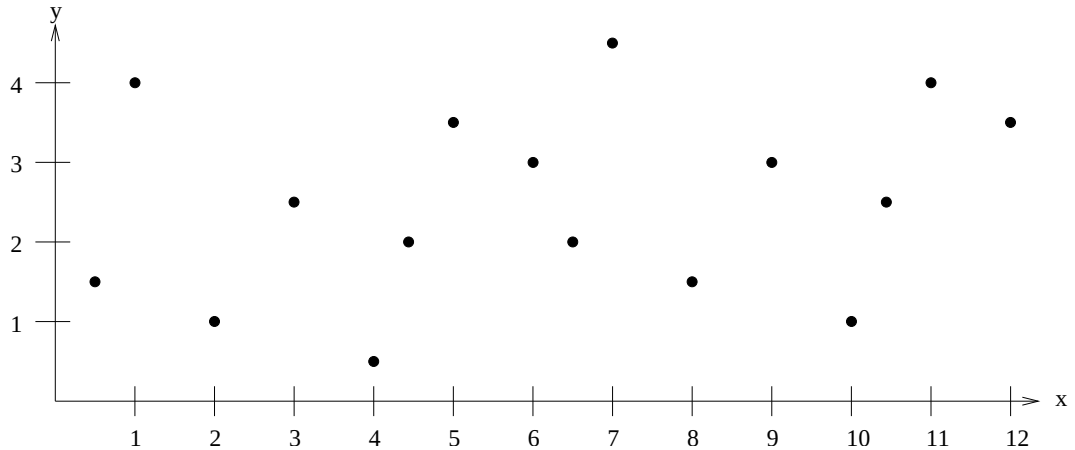
4. **Triangles (5 EXTRA-CREDIT points)**
Let S be a set of n triangles in the plane (each given by its three endpoints). The boundaries of the triangles are disjoint, and no triangle lies completely inside another triangle. Let P be a set of n points in the plane.



Give a sweep-line algorithm which reports all points of P that are not contained in any of the triangles. Your algorithm should run in $O(n \log n)$ time.

- What are the events?
- Let your sweep-line status be an ordered set of intervals (i.e., the "scene" intersected at the sweep-line)
- At any moment in "time" you should have reported all the points *left* of the sweep-line (if they lie outside the triangles).

5. **Range trees (12 points)**
Let $P = \{(0.5, 1.5), (1, 4), (2, 1), (3, 2.5), (4, 0.5), (4.5, 2), (5, 3.5), (6, 3), (6.5, 2), (7, 4.5), (8, 1.5), (9, 3), (10, 1), (10.5, 2.5), (11, 4), (12, 3.5)\}$ be a set of two-dimensional points.



FLIP OVER TO BACK PAGE ⟹

a) (4 points) Construct their primary range tree (with the keys being the x-coordinates). Make sure to store the two-dimensional points in the leaves, or pointers to them (and not just their x-coordinates).

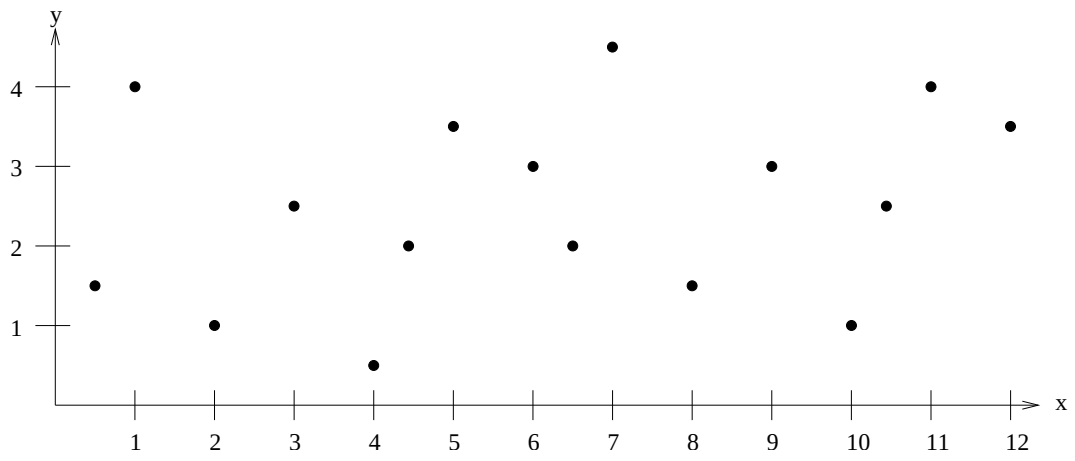
You do not have to show how the tree is constructed; just show the tree. (You may use the algorithm from problem 1. to construct it).

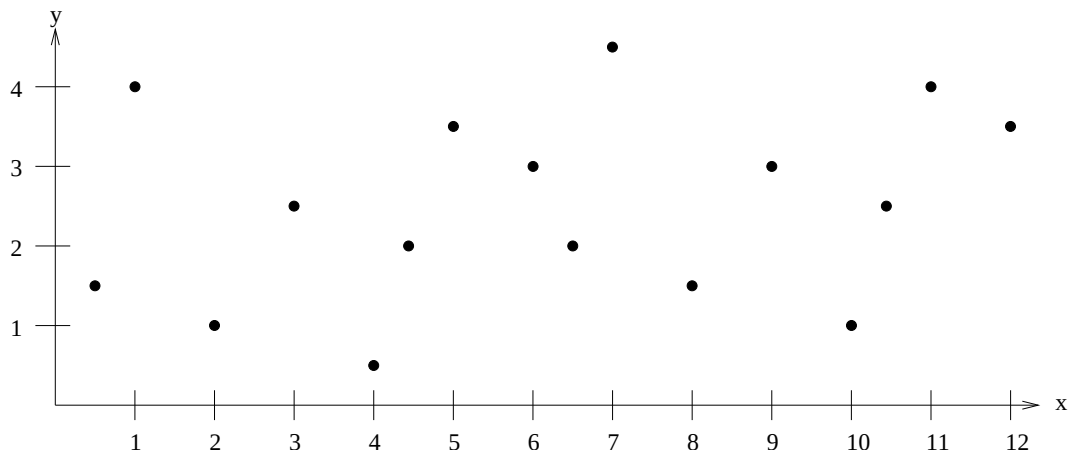
b) (4 points) Now construct all the secondary range trees (with the keys being the y-coordinates). Make sure to store the two-dimensional points in the leaves, or pointers to them (and not just their y-coordinates).

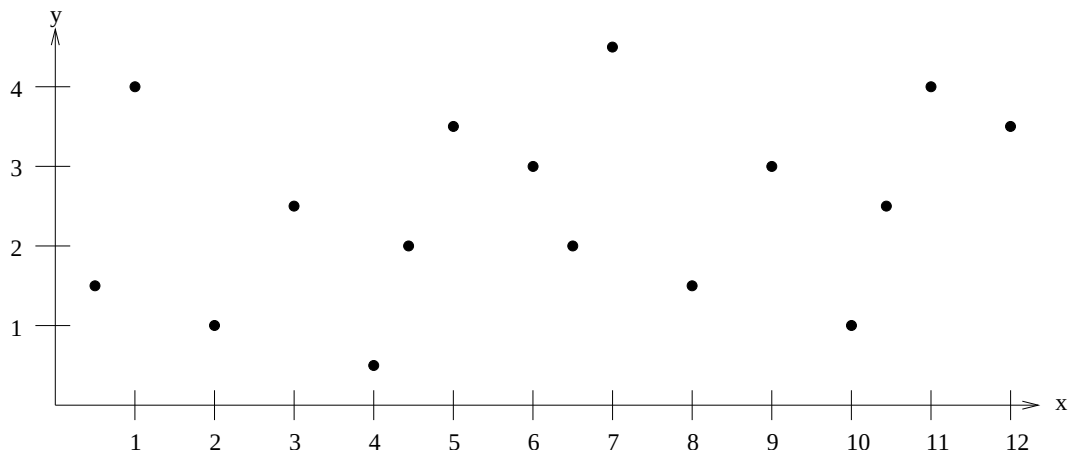
c) (4 points) Consider the query rectangle $[x_1 = 1, x_2 = 6] \times [y_1 = 1.5, y_2 = 4]$. Show how the range reporting query (i.e., "print out all points in the query rectangle") proceeds in the range tree:

- Show the split nodes (in the primary tree, and in the secondary trees).
- Show the search paths (in the primary tree, and in the secondary trees).
- Show which secondary trees are queried.
- Show which points are output (mark the corresponding leaves in the secondary trees).

The back of the homework contains a few copies of the point set, in case you need them.


y
4
3
2
1
x
1
2
3
4
5
6
7
8
9
10
11
12


y
4
3
2
1
x
1
2
3
4
5
6
7
8
9
10
11
12


y
4
3
2
1
x
1
2
3
4
5
6
7
8
9
10
11
12