UNIVERSITY OF WISCONSIN PARKSIDE

Department of Computer Science

CSCI 340/CIS 540 Data Structures/Algorithm Design

Homework #4: 50 points

Outcomes:

- Write programs that use Search Trees

Scoring:

- If you do not submit a .java file containing your source code, your score will be zero.
- If you submit the source code that does not compile, your score will be zero.
- If you submit source code without the correct class name (TwoDTree) or your program partially works, you will receive at most half credit for this assignment.
- Deductions will be made for not meeting the usual requirements:
    - The source code is not formatted according to the usual style guidelines, including commenting on each method to explain its purpose.

Create an IntelliJ Java Project called Program4. Write the comment lines to include the author's name(s) and complete the project following the instructions.

Consider the problem of storing (x, y) points in a binary search tree. Binary search trees expect the values stored in them to have an ordering. That is, we go left or right during our searches based on some sort of less than/greater than relationship. (x, y) points have two orderings, one based on x and the other based on y. Our goal is to implement a data structure, Two-dimensional (2D) Trees, that let us use both orderings.

Two-dimensional trees are a data structure very similar to binary search trees. They divide up geometric space in a manner convenient for use in range searching and other problems.
The idea is to build a binary tree using both the x and y coordinates of the points as keys. The levels of the tree can split either on the x or the y coordinate; typically, the splits alternate between the two dimensions.

At every node in the two-dimensional tree that splits on x, the following invariant holds:
- all the points that have an x coordinate value less than or equal to the nodes x value are found in the left child;
- all the points with a greater x value are in the right child.

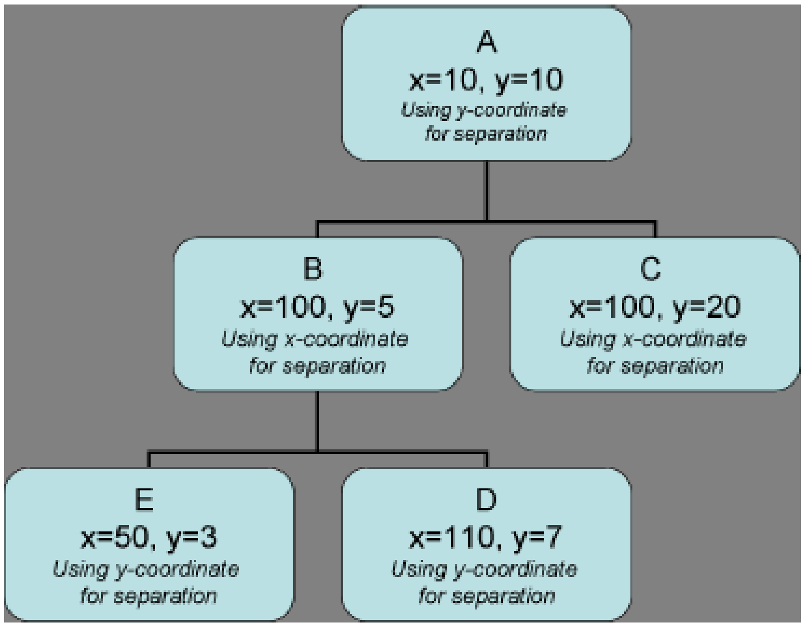
Nodes that split on y are similar: less than or equal to y values go on the left, larger on the right.

Searching in the tree
Like binary search trees, a 2D tree can be used to find elements it contains, by walking recursively from the root of the tree downward, making left/right choices at each node depending on the value of the point being searched for relative to the split value and axis of the current node.

In a 2D tree, it is also possible to search for all elements found within some spatial extent, such as a rectangle in the 2D space. To implement this kind of search, the rectangle is compared against the split value at each point. If the rectangle overlaps the split value, the algorithm must search both children of the current node.

Creating a tree
To create the tree each point is inserted into the tree. The algorithm walks down the tree to find the location that would contain the point if it were in the tree. When this leaf is reached, a new node (containing the

point) is created and inserted there. For example, suppose we want to insert the following points into a 2D-tree:
A(10, 10), B(100,5), C(100,20), E(50, 3), D(110,7)
The result of this sequence is shown below.


A
x=10, y=10
Using y-coordinate for separation
B
x=100, y=5
Using x-coordinate for separation
C
x=100, y=20
Using x-coordinate for separation
E
x=50, y=3
Using y-coordinate for separation
D
x=110, y=7
Using y-coordinate for separation

Your Assignment

You are to develop a class named `TwoDTree` with the following constructors and methods:

1. A default constructor that creates an empty `TwoDTree`.
2. A constructor that takes an `ArrayList<Point>` and creates a tree with all the points inserted into it. Note you should use `java.awt.Point`
3. `public void insert (Point p)` - inserts `p` into the `TwoDTree.` To keep everybody's code consistent, please implement the root as a node making decisions based on y.
4. `public boolean search (Point p)` – returns true if the `Point2D` is in the tree, false, otherwise.
5. `public ArrayList<Point> searchRange (Point p1, Point p2)` – returns an `ArrayList` of all points lying in the rectangle bounded by p1 and p2, inclusive.

Note that using `searchRange()` it is fairly easy to search on just a range of y-values. You simply set `p1.x` to `Integer.MIN_VALUE` and `p2.x` to `Integer.MAX_VALUE`. Similarly, for searching just a range of x values.

You will want to create a private inner interface `TwoDTreeNode` and two private inner classes, `TwoDTreeNodeX` and `TwoDTreeNodeY`, which make decisions based on x or `y`, as appropriate.

This assignment isn't too long or too hard. The real tough part is wrapping your brain around the ideas and translating them into code. There can be some subtle bugs. Start early, so you have time to find and fix the bugs.

Use the `TestTwoDTree.java` file provided with this assignment to make certain your code runs correctly. Your instructor will design different tests for grading.

Grading Rubric:

Requirement	Full credit	Partial credit
Implement the `TwoDTree` Class using the search tree invariants provided (40 points)	Your program implements the `TwoDTree` Class as described.	Your program implements the class, but with some errors.
Format code (follow Style Guidelines) and output as specified (10 points)	Your output is formatted as specified, including proper spacing, spelling, and so on.	You did not follow some or all of the requirements for output.

What to Submit:
Electronically submit your `TwoDTree.java` file. You do not need to submit the test main your instructor has provided. Your code will be tested against a different test main. Do not use any packages.

Good software engineering is expected. Use lots of comments, appropriate indentation, etc. when writing your program.