PRINCETON UNIVERSITY

COS 226–Algorithms and Data Structures

Week 5: *BST's & Kd-Trees (Videos §8.D §10.C & Algorithms §3.2)*

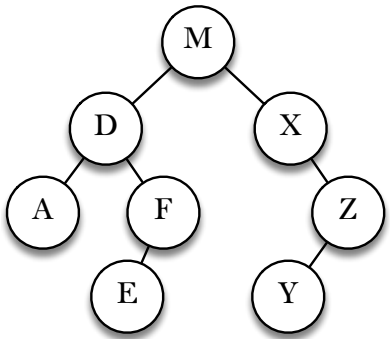
Version: October 12, 2017

Exercise 1 – Binary Trees and Binary Search Trees

A. Pre-order, in-order, and post-order traversals all use a similar recursive strategy for visiting nodes of a binary tree. However, the nodes are printed in different orders depending on the order of the recursive calls relative to the print statement, as illustrated in this code (currently set to output the pre-order of a binary tree).

```
private static void genericTraversal(Node x) {
  if(x != null) {
    StdOut.println(x.key);  // we are doing PRE-ORDER
    genericTraversal(x.left);
    // StdOut.println(x.key);  // uncomment if doing IN-ORDER
    genericTraversal(x.right);
    // StdOut.println(x.key);  // uncomment if doing POST-ORDER
  }
}
```

Write down the order in which the keys in this tree are printed for various traversals.



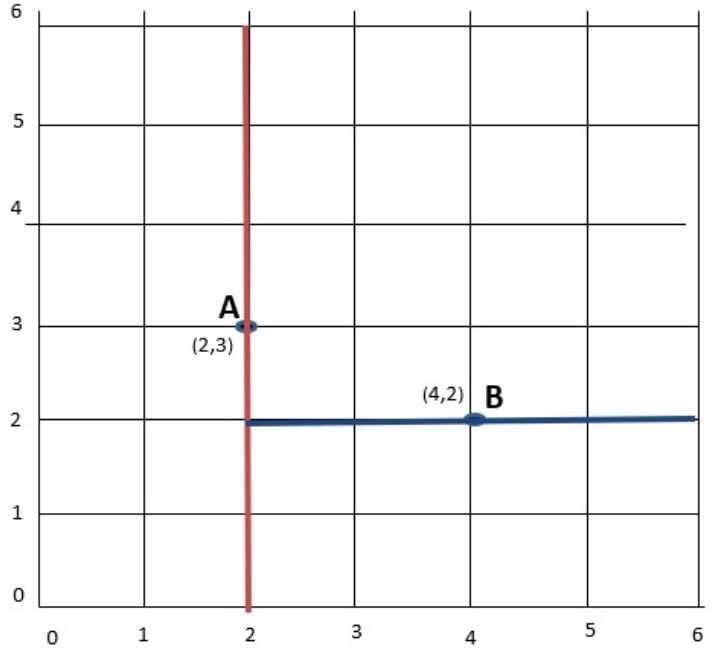
- Pre-Order traversal: _____
- In-Order traversal: _____
- Post-Order traversal: _____

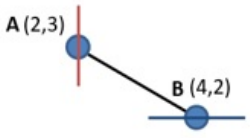
B. Draw a *binary tree* (not necessarily a binary search tree) which has the following pre-order traversal: Q, B, A, C, Y, X, Z. Are there other binary trees that have this pre-order traversal, or is this tree unique?

C. Draw a *binary tree* (not necessarily a binary search tree) which has the following pre-order traversal: Q, B, A, C, Y, X, Z and in-order traversal B, A, Q, C, Z, X, Y. Are there other binary trees that have this pre-order traversal and in-order traversal, or is this tree unique?

Exercise 2 – Kd Trees

A. Suppose we are inserting the 2D points $[A(2,3), B(4,2), C(4,5), D(3,3), E(1,5), F(4,4)]$ into a Kd-Tree. We have shown the Kd-Tree after inserting the first two points including how the 2D plane is bisected as points are inserted. To the right we show the 2D binary tree that represents the Kd-Tree. Insert the other points into the Kd-Tree and show how the plane is bisected. Be careful when inserting $(4,4)$ as ties are treated the same as greater than. On the axes, draw each point as well as the horizontal or vertical lines that bisect the plane through each point.





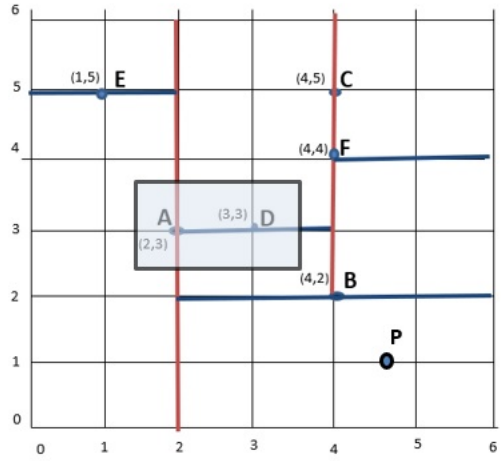
B. List the bounding rectangles (RectHV) for each point in the table below. The bounding rectangle RectHV of A(2, 3) is given as

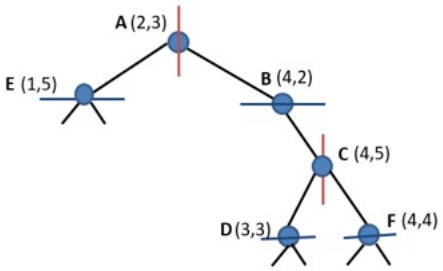
$[(-\infty, -\infty), (\infty, \infty)]$

Write down RectHV for all points.

Point	RectHV
A	$[(-\infty, -\infty), (\infty, \infty)]$.
B	
C	
D	
E	
F	

C. Consider a range query on the shaded rectangle. Write a number next to each node in your KdTree (drawn in Part A) that is considered during the KdTree traversal by the range search algorithm. Start with number 1, and then sequentially increase the numbers to show which order the nodes were considered in the range search algorithm. Circle the nodes that were found to be inside the shaded rectangle. Do not count null nodes or nodes whose corresponding rectangles do not intersect the query rectangle. It is fine to use your solution in Part A to mark the nodes that were considered.





D. Consider a nearest neighbour query on point p. For your convenience, the correct answer from part A is provided below. Number each node (starting with 0) by the order in which it is visited by the nearest neighbor algorithm UNLESS that node's corresponding rectangle rules out that node or its children. For nodes that are pruned based on rectangle distance, write an X over the node. Do not number null nodes.