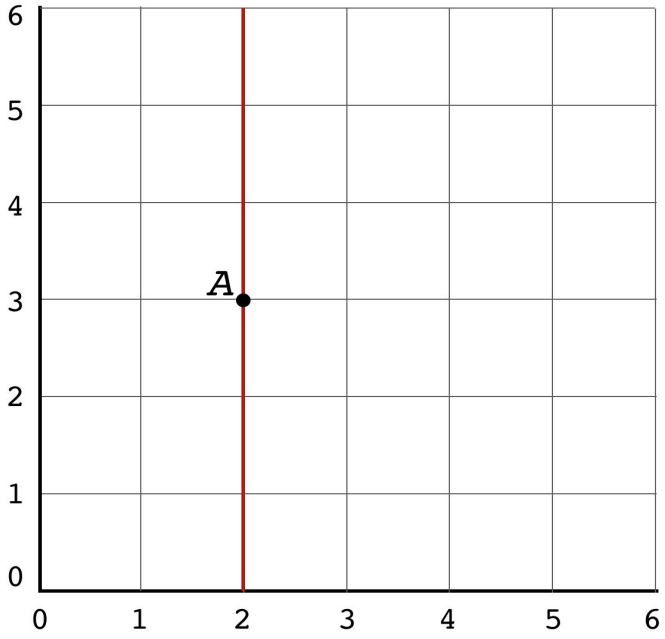
EXERCISE 1: Kd-Trees

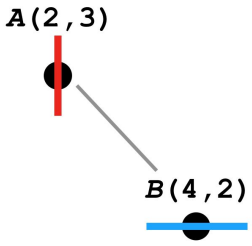
(a) Draw the Kd-tree that results from inserting the following points:

`[A(2, 3), B(4, 2), C(4, 5), D(3, 3), E(1, 5), F(4, 4), G(1, 1)]`

Draw each point on the grid, as well as the vertical or horizontal line that runs through the point and partitions the plane, or a subregion of it.

Note: While inserting, go left if the coordinate of the inserted point is less than the coordinate of the current node. Go right if it is greater than ***or equal***.



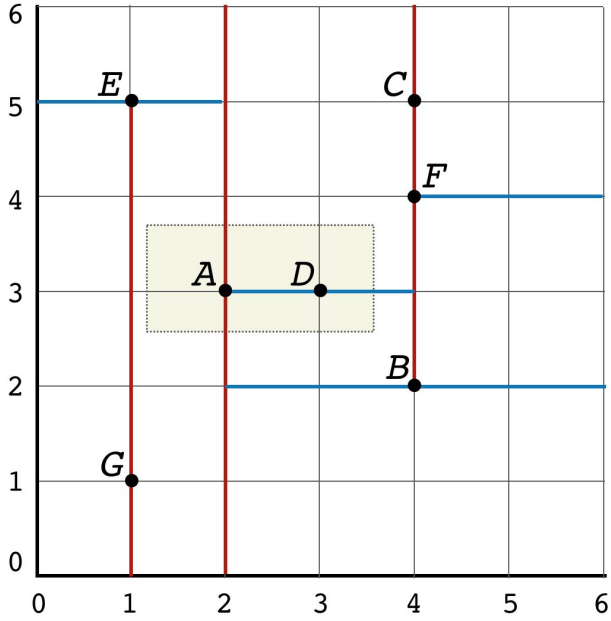


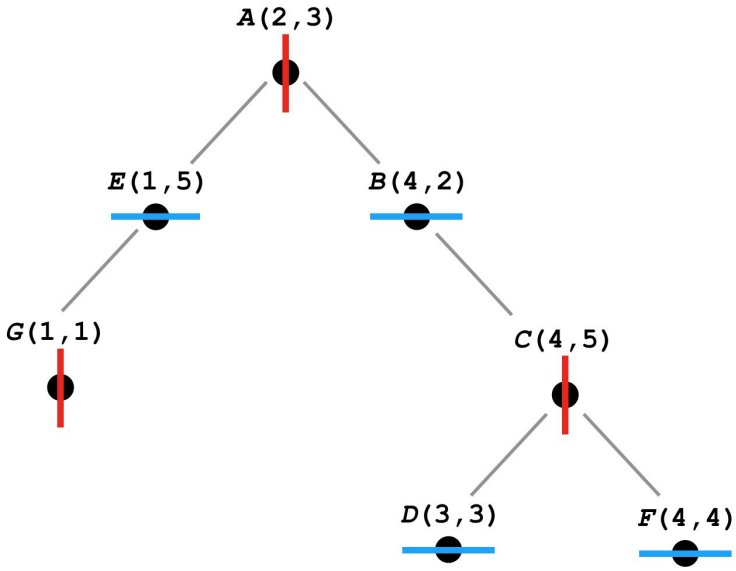
(b) Give each point's bounding rectangle.

`A(2, 3)`	$[(-\infty, -\infty), (+\infty, +\infty)]$
`B(4, 2)`	
`C(4, 5)`	
`D(3, 3)`	
`F(4, 4)`	$[(4, 2), (+\infty, +\infty)]$
`E(1, 5)`	$[(-\infty, -\infty), (2, +\infty)]$
`G(1, 1)`	

(c) Number the tree nodes according to the visiting order when performing a *range query* using the rectangle shown below. Label pruned subtrees with ✘.

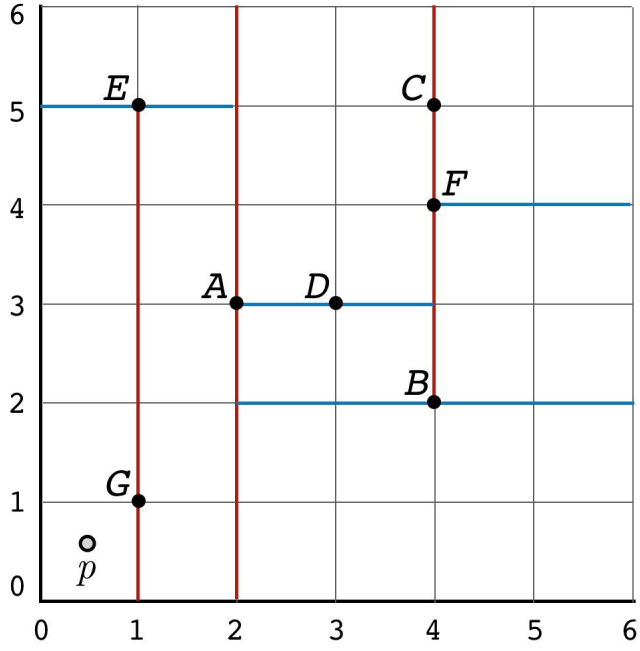
Remember. The range search algorithm recursively searches in both the left and right subtrees unless the bounding rectangle of the *current* node does not intersect the query rectangle.

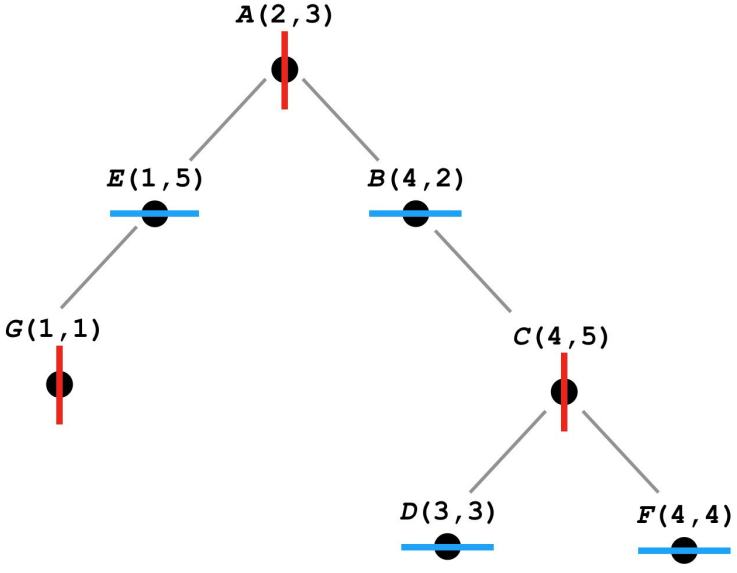




(d) Number the tree nodes according to the visiting order when performing a *nearest neighbor (NN) query* using the point p shown below. Label pruned subtrees with ✘.

Remember. The NN algorithm recursively searches in both the left and right subtrees unless the distance between *p* and the bounding rectangle of the *current* node is larger than the distance between *p* and the nearest point found so far.





EXERCISE 2: Ordered BST Operations

(a) Download `precept5.zip` from the precepts page. Unzip the file and open the project using IntelliJ.

(b) Implement `int rank(Key key)` in `BST.java`. This method should return the number of keys in the BST that are strictly less than the given key.

(c) Implement `int countRange(Key lo, Key hi)`. This method should return the number of keys in the BST that are between `lo` and `hi` (inclusive).