Ve281 Data Structures and Algorithms
Written Assignment Five

This assignment is announced on Nov. 8th, 2018. It is due by 5:40 pm on Nov. 16th, 2018. The assignment consists of four problems.

1. (30%) Binary search tree

   (a) (18%) Suppose that we insert a sequence of keys 4, 9, 2, 5, 1, 7, 6, 3, 8 into an initially empty binary search tree. Draw the resulting tree.

   (b) (6%) Suppose that we further delete the key 5 from the tree you get in Problem (1a). Draw the resulting tree.

   (c) (6%) Suppose that we further delete the root from the tree you get in Problem (1b). Draw the resulting tree.

2. (20%) Given a binary tree, in which each node is associated with an integer key, it may not possess the binary search tree property. Describe a **most runtime-efficient** algorithm that determines whether such a tree is indeed a binary search tree. Also, tell us what the runtime of your algorithm is. Show us why your algorithm is a most runtime-efficient one.

   Note: You can describe your algorithm in English. However, if you find that writing pseudo-code/code is helpful for your illustration, feel free to do so.

3. (30%) Suppose the node of a binary search tree is defined as follows.

```
struct node {
    Key   key;   // key
    node* left;  // left child
    node* right; // right child
};
```

   Implement the following function which gets the predecessor of a given key in the tree:

```
node* getPred(node* root, Key key);
// REQUIRES: The tree rooted at "root" is non-empty.
//           "key" is in the tree rooted at "root".
// EFFECTS:  Return the predecessor of "key" in the tree rooted at "root".
//           Return NULL if there is no predecessor
```

You can assume the following function is availabe:

```
node* findMax(node* root);
// REQUIRES: The tree rooted at "root" is non-empty.
// EFFECTS:  Return the node with the maximal key in the tree rooted
//           at "root".
```

4. (20%) Suppose nodes $A, B, ..., J$ are located on a 2-D plane shown in Figure 1. We insert these nodes in the order $A, B, ..., J$ into a k-d tree. Show the final tree. Assume the comparison dimension of the root is the x dimension.

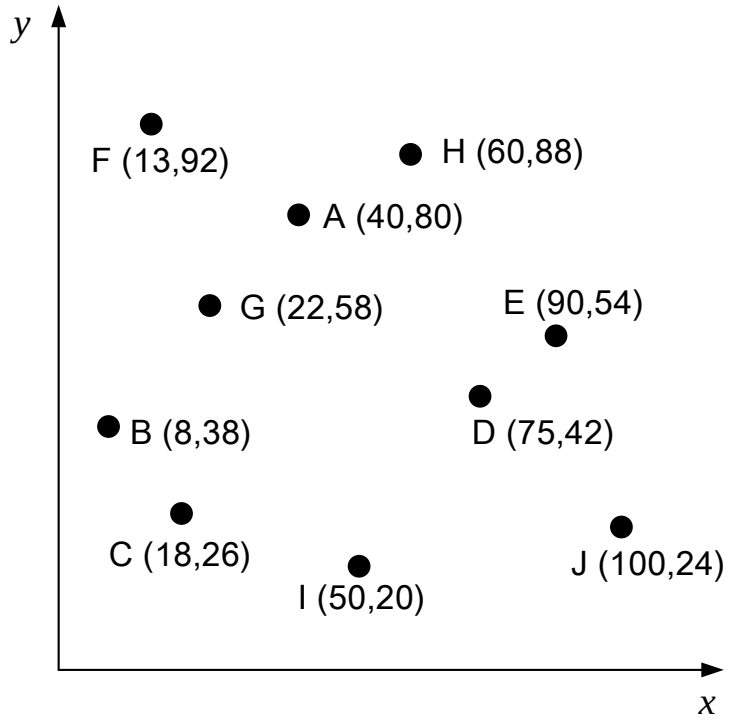

Figure 1: The locations of a number of nodes in a 2-D plane.