Homework 3: Binary Search Trees and its Generalizations

Due Date Regarding your submission, please upload your submission on or before 11 : 59 PM, 11/18/2016. 24 hours extension rule apply. Also, you need to upload your signature file (see blackboard for the due date).

Important Note Your work will be tested by the graders in Ubuntu environment, using g++ version 5 or later. It is your responsibility to test your code in the environment before submission. If you code cannot be compiled there, your submission will not graded and no further re-submission are allowed.

Objective In this programming homework you will implement two data structures:

a). Binary Search Trees (abbrev. as BST): this is the basic version with the usual dictionary operations: insert, delete and search. No balancing actions will be implemented.

b). *kd* Trees: this can be considered as a generalization of binary search trees for higher dimensional data. You will need to implement kd trees when $k = 2$. together with the usual dictionary operations: insert, delete and search.

You are asked to implement the above data structures as `C++ classes` and to conduct simple computational experiments (via the `C++` language) to verify your implementation of these two data structures. In addition, you will need to implement a show function for each of the data structures. This function will create a `dot` file which can be use to generate image files to display the data structures visually.

Introduction The design of a data structure that maintains a set of data properly with respect to insert, delete and search operations is often referred as the *dictionary* problem. In this context, one way to implement a dictionary is to use binary search trees. In this homework, we will implement the basic version (abbrev. as BST) and one of its variant (abbrev. $2d$ trees) which is designed to handle two dimensional data.

Requirements

a). General

- Nodes: the nodes of a these trees must have both a *left link* and a *right link.* For each node, an data item is stored in it (or *being referenced to* from the node).
- Data Items:

  An data item that are stored in a Binary Search Tree are records with two fields: key (of type int) and value (of type int). We will use the notation [key value] to represent a data item. For example, [5 20] represents the data item with key =5 and value = 20.

  An data item that are stored in a $2d$ Tree are records with three fields: x (of type int), y (of type int) and value (of type int). We will use the notation [x y value] to represent a data item. For example, [5 10 20] represents the data item with x =5, y=10 and value = 20.

  Note that, in this homework, it is possible that there are multiple data items with the same key. The comparisons we perform will use the following convention as illustrated in the diagram below:

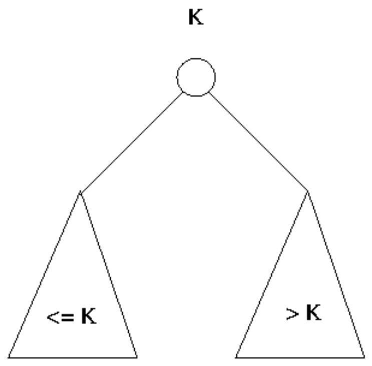

Figure 1: The rule that describes the ordering of data items in BST and $2d$ Trees with respect to the key K

- Height:

  The height of a BST tree or a $2d$ tree is defined in the same way. Let T be either a BST tree or a $2d$ tree, the height of T, denoted by $H(T)$, is defined as follows:

$$H(T) = \begin{cases} -1 & T \text{ is the empty tree} \\ max\{H(T_l), H(T_r)\} + 1 & \text{Otherwise} \end{cases}$$

  T_l and T_r are the left and right subtrees of T respectively.

b). **Binary Search Trees**

Your implementation, in C++ should satisfy the following:

1). A BST class: This models the binary search trees we studied in class. It should contain the essential dictionary operations insert, delete and search

2). A member function Height which will return the height of the BST tree

3). A member function Show which will return a dot file for the BST tree (A dot file is one that specify a graph and can be used by some graph drawing programs (e.g. dot, neato etc.) to create image file of the graph being specified there.

c). $2d$ **Trees** Your implementation, in C++ should satisfy the following:

1). A $2d$-Tree class: This models the binary search trees we studied in class. It should contain the essential dictionary operations insert, delete and search

2). A member function Height which will return the height of the $2d$ tree

3). A member function Show which will return a dot file for the $2d$ tree (A dot file is one that specify a graph and can be used by some graph drawing programs (e.g. dot, neato etc.) to create image file of the graph being specified there.

Graphic files Instructions on dot files and the creation of graph image files via the dot program are posted within our blackboard site.

Experiments, Test Cases, Submission They will be stated in a separate document.