COMP2211

Project 1

Due : Friday, March 6, 2015

This project consists of two tasks. The result of your work will include some code modules as well as a document. Both code and document should be zipped and uploaded to the submission area on OurVLE on or before the submission deadline. The project should be completed in groups (where each group will have a maximum of two people). The name of the zipped file to be submitted should contain the id numbers of all group members.

TASK 1

Comparison of algorithms (20 Marks)

The accompanying files myList.java, and SearchDriver.java, represent respectively a file containing two competing search implementations, and a tester module. The tester module operates by accepting a list size, and then generates a list of the size specified populated with randomly increasing numbers. The tester then accepts the type of search being tested, and then allows the user to search for a value. (The values accepted by the tester procedure are integers, and therefore values used to test should be less than two billion. All test values should be positive). You will need the JDK installed. An interface that can interpret java programs such as BlueJ will also be useful. The function seqSearch implements a sequential search in the list, while binSearch implements a binary search of an ordered list.

a. Complete the functions seqSearch and binSearch, found in mylist.java (8 Marks)

b. Analyze both seqSearch and binSearch thereby stating the respective orders of growth. (5 marks)

c. Tabulate the times taken by each search method when searching for(7 marks)

(i) the value 5,000 in a list of length 100

(ii) the value 50 in a list of length 100

(iii) the value 800,000 in a list of length 10,000

(iv) the value 5,000 in a list of length 10,000

(v) the value 80,000,000 in a list of length 1,000,000

(vi) the value 500,000 in a list of length 1,000,0000

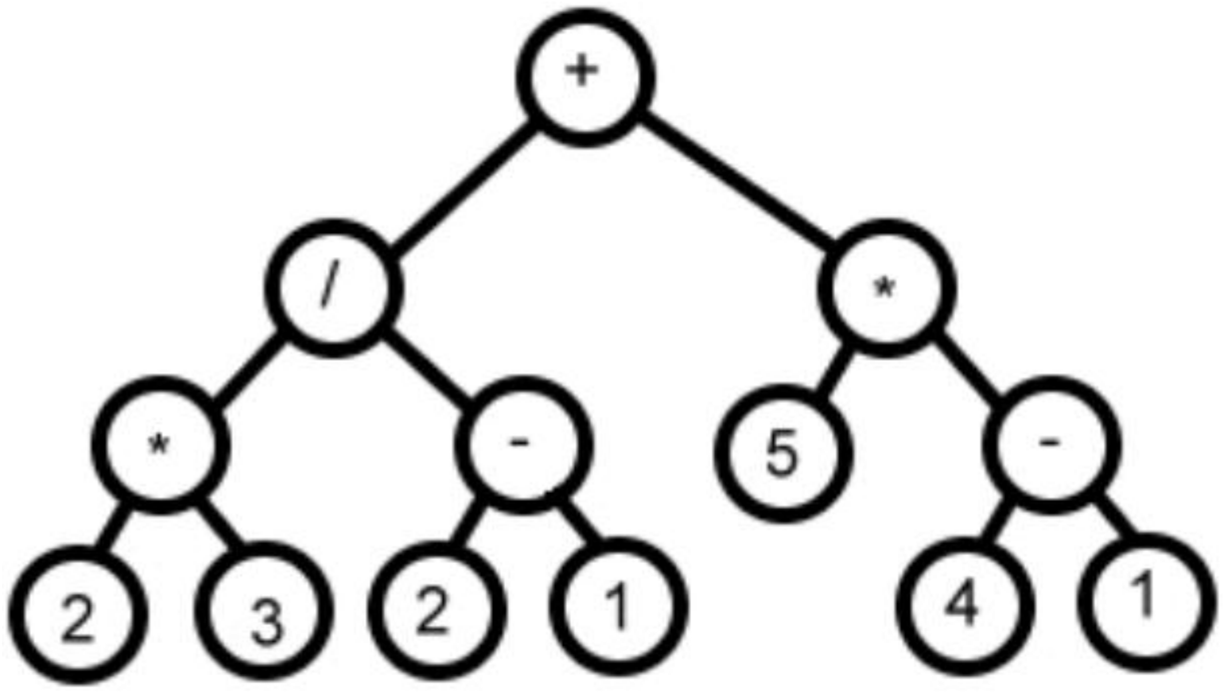
TASK 2:DYNAMIC DATA STRUCTURES

PART A:BINARY TREES(20 Marks)

The classes Node.java and BinaryTree.java comprise the starting point for a binary tree ADT that is to be used for this section.

Write a new class called Tester1 that contains the following methods:

a. **countA** (takes as input a BinaryTree and returns as result an integer; the result should be a number of nodes in the input tree which contain the character 'A'.)[4 marks]
b. **countLeaves** (takes as input a BinaryTree and returns as result an integer; the result is the number of leaves present in the tree.) [4 marks]
c. **postOrder (**takes as input a BinaryTree and returns as result a String; the String should contain the characters present in the nodes of the tree determined according to a post-order traversal of the tree.) [2 marks]
d. **inOrder (**takes as input a BinaryTree and returns as result a String; the String should contain the characters present in the nodes of the tree determined according to a in-order traversal of the tree.) [2 marks]
e. Finally, insert in the class a main method which constructs the tree shown in the figure below (by performing an appropriate sequence of insertNode, getLeft, and getRight operations) and then it calls all the four methods above, printing their output on the screen. [8 marks]



PART B: BINARY SEARCH TREES (30 marks)
Create a BST class that extends the Binary tree class by adding to the ADT the following operations:

1. An operation which returns as result the largest key present in the tree. [4 marks]
2. An operation which returns as result the smallest key present in the tree. [4 marks]
3. An operation that searches the tree for a specific value, and returns either the value NULL, or a reference to the tree that has the key being searched for as the root. [4 marks]
4. An operation that inserts a key into the correct place in the BST. [4 marks]
5. An operation that deletes a key from the BST. [8 marks]
6. Test the BST class by building another class (called Tester2) which contains main method.

   [6 marks]

The main method should allow the following actions to be performed:

1. Insert an item to the tree
2. Search the tree for an item
3. Print the minimum key in the tree
4. Print the maximum key in the tree
5. Remove an item from the tree
6. Perform an in-order traversal of the tree, printing the values of all nodes

PART C: RED-BLACK TREES (BONUS:25 marks) (Bonus Applies to this assessment only**)**
Create classes **RBNode**, **RBCluster** and **RBTree** that extend appropriate existing classes such that:

1. A **RBNode** has a colour which is either red or black. [2 marks]
2. A **RBCluster** is an aggregate of **RBNodes**, and has an integer attribute of **numNodes** and a boolean attribute named **critical**. [3 marks]
3. A **RBCluster** also has a method named **fixCluster,** which is fired if **critical** is true. [5 marks]
4. The **RBTree** has a method named **addNode**, which adds a unique key. [5 marks]
5. The **RBTree** also has a method named **deleteNode**, which removes a unique key. [10 marks]