ESO207 Programming Assignment 2 Semester II, 2016-17

Due Date: 23rd Feb, 2017

---

1. In class, we learnt about Binary Search Trees (BSTs) and defined nearly balanced BSTs and balanced BSTs. A *perfectly balanced BST* is a BST where for every node v the difference between the height of the left and right subtree of v is at most 1. A *nearly balanced BST* is a BST such that for every node v in the tree,

$$\max\{|\text{subtree}(\text{left}(v))|, |\text{subtree}(\text{right}(v))|\} \leq \frac{3}{4}|\text{subtree}(v)|.$$

In this assignment, you have to code up these variants, and compare their performances on random input sequences. You have to :

1. Implement a function used to insert a given value into the BST
2. Implement a search function for a given value
3. Checking perfectly balanced condition at a node
4. Checking nearly balanced condition at a node
5. Creating a BST from a sorted array (this must be done in $O(n)$ time, where n is the size of the array)
6. Creating a sorted array from a BST (this must also be done in $O(n)$ time, where n is the number of nodes in the BST)

You should strictly adhere to the following function prototypes for the above functions and the struct as provided in the attached *.cpp* file. In the comments, it is mentioned what that function is expected to do. You should only write code in the function body. You should not make use of any other functions. If you do not follow these guidelines, then the individual functions that you write may not be checked, and the code may receive zero score if it does not pass the test data.

You are recommended to verify whether your insert, balancing and search functions work correctly by working the operations out on short random sequences by hand.

Next you need to submit a report in which you compare the following on inserting random sequences of integers of length 10000:

1. A nearly balanced BST, in which you rebalance nodes whenever they are **not** nearly balanced.
2. A perfectly balanced BST, in which you rebalance nodes when they are **not** perfectly balanced.

Rebalancing a node means making the binary tree that is rooted at that node, perfectly balanced.

For the nearly balanced BST, we constrained the size of the larger subtree to be at most $\frac{3}{4}^{th}$ of the size of the complete tree. In this assignment, you should try the ratios $\frac{5}{6}$ and $\frac{7}{9}$ as well. You have to plot the time taken by all these algorithms versus the number of insert queries and explain your observations. The plots should be clean, and the points should be clearly marked. Any unclean or unmarked plots will lead to deduction of some of the alloted marks. You should also draw a single plot comparing all these algorithms. Explain your observations accordingly. You must give formal, theoretical arguments supporting your analysis and observations.