CS210A: Data Structure and Algorithms

Semester I, 2014-15, CSE, IIT Kanpur

Programming Assignment 2 (Deadline: 7:00 PM on 7th September)

Binary Search Trees

In this course, we invented the novel data structure called the Binary Search Tree (BST). Along with that, the concept of *perfectly-balanced* binary search trees and *nearly-balanced* binary search trees was also introduced. It was shown that in case of a *normal* BST, if the data is inserted in increasing/decreasing order, the tree becomes skewed, and the time to search an element in this case can be as bad as searching in a linked list. Moreover, it was claimed that if we maintain a *nearly-balanced* BST, then the total time taken to handle any sequence of n insertions will be $O(n \log n)$. The algorithm to maintain *nearly-balanced* BST was simple and intuitive (periodic rebalancing of the tree). However, its analysis was not at all obvious. Though, the theoretical analysis/proof for this claim is deferred towards the end of this course, this is the right time for you to verify this claim experimentally. In addition we also want to determine how bad will an ordinary BST perform on a real life application. These two goals will be achieved through the following two parts of this assignment:

1. Each group has to signup on `http://aca.cse.iitk.ac.in/cs210`, with login-id and team-name as <Roll No. 1> _ <Roll No. 2>, for example, 13001_13002 (**if you already have an account from the 1st assignment, use the same**).

   There are two problems on the judge, one corresponding to an ordinary BST and another corresponding to *nearly-balanced* BST. You have to code these two types of binary search trees with insertion and search operations. Submit the two codes and the online judge will check for their correctness and efficiency. You must strive for the most efficient implementation. Otherwise, your solution for *nearly balanced* BST will not be accepted by the judge for large input instances.

   Note: The input/output format and other details will be available on the judge by the evening of 28th August.

2. The second part of the assignment is to analyze the efficiency of an ordinary BST assuming that the elements to be inserted are generated randomly uniformly from a given domain. In most of the applications in real life, this assumption is indeed true. You have to find out how badly do ordinary BST perform on most-common input distribution.

   Write a program that builds an ordinary BST on n numbers generated radomly uniformly from interval $[0, 1]$. Let $T(n)$ be the actual real time needed to build the BST on a random input of size n. Let $h(n)$ denote the height of the resulting BST. You will have to empirically find out $T(n)$ and $h(n)$ and fill in the followig table.

n	$T(n)$	$T(n)/(n \log n)$	$h(n)$	$h(n)/\log n$
500				
2000				
10^4				
10^5				
10^6				
10^7				

   Write 4 most important inferences that you can draw from the data you fill in this table. This part of the assignment has to be submitted as a **one-page report**.