Indian Institute of Technology Mandi
August-November 2018 Semester

CS202 - Advanced Data Structures and Algorithms

Programming Assignment 2

Course Instructor : Aditya Nigam

TA : Sagar Gupta, Gaurav Purohit

18 September 2018

1 Instructions

- Plagiarism is strictly prohibited. In case of violation of the same zero marks will be rewarded for this assignment and strict action will be taken.
- You must **not use STL library** classes and functions.
- Submit a Makefile which compiles all your codes.
- Students using CodeBlocks or other compilers are requested to make sure their code runs perfectly on Linux as mentioned in each problem. Your evaluation will be on computers in PC lab.
- You are required to **use trees designed by you only** in this assignment for every problem even if a better (or simpler) solution exists using stack, queue, merging etc.

2 Implementing Tree Data Structures

Implement the following tree data structures using C++ programming language.

1. Binary Tree **(Marks-5)**
2. BST (BSTree.hpp) by inheriting the functions and node structure (BinaryNode class) of binary tree (BinaryTree.hpp). **(Marks-10)**
3. Implement AVL tree (AVL.hpp) by inheriting the BST (BSTree.hpp). **(Marks-15)**
4. Implement the RB tree (RBTree.hpp) by inheriting BST (BSTree.hpp) also inherit BinaryNode class by creating RBTNode class. **(Marks-20)**

Above marks will be provided only if below mentioned files are present.
Write a separate main programs to evaluate the functions in data structures binary tree, binary search tree (BST), balanced search trees such as AVL tree and red-black (RB) tree. The main functions should have the options to read inputs from user and display.

For the evaluation of the above trees below mentioned main.cpp is also necessary (otherwise your code won't be evaluated). A main file which takes an input $Q(1 <= Q <= 100000)$, number of queries. Next Q lines contain queries of the given form.

- 1 x (insert x into the tree)
- 2 x (delete one occurence of x from tree if present)
- 3 x (find x in tree. Print $1/0$)
- 4 (print value of maximum element in the tree)
- 5 (print value of minimum element in the tree)

Note:

1. x can be of any type - int, long long, double, string etc.
2. Compile the main.cpp to create an executable file (named *trees*) which runs in the below format.
   ./trees tree-name data-type
   We will run the code in given format
   ./trees bst string < **in.txt** > **out.txt**
   ./trees avl int < **in.txt** > **out.txt**
   ./trees rbt double < **in.txt** > **out.txt**
   out.txt will be matched against actual results file.
3. Do not change the class names. It is expected to strictly use the interfaces provided in the classes to implement the tasks. **(Marks-10)**

3 Problems

1. World is full of Trolls and Troll Hunters. There are **n** hunters numbered from 1 to n, each has a Strength coefficient associated with them. If a hunter **i** with strength coefficient **A[i]** is surrounded by Trolls, then a hunter **j** with strength coefficient **A[j]** such that **A[j]**>**A[i]** can protect him. The time taken by Hunter **i** to reach Hunter **j** is given by formula |**i-j**|. For every Troll Hunter, You have to calculate the minimum time after which the given troll hunter, could be saved from troll.
   Consider at a time only one troll hunter can be attacked. Attack on a Hunter is independent of each other. **(Marks-20)**
   Input
   Integer n $(1 <= n <= 100000)$ - number of Troll Hunters. Next line contains n space separated values i^{th} of them denoting the strength coefficient of hunter positioned at i^{th} index. $(1 <= A[i] <= 1e18)$
   Output
   n space separated values i^{th} one corresponding to the minimum time required so that i^{th}

Hunter can be saved from Troll. If some Hunter can't be saved print **-1**

2. *Older the wine the better it tastes.* One day Barney decided to throw a grand party to all his friends. He had **n** friends standing in a straight line and he gave each of them a bottle of wine. Each wine bottle has the year of manufacturing written on it. Every friend can see the manufacturing year of his own bottle and all the bottles ahead of him ie to his right (but not left). Friends being of jealous type, each of them wanted to know how many of the friends standing ahead got a better tasting wine. **(Marks-20)**

   Input

   Integer n ($1 <= n <= 100000$) - number of friends Barney invites. Next line contains n space separated values i^{th} of them denoting the year of manufacturing of the i^{th} bottle. ($1 <= A[i] <= 1e18$)

   Output

   n space separated values i^{th} one corresponding to the number of tastier wines ahead of it.