Indian Institute of Technology Kharagpur

CS29003: Algorithms Laboratory, Spring 2023

Assignment 2: Binary Search Trees

2PM – 5PM 17th January, 2023

General Instructions (to be followed strictly)

Submit a single C/C++ source file.
Do not use global variables unless you are explicitly instructed so.
Do not use Standard Template Library (STL) of C++.
Use proper indentation in your code and include comments.
Name your file as `<roll_no>_a1.<extn>`
Write your name, roll number, and assignment number at the beginning of your program.

Let T be a binary search tree with integer key values and let k, ℓ be 2 integers with $k \leq \ell$. The height of a node x in the binary search tree is defined as the length of the longest downward path from x to a leaf node. Heights of leaf nodes are 0. And the maximum height of the tree $h_{\max}$ is the height of the root node. Your task is to print all nodes/keys in T with height h with $k \leq h \leq \ell$.

(a) Define a data type to store a node of the BST. A node should contain the following: an integer key, two pointers – one to the left child, one to the right child and the height of the node (also an integer). The height of every node should be initialised to 0.

(b) Write a function *insert()* that takes as input an integer x and inserts it in BST T. The function should return a pointer to the root of the modified tree after insertion. Note that the function should not change the tree if x is already present in the tree.

(c) Write functions *preorder()* and *inorder()* to print the pre-order and in-order traversals of T. Use these in a function *print_tree()* that prints both traversals of T.

(d) Write a function *height()* that computes the height of every node in T. The function should run in $O(n)$ time where n is the number of nodes in T.
Hint: Use recursion and call *height()* on the root node, which will in turn update the height field in all nodes of T.

(e) Write a function *print_hrange()* that takes as input two integers k, ℓ and prints all nodes in T with height in the range $[k, \ell]$ in sorted order. The output should contain both key value and height of each node printed. This function should run in $O(n)$ time.

In all the functions defined above, T (pointer to the root of a BST) is provided as an input.

In the *main()* function, do the following.

- Initialise T to an empty BST.
- Read n, the number of keys to be inserted in T. Then read n integers $x_0, x_1, \ldots, x_{n-1}$, each to be inserted in T using the *insert()* function, with height field initialised to 0.
- Print T by calling the *print_tree* function.
- Run *height()* to compute heights of all nodes in T and print maximum height $h_{\max}$ (i.e., height of the root).
- Call *print_hrange* to print all nodes of T with height in the range $[k, \ell]$, along with the corresponding heights in the format (*key* : *height*) in sorted order of key. Assume that $k, \ell \in [0, h_{\max}]$.

Sample Output

```
n = 15

Enter keys to insert
   42 37 8 19 26 61 12 57 3 93 32 68 23 17 75

Traversals of the created BST
   Preorder  : 42 37 8 3 19 12 17 26 23 32 61 57 93 68 75
   Inorder   : 3 8 12 17 19 23 26 32 37 42 57 61 68 75 93

Heights of all nodes computed.
   Maximum height: 5

k = 1
l = 3

key:height for all nodes in the height range [1,3]
   (8:3),(12:1),(19:2),(26:1),(61:3),(68:1),(93:2)
```