CS6013: Advanced Data Structures and Algorithms IIT Hyderabad, Fall 2019

Programming Assignment 2 (Treaps)

- Out on: **29th Sept 2019**. Due Date: **11:59am, Thursday 10th Oct 2019**.
- The total marks for this assignment is 50.
- For submission and upload instructions, see the section after the questions.

Problem Description

In this assignment, you will implement a *randomized* data structure called a *Treap*. As the term implies, this data structure is the combination of a Binary Search Tree and a Heap.

> In a treap, every node has *two* fields: a key and a priority. A treap is a binary search tree with respect to the keys, and a *min-heap* with respect to the priorities.
>
> For every node u:
>
> 1. For any node v in the left subtree of u, $v.key < u.key$. For any node w in the right subtree, $w.key > u.key$.
> 2. (Min-heap) If x is a child of u, then $x.priority \geq u.priority$.

When inserting a key into a treap, we choose a *random* priority for that key, and insert it in a position so as to preserve the above min-heap property with respect to priorities, and BST property with respect to the keys. Due to the randomness of the priority value, and the min-heap property, the resulting tree structure is almost balanced or complete. This is similar to inserting the keys in a random order into a usual BST. We had mentioned in class, that if a binary search tree on n keys is built *randomly*, i.e. using a random ordering of the keys, then the expected height H of the resulting tree is: $\mathbb{E}[H] \leq O(\log n)$. This is why a treap is almost balanced.

Please refer to **Question 13-4** in CLRS for a complete description of the treap data structure. The question also describes how to insert a key into the treap. You do not need to submit solutions to the problems asked in that question for this assignment.

Operations to be implemented

You are to implement the methods given below for treaps in C++. You may assume that all keys inserted as input into the treap are distinct.

1. `Insert_key(r,k)`: inserts key k into the appropriate position in the treap with root node r. The key is assigned a random priority.

2. `Delete_key(r,x)`: here, x is a pointer/reference to the node to be deleted. Deletes the node x from the treap rooted at r.

3. `Search_key(r,k)`: search for a key k in the treap rooted at r

4. `Split(r, k)`: Split the Treap rooted at r into two treaps T_1 and T_2, where for any node $u \in T_1$, and $v \in T_2$, we have: $u.key < k < v.key$.

5. `Join(r1, r2)`: Given two treaps T_1, T_2 with roots `r1`, `r2` respectively, that satisfy: every key in T_1 $<$ any key in T_2. Merge the two treaps into a single treap.

   You may use the insert and delete methods in implementing split and join operations, or vice versa.

Besides the above, you are asked to implement a few more BST functions: successor, predecessor, finding number of keys less than a given key. Check the header file for declarations; they are self-explanatory.
Assumptions: The number of keys n stored in the treap will be at most 10^5. You may choose your random priority generator accordingly with this in mind.

Programming Instructions:
You should write your code in C++. The code for this assignment will be split across three files:

1. **Header File (`mytreap.h`):** This contains the function declarations for the treap data structure, and the definition of the node struct. This file is provided by us and should not be modified.

2. **Function implementation file (`your-roll-number-treap.cpp`):** This is the file where you will provide the definitions of the functions declared in (`mytreap.h`). This is the only file you submit that will be evaluated.

3. **Driver file (`main.cpp`:)** Contains methods to test your function implementations, including the `main()` function. We will use our own driver file to test your implementations; you may use your own. This file, even if submitted, will not be evaluated.

If you aren't familiar with the use of multiple files for C/C++, please take a look at the following video for a brief introduction: https://www.youtube.com/watch?v=IMuf781DAQc. Figure out how to compile the same on your machine. Programs will be compiled using `g++`.
Example: Provided is a simple implementation of a function `test_func` in the file "func.cpp". The file "main.cpp" then calls the function `test_func`. In order to compile this project, we will use:

```
g++ -Wall main.cpp func.cpp
```

This will produce one executable file (`a.exe` or `a.out` depending on your platform), which you can then execute from the commandline.

Submission Guidelines

1. Your submission will be one zip file (or `.tar.gz` file) named `<roll-number>.zip` (or `<roll-number>.tar.gz`), where you replace `roll-number` by your roll number (e.g. `cs19mtech11003.zip`), all in small letters. The compressed file should contain the below mentioned files:
   (a) The function implementation file with your code, named `roll-number-treap.cpp` (e.g. `cs19mtech11003-treap.cpp`).
   (b) The supplied header file `mytreap.h`.
   (c) A brief description of your solution in **pdf** format named `report.pdf`. This file should describe the algorithms you use for implementing the functions, and briefly state what each function in your code does.
2. The preferred way to write your report is using LaTeX(Latex typesetting). [Not mandatory].
3. **For the inorder print statement, follow the output format strictly. Do not print a single extra character space over what is needed.**
4. You **should not** use C++ STL libraries for this assignment.
5. Upload your submission to the Google classroom link. You may reupload multiple versions till the deadline; only the final version will be considered.
6. Failure to comply with the above instructions will result in your submission not being evaluated (and you being awarded 0 for the assignment).
7. **Late submission policy**: For every day (24 hours) late: -10 from your score on the assignment.
8. **Plagiarism policy:** If we find a case of plagiarism in your assignment (i.e. copying of code, either from the internet, or from each other, in part or whole), you will be awarded a **zero**. Note that we will not distinguish between a person who has copied, or has allowed his/her code to be copied; both will be equally awarded a zero for the submission.

1 Evaluation

You will be marked for the following aspects:

1. Correctness and Efficiency of the implementation.
2. Code clarity and comments.
3. Report presentation.