Bilkent University
Computer Engineering Department
CS 202
Spring 2018

Homework #3 – Heaps and AVL Trees

Important notes

- Before 23:55, April 18th, upload your solutions in a single **ZIP** archive using the Moodle submission form. Name the file as **`studentId_hw3.zip`**.
- Your **ZIP** archive should contain the following files:
    - **`hw3.pdf`**, the file containing the answers to Questions 1 and 2
    - A folder named **q2** which includes **`AVLTree.h`**, **`AVLTree.cpp`**, **`analysis.h`**, **`analysis.cpp`** files and **`main.cpp`** files containing the C++ source codes and **`Makefile`** which compiles all your code and creates an executable file named **q2**.
    - A folder named **q3** which includes **`MaxHeap.h`**, **`MaxHeap.cpp`** and **`main.cpp`** files containing the C++ source codes and **`Makefile`** which compiles all your code and creates an executable file named **q3**.
    - **`readme.txt`**, the file containing anything important on the compilation and the execution of your program in Question 2 and 3.
- Do not forget to put your name, student ID, and section number in all of these files. Well comment your implementation. Add a header as the following to the beginning of each file:

```
/*
* Title : Heaps and AVL Trees
* Author : Name Surname
* ID : 21000000
* Section : 0
* Assignment : 3
* Description : description of your code
*/
```

- Do not put any unnecessary files such as the auxiliary files generated from your favorite IDE. Be careful to avoid using any OS dependent utilities.
- You should prepare the answers of Questions 1 and 3 using a word processor (in other words, do not submit images of handwritten answers).

- Although you may use any platform or any operating system to implement your algorithms and obtain your experimental results, your code should work on the **dijkstra** server (***`dijkstra.ug.bcc.bilkent.edu.tr`***). We will compile and test your programs on that server. Thus, you may lose significant amount of points if your C++ code does not compile or execute on the dijkstra server.
- This homework will be graded by your TA, Leonard Dervishi. Thus, please contact him directly for any homework related questions.

Attention: For this assignment, you are allowed to use the codes given in our textbook and/or our lecture slides. However, you ARE NOT ALLOWED to use any codes from other sources (including the codes given in other textbooks, found on the Internet, belonging to your classmates, etc.). Furthermore, you ARE NOT ALLOWED to use any data structure or algorithm related function from the C++ standard template library (STL).

Do not forget that plagiarism and cheating will be heavily punished. Please do the homework yourself.

Question 1 - 25 points

(a) [5 points] Draw all valid **min-heaps** containing these 4 elements 5, 7, 6, 1.

(b) [5 points] How many valid **max-heaps** containing 5 distinct elements can be built?

(c) [10 points] This question has three subparts which follow one another.

i. Insert 46, 59, 51, 31, 68, 63, 25, 75, 40 to an empty AVL tree, in the given order. Show **only the final tree** after all insertions.

ii. Add one more element such that it causes a **single right rotation** in the tree. Show the final tree after the insertion.

iii. Insert 1; to the AVL tree. Now, delete one element such that it causes a **single left rotation** in the tree. State the deleted number and show the final tree after the deletion operation.

(d) [5 points] ***True or false***: "Insertion order of the same set of elements into an AVL tree affects the resulting tree structure".
If yes, why; if not, provide a counter-example.

Question 2 - 25 points

Implement AVL tree data structure named as AVLTree for maintaining a list of integer keys with the following methods:

`void insert(int value)`; //inserts an element into the tree

`int getNumberOfRotations()`; //returns the number of rotations performed so far while constructing the AVL tree

After that you will analyze the average number of rotations and determine if different patterns of insertion affect it. Write a global function, `void rotationAnalysis()` which does the following:

(a) Creates 1000 random numbers in the range [1,100000] and inserts them into an empty AVL tree. While inserting elements into the AVL tree, it counts the number of rotations and it outputs the total number of rotations in the end. Repeat the experiment for the following sizes: {2000, 3000, 4000,..., 10000}

(b) Instead of creating arrays of random integers, create arrays with elements in the same range in ascending order and repeat the steps in **part (a)**.

(c) Instead of creating arrays of random integers, create arrays with elements in the same range in descending order and repeat the steps in **part (a)**.

Put function's code into `analysis.h` and `analysis.cpp` files. Create a `main.cpp` file which calls `rotationAnalysis()` function. When the `rotationAnalysis()` function is called, it needs to produce an output similar to the following one:

Array Size	Random	Ascending	Descending
1000	x	x	x
2000	x	x	x
3000	x	x	x
...	x	x	x

After running your program, you are expected to prepare a report about the experimental results that you obtained in Question 2. In your report, you need to discuss the following points:

- Do your findings related to average number of rotations in the AVL tree agree with theoretical results?
- Do different patterns of insertion affect the number of rotations in the AVL tree? If so, explain how. If not, explain why not.

Question 3 - 50 points

Caezar, the leader of the Planet of the Apes, gained his intelligence from the ALZ - 112 drug and since then he has been learning computer science and hacking. A few days ago he visited Bilkent and hacked into your computers while you were practicing heaps. He takes the leaves of the heap that you have built from a given file and permutes them by picking a permutation uniformly at random. If the heap properties are not satisfied after the permutation, you need to fix the heap with appropriate methods. He also offers you a deal where he will either pay 10TL for each swap operation that you perform in the course of fixing the heap, or a 20 TL ALL worth treat (independently of the count). You need to write a program for a given file containing integers advising you about the best deal for you.

You need to implement a **`MaxHeap`** class which includes its appropriate methods and `permutation(…)` method. In **`main.cpp`** class you should take the input from a file named **“input.txt”** and build the heap by using the implemented methods. After that you need to generate every permutation of the leaves and sum up the total number of swaps that is done for each case. Then you need to compute the average number of swaps and choose the betterdeal accordingly.

Sample input:

```
15
8299 49586 21674 17523 6216 13185 47201 22952 48878 44192 5868 34529 40470 5003 18703
```

Note: The first line of input contains N - the number of nodes and the second line contains N unique integers from which a max heap is to be built.

Sample output:

```
The total number of swaps is 20160.
The average number of swaps per permutation is 0.5.
Taking 20 TL is a better option.
```

Note: You can print “10 TL per swap is a better option” for the other case.