CS 202, Summer 2023

Homework 3 – Heaps and AVL Trees

Due: 23:59, August 1, 2023

Important Notes

Please do not start the assignment before reading these notes.

1. Before 23:59, August 1, upload your solutions in a single **ZIP** archive using Moodle submission form. Name the file as studentID_secNo_hw3.zip.
2. Your ZIP archive should contain the following files:

   a. **`hw3.pdf`**, the file containing the answers to Questions 1 and 2. This **pdf** must also include the report of the programming part.

      - You should prepare and upload **handwritten** answers for Question 1 (in other words, do not submit answers prepared using a word processor).
      - Use the exact algorithms shown in lectures.

   b. **`heap.cpp`**, **`heap.h`**, **`heapsort.cpp`** files which contain the C++ source codes, and the **`Makefile`**.

   You have to follow the following instructions about the format, programming style and general layout of your program.

   - You can use the codes provided in your textbook or the lecture slides. However, you cannot use any external heap implementation such as STL's in your code.
   - 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 given below to the beginning of each file:

```
/*
 * Title: Heaps and AVL Tree
 * Author: Name Surname
 * ID: 21000000
 * Section: 1
 * Assignment: 1
 * Description: description of your code
 */
```

3. This homework will be graded by your TA, Cihan Erkan. Thus, please contact him directly (cihan.erkan at bilkent edu tr) for any homework related questions. There will also be a forum on Moodle for questions.

- 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. If your C++ code does not compile or execute in that server, you will lose points.

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.

1) Question Part (50 points)

a) (10 points) Show the result of inserting 10, 3, 1, 8, 2, 13, 7, 4, 5 and 9 in that order into an initially empty AVL tree. Show the tree after each insertion, clearly labeling which tree is which.

b) (10 points) This problem gives you some practice with the basic operations on binary min heaps. Make sure to check your work.

- Starting with an empty binary min heap, show the result of inserting, in the following order, 5, 1, 6, 20, 2, 4, 16, 25, 30, 13 and 8, one at a time, into the heap. By show, we mean, "draw the tree resulting from each insert."
- Now perform two deleteMin operations on the binary min heap you constructed in part (a). Show the binary min heaps that result from these successive deletions, again by drawing the binary tree resulting from each delete.

c) (10 points) Consider a binary heap. Print the keys as encountered in preorder traversal. Is the output sorted? Justify your answer. Attempt the same question for inorder and postorder traversals.

d) (10 points)

- Give a precise expression for the minimum number of nodes in an AVL tree of height h.
- What is the minimum number of nodes in an AVL tree of height 15?

e) (10 Points) Write a function in pseudocode that determines whether a given binary tree is a min heap.

2) Programming Part (50 points)

In this assignment, you will write a Heap class and use it to implement HeapSort. The Heap class must be defined in two files: heap.cpp and heap.h. The Heap class must support, at least, the following functions.

```
void insert(const int a)
int maximum()
int popMaximum()
```

Your program will read input from an input file which contains one integer per line, and write sorted output to an output file which contains the same integers in sorted order. Moreover, your program should write the number of comparisons made during the sorting function to standard output.

Deliverables:

- A heap.cpp and heap.h file which implement the Heap data structure.
- A heapsort.cpp file which uses the Heap data structure to implement the heapsort algorithm
- A report describing the heap data structure and heapsort functions including the theoretical and actual number of comparisons required for each algorithm.
- Run your program on five sample input files. In your report, report the number of data points, n, in each file, as well as the number of comparisons heapsort uses to sort them. Please remember that your program will be tested with additional input files.

The program must compile using the following command on dijkstra.cs.bilkent.edu.tr

```
g++ heapsort.cpp heap.cpp -o heapsort
```

The program must run using the following command on dijkstra.

```
./heapsort input_filename output_filename
```