CS 202 Fundamental Structures of Computer Science II

Assignment 3 – Heaps and AVL Trees

Date Assigned: November 6, 2015
Due Date: November 21 23:55 (sharp), 2015

1) Question Part (40 points)

a) (10 points) Show the result of inserting 13, 8, 5, 9, 4, 6, 12, 2, 1 and 3 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, 12, 9, 3, 8, 5, 6, 14, 1, 7, 11 and 2, 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 a CS network computer.

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

The program must run using the following command on a CS network computer.

./heapsort input_filename output_filename

Code Format and Notifications

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 text book or the lecture slides. However, you cannot use any external heap implementation such as STL's in your code.
- Don't forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of every file that you are submitting.
- Don't forget to write comments at important parts of your code.
- You are free to write your programs in any environment (you may use either Linux or Windows). On the other hand, we will test your programs in a Linux environment and we will expect your programs to compile and run on Linux. If we cannot get your program work properly on Linux, you will lose a considerable amount of points. Therefore, we recommend you to make sure that your program compiles and properly works on Linux before submitting your assignment.
- This assignment is due by 23:55 (sharp) on ~~Friday~~ **Saturday**, **November 21**, 2015. You should upload your solutions to Moodle. You should upload a **single zip** file that contains:
    a. Your solution to first part as **pdf** (do NOT send photos of your solution), this **pdf** must also include the report of the programming part.
    b. Code files (only the "**.cpp**" and "**.h**" files that you write for the homework) of the second part and a "**Makefile**" for the compilation of your code that produces the executable.
    c. In the end, your zip file should contain ONLY **code files**, "**Makefile**" and a **pdf**; any violation of these causes a significant loss from your grade. Name of the pdf should be "***Section_ID_SurnameName.pdf***".
- If you do not know how to write a Makefile, you can find lots of tutorials on the Internet. Basically they are files that contain a list of rules for building an executable that is used by "make" utility of Unix/Linux environments. "make" command should build an executable called "**heapsort**", so write your Makefile accordingly (i.e. at the end, when you type "make" in terminal on your working directory, it should produce "**heapsort**" executable)
- Late submissions will not be graded.
- This homework will be graded by your TA Gündüz Vehbi Demirci (gunduz.demirci@bilkent.edu.tr). You may contact him for further questions.

DO THE HOMEWORK YOURSELF. PLAGIARISM AND CHEATING ARE HEAVILY PUNISHED!!!