Project 2

CS 241 – 02

Minwoo Soh

[msoh@cpp.edu](mailto:msoh@cpp.edu)

Tuesday, February 13, 2018

Section 1. Project Description:

In this project, you are going to build a max-heap. You will use an array to implement the heap. Your program should:

• Allow the user to select one of the following two choices (Note that your program needs to implement both choices):

o (1) test your program with 100 randomly generated integers (no duplicates, positive numbers with proper range);

o (2) test your program with the following 100 fixed values from 1, 2, 3, ..., and 100.

• Implement both methods of building a max heap:

o Using sequential insertions (its time complexity: 𝑂(𝑛𝑙𝑜𝑔𝑛)).

o Using the optimal method (its time complexity: 𝑂(𝑛)).

For both methods, you need to keep track of how many swaps (swapping parent and child) are required to build a heap.

• For choice (1), you need to generate 20 sets of randomly generated integers; compute, print and document (in your project report) the average number of swaps for both methods. Your program should output the average number of swaps for both methods (an average over 20 sets).

• For choice (2), your program should output the first 10 integers in your array and the number of swaps for both methods. Then perform 10 removals on the heap and output the first 10 integers.

In your project report, you need to analyze both methods of heap implementation in terms of their efficiency theoretically and experimentally.

Section 2. Project Specification:

The project required us to create a Max Heap Tree and test out the sorting algorithm through trials. All default methods such as add, removeMax, and reheap were included in the MaxHeap class. Few other methods like the firstTen and getOptimalCount had to be included to test out the specifications of this project. The method firstTen was used to get the first ten entries in the Max Heap’s array, and the getOptimalCount method was used to get the total number of swaps when using the optimal method of adding. A main program, named Project2, was made to test out the heap algorithm and check if the time complexities of sequential insertions and optimal insertion method was correct. The program allowed us to see the average number of swaps for both methods when using randomly generated arrays or a fixed array with predetermined values.

Section 3. Testing Methodology:

Testing to see if the code worked correctly came down to run the program repeatedly and determine if the results were acceptable. Besides choosing between two testing options, this program did not require any user input that would directly affect the data in the array. Therefore, the user must rerun the program and get a randomly generated array for testing choice 1. Even with the randomly generated array, the program returned consistent results. Results for sequential insertion were normally between 108 and 112, and results for optimal method were around 68 and 71. The average number of swaps stayed nearly identical even after changing the range of the randomly generated numbers. Testing out choice 2 was much easier since it was using a fixed array between 1 and 100. The results for this option perfectly matched the results of the example provided in the project instructions.

Section 4. Lesson Learned:

Testing out the program and analyzing the number of swaps helped me to understand how the time complexity of a Max Heap Tree was determined. I initially believed that changing the range of randomly generated numbers would affect the outcome, but after testing it out I realized that was not the case. However, changing the number of entries showed varied results for number of swaps. It was evident that more entries led to a higher number of swaps. I concluded that the time complexity of sorting a Max Heap Tree was influenced by the number of entries/nodes.

Section 5. Analysis of Output:

After fixing all errors and making sure the program did not crash on any occasion, every input and commands ran correctly. The program was tested multiple times with different ranges and different sizes of trees. Every result for choice 1 showed consistent results with barely any deviations. Results for choice 2 were perfectly identical to the example shown in the project description. The output of the results is included within the file in output.txt.