Meetkumar Patel

Professor Hao Ji

CS 2400.01

19 November 2018

Java Project 4 Report

Sample output:

==================================================================

CS 2400 Fall 2018 Java Project 4

Please select how to test the program:

(1) 20 sets of 100 randomly generated integers

(2) Fixed integer values 1-100

Enter choice: 1

Average swaps for series of insertions: 108

Average swaps for optimal method: 69

==================================================================

CS 2400 Fall 2018 Java Project 4

Please select how to test the program:

(1) 20 sets of 100 randomly generated integers

(2) Fixed integer values 1-100

Enter choice: 2

Heap built using series of insertions: 100,94,99,77,93,98,61,68,76,84,...

Number of swaps: 480

Heap after 10 removals: 90,89,62,77,88,53,61,68,76,84,...

Heap built using optimal method: 100,95,99,79,94,98,63,71,78,87,...

Number of swaps: 96

Heap after 10 removals: 90,89,63,79,88,55,62,71,78,87,...

Note: java interface and generic data type are used!

Methods: checkInitialization(), checkCapacity(), getMax(), isEmpty(), getSize(), clear(), toArray(), getInsertionSwap(), getSmartSwap(), ensureCapacity(), removeMax() are all self-explanatory. They do as their name implies.

For the sequential and smart methods:

We are working with array implementation and are creating a complete binary tree. For sequential method, adding an entry requires an up-heap operation, while removing the root requires down-heap operation. The index of the parent is the index of the entry divided by two. The maximum number of swaps that can occur is equivalent to the height. In the worst case, the method follows the path from leaf to root. So, sequential method’s time complexity to create a heap is O(n\*log(n)).

We are working with array implementation and are creating a complete binary tree. For smart method, we simply add all the entries in the array at the beginning. Then, we begin reheap operation beginning the last non-leaf index and go till the root to make a heap. So, smart method’s time complexity to create a heap is O(n).

Since sequential method’s time complexity is greater than smart method, it’s logical that the number of insertion swaps will be greater than the number of smart swaps. This can be verified by looking at the sample output. The number of insertion swaps is always greater than the number of smart swaps. This hold true for both cases: first and second option.