Project 2

Heap

CS241-01

Junda Lou

[lou@cpp.edu](mailto:lou@cpp.edu), [loudakkk@hotmail.com](mailto:loudakkk@hotmail.com)

2.13.2017

Project Description

In this project, we are asked to implement a heap using arrays. By forming a heap using a series of entries, we have to implement in two ways, serial insertions and optimal. This can show us which way is efficient. In the user interface, users will choose which way to test, which are 20 sets of randomly generated integers, or 100 fixed integers from 1 to 100. Number of swaps for two ways will be shown so that it is easy to see the difference. For the fixed integers, we have to print out the first 10 elements in the heap, and do it again after 10 removals.

Project Specification

The optimal way is putting everything into a heap first, and perform reheap() on all non-leaf nodes. When calling add() method, we have to check if the newEntry is at where it should at, and perform up-heap if necessary. This is required 100 times if we add 100 integers into the initially empty heap. However, using reheap() has a run-time efficiency of O(log(n)). It is performed only on non-leaf nodes.

Testing methodology

Testing is easier for this project compared to the BST implementation. We only need to test it with random integers and fixed integers from 1 to 100. Because a sorted list of integers in ascending order is the worst case for a Max-Heap, it becomes a necessary test case here. In order to make the test case general, we use 20 sets of random integers, and take the average result from them to reduce errors. During the process, I encountered NullPointerException and Cast problems. I found that I have to be careful with the scope. If I declare the input array inside the loop, and I call insertion() and optimal() outside the loop, the entry array will be passed all “null”s in its cells. Secondly, I had “[Ljava.lang.Comparable] cannot be casted into [java.lang.Integer]” complier error when I tried to pull out the heap array from the implementation class. I then looked up what is going on on the internet and decided to return a Comparable[] directly back to where it is called.

Lessons Learned

After comparing the results from insertion() and optimal(), I realize how important run-time efficiency is. I tested random integers a couple times and sometimes the swaps is huge compared to optimal(). As a programmer, we have to save time and space as best as we can do. It really make a huge difference. I also have to be careful of details in the code. I had NullPointerException in this project because I did not notice the scope of my declaration, and it took a long time for me to locate this problem. Now I had a “lesson” with it, I will be careful with it next time.