Project 2

Heap

CS 241

Jonathan Allen Grant

[jonathanallengrant@gmail.com](mailto:jonathanallengrant@gmail.com)

due May 9th, 2014

Section1. 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: (1) test your program with 100 randomly generated integers (no duplicates, positive numbers with proper range); (2) test your program with the following 100 fixed values from 1, 2, 3, ..., and 100. Note that your program needs to implement both choices.
* Implement both methods of building a max heap: (1) with sequential insertions and (2) with the optimal method.
* 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.

Section 2. Project Specifications:

The requirements for this project ordered my code to be written in Java, which it is. The user guided menu was created with a graphical interface, built using JFrame. I allowed the user to use checkboxes instead of binary keyboard input (i.e. inputting “1” or “0”). As for the heap implementation, I used recursive methodology to ensure the reheapification process. However, because Optimal Method requires reheaping downwards while sequential insertion requires reheaping upwards, so I made two different methods, which was a very challenging and difficult puzzle. I had global variables instantiated for the swap counters. When I printed them out for option 1, I divided the number by 20 to average the counts per attempt. For visually showing choice one, I created another graphical interface using my own buttons and fonts to enhance the experience. The numbers fly into the screen in a smooth way. For both options, I implemented the heap using an array. More specifications are in my comments of the Project2.java file.

Section 3. Testing Methodology:

As far as testing the heap implementation, I had to draw on paper several heaps, and implement them with both optimal method and sequential insertion. I then compared the drawings with my output, until I reached perfection. The heap testing was simple and easy.

The graphics testing was not. I wanted the interface to be absolutely seamless, and so I had to test after inserting every element. The biggest problem I faced was getting the title movie to play. At first, designed the video to be an avi type, with sound included. However, Java does not support playing avi’s easily, so I transformed the video into an animated GIF, and extracted the song I made in Garageband. I then loaded the GIF and played the sound in two different classes. After working on Stack Overflow and getting way too angry at my code for not doing exactly what it is programmed to do, I overcame frustration and implemented beauty.

Section 4. Lessons Learned:

I learned how to implement GIF’s in java, which is extremely important to me, because it allows title screens like mine that welcome the user in a memorable way. I also learned how to use custom fonts.

Section 5. Analysis of Output

According to my output, the optimal method works with less swaps every single time. I’ve ran my program hundreds of time, with a mean difference between amount of swaps for optimal and sequential to be forty. Experimentally, Optimal method is much better. Theoretically, Optimal method is better. Optimal method actually has thought applied to it, whereas Sequential insertion just throws numbers into an array.