**CSCE 221-200: Honors Data Structures and Algorithms  
Assignment Cover Page  
Spring 2021**

|  |  |
| --- | --- |
| **Name:** | Priyanshu Barnwal |
| **Email:** | [priyanshu@tamu.edu](mailto:priyanshu@tamu.edu) |
| **Assignment:** | PA 3 |
| **Grade (filled in by grader):** |  |

Please list below all sources (people, books, webpages, etc) consulted regarding this assignment (use the back if necessary):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CSCE 221 Students** | **Other People** | **Printed Material** | **Web Material (give URL)** | **Other Sources** |
| 1. | 1. | 1.Textbook | 1. | 1. |
| 2. | 2. | 2. | 2. | 2. |
| 3. | 3. | 3. | 3. | 3. |
| 4. | 4. | 4. | 4. | 4. |
| 5. | 5. | 5. | 5. | 5. |

Recall that TAMU Student Rules define academic misconduct to include acquiring answers from any unauthorized source, working with another person when not specifically permitted, observing the work of other students during any exam, providing answers when not specifically authorized to do so, informing any person of the contents of an exam prior to the exam, and failing to credit sources used. *Disciplinary actions range from grade penalty to expulsion.*

"On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work. In particular, I certify that I have listed above all the sources that I consulted regarding this assignment, and that I have not received or given any assistance that is contrary to the letter or the spirit of the collaboration guidelines for this assignment."

|  |  |
| --- | --- |
| **Signature:** | Priyanshu Barnwal |
| **Date:** | 3/18/2021 |

**Introduction:**

The objective of this assignment is to learn and build binary heaps using different methods to handle restructuring. I will be inserting numbers in these heaps and analyzing the runtimes for restructuring.

**Theoretical Analysis:**

As given by the textbook, the quicksort’s average runtime is O(Nlog(N)), and I have to assume that it is the same for both the random pivot and the first value pivot, because the textbook doesn’t give any values otherwise. The heapsort has a 2N log(N) − O(N log(log(N))) runtime. This can be simplified to 2N log(N) − O(N). I’m not exactly sure how to analyze this runtime, because it is incredibly complicated, however, since it’s double the other runtime minus O(N), we can see that it is actually slower. Since Nlog(N) is more than O(N), subtracting O(Nlog(N)) from 2 Nlog(N) is actually faster than just subtracting O(N). Thus, the quicksort should be slightly faster than the heap-sort.

**Experimental Setup:**

I am currently running this on a computer with 32GB RAM, however, Visual Studio only allocates 2 GB of processing memory for any given project. I am using Visual Studio 2019, which runs C++17 with experimental features of C++20. For my timing mechanism I used the high-resolution clock inbuilt function to accurately time how fast my program ran. I only ran each tree for one trial, but ran them 3 times by inserting increasing, decreasing, and random values.

**Experimental Results:**

As can be seen by the results in the graph, all three types of input insertions increase linearly with runtime (on a log-log graph). This means that all three of the runtimes have a linear runtime. Heap sort does the best by a small margin in all three situations, especially as numbers increase. Quicksort with a random pivot is by far the slowest in all cases, even though it still has a linear runtime. Unfortunately, the quicksort is supposed to slightly faster than the heap sort. This error could either be caused by a problem in the code, which makes it less efficient than possible, or it could be caused by the very small sample size.