Zack Aidarov

7 August 2023

Heap Sort Study Report

The Heap Sort Assignment included the creation of a class for a binary heap of integers in order to use it for a sorting algorithm. The project included the calculation-based method of story binary heap in an array rather than as a linked data structure. This project's premise is to compare two kinds of sorts, the predisposition Merge Sort, versus the Heap Sort that was designed by me.

My hypothesis to the experiment is that the Heap Sort method will take longer to process compared to the Merge Sort. The reason as to why I believe that Merge Sort will beat out Heap Sort is because Merge Sort tends to have a great boost in effect when paired with a fast system, therefore beating out its disadvantages with merge sort dividing an array into two. In previous experiments during the semester, we have proven that Quick Sort is the quickest way of sorting this particular algorithm when faced with a low number of numbers it needs to generate. However, with tests we've also proved that Merge Sort combined with good hardware makes a better match for larger arrays, as it sorts faster. Considering the previous information, I think it has a better chance of handling the task at hand compared to the Heap Sort.

When I ran my tests, I was surprised to see that Heap Sort wasn't too behind Merge Sort when it comes to sorting algorithms. In my finds I've found that on average, Merge Sort (0.078642 seconds) was 0.013266 seconds faster than Heap Sort (0.091908 seconds). With my findings I've also found that Merge Sort had beat out Heap Sort in the best and the worst runs it had.

99	0.079413	0.092055
100	0.077318	0.090597
	Merge	Heap
MINIMUM	0.075919	0.088454
MAXIMUM	0.097543	0.110138
AVERAGE	0.078642	0.091908

In conclusion, my hypothesis was proven correctly, as by using the previous experiments, I was able to deduct that Merge Sort tended to work better when it came to large arrays, as well as both kinds of sorts having a similar amount of complexity to them.