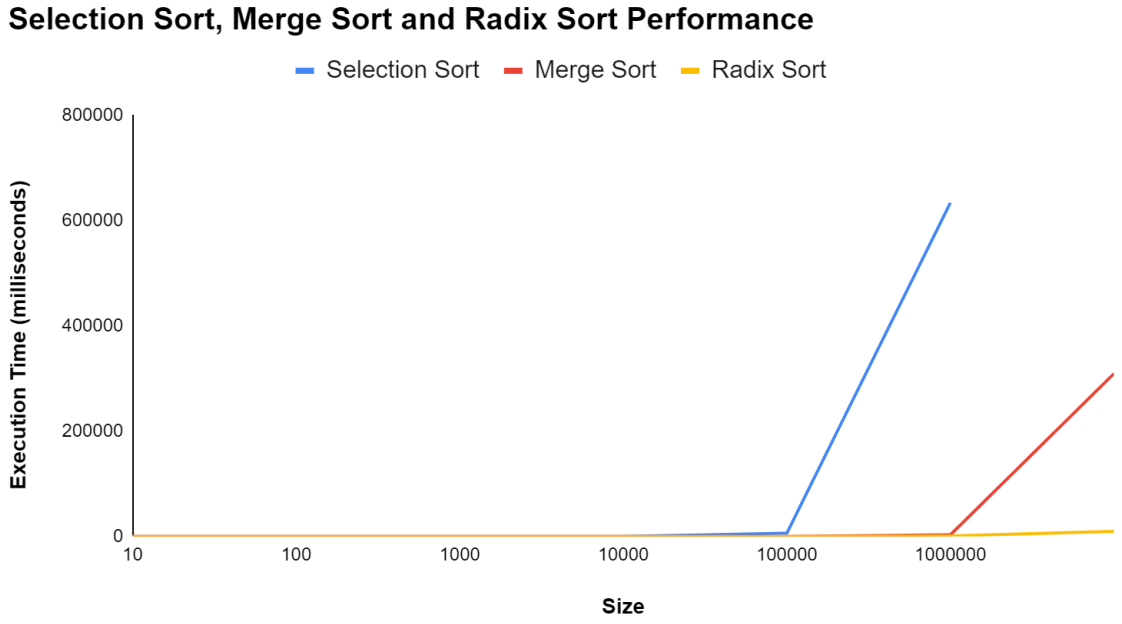
Zaid Abujumaiza
Professor Itani
CS288-005
11/6/2022



The performance of these sorting algorithms were tested by sorting a data set of 10 integers up to 10 million. Using C, selection, merge and radix sort were implemented and the integer data set was allocated on the heap to make the algorithm as fast as possible. Based on the data captured above, a graph can be plotted to show the performance of each algorithm as the size of the integer data set increases. The graph shows the execution time in milliseconds on the y axis and the size of the data set on the x axis using a logarithmic scale of base 10. The legends have colors to correspond with the sorting algorithm. Selection sort had to be stopped at 1 million due to memory limitations but merge and radix sort were tested up to 10 million.

Based on the graph, conclusions can be made about the complexity of each sorting algorithm. For selection sort on the blue line, once the size starts increasing, the execution time increases quadratically. The line drawn by the runs of selection sort depicts the same line as O(n^2) on the Big-O complexity chart which proves that it has a time complexity of O(n^2). For merge sort on the red line, once the size starts increasing, the execution time increases significantly but not as drastically as selection sort. The line drawn by the runs of merge sort depicts the same line as O(n log n) on the Big-O complexity chart which proves that it has a time complexity of O(n log n). For radix sort, it can be seen that the yellow line seems to have almost a straight line but it is slightly slanted upwards. This means that as the size increases, the execution time will get linearly increased. This linear line depicts an O(n) time complexity for radix sort because it is increasing linearly similar to O(n) on the Big-O complexity chart. It can be concluded that the performance of these sorting algorithms from best to worst is radix sort, merge sort and selection sort, respectively.