Alex Sadler

EECE 2560 – Homework 4

Problem 1

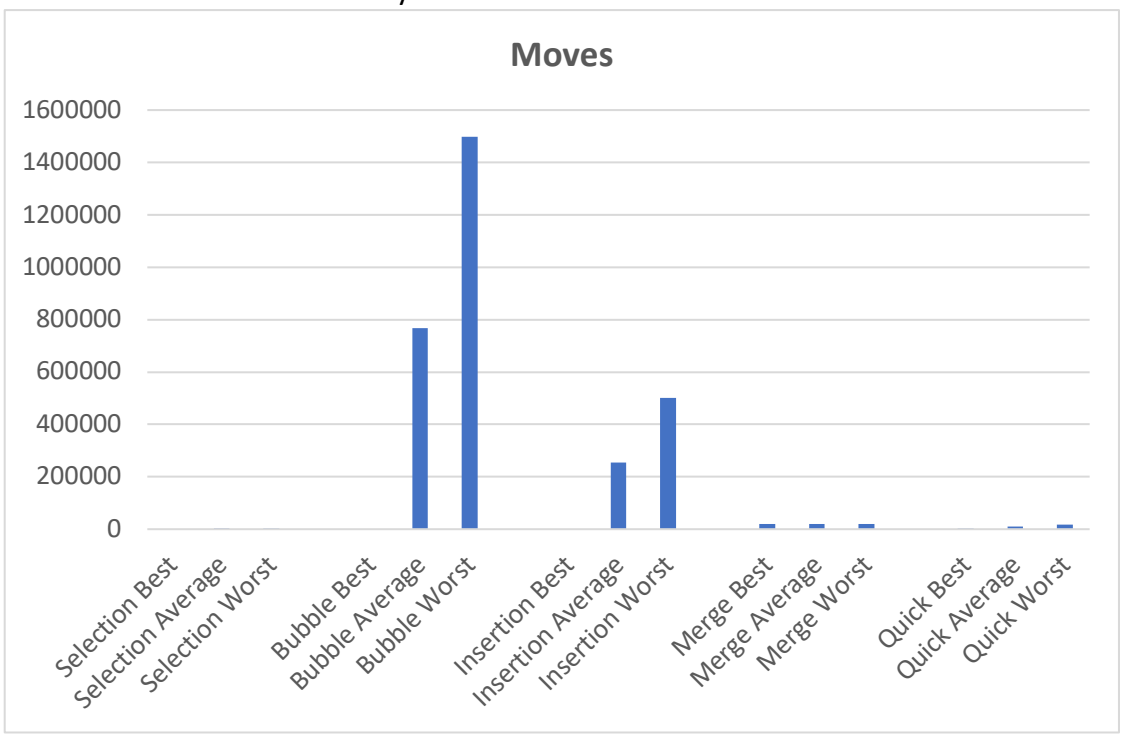
i. The selection, bubble, and insertion sort algorithms all only make moves/swaps when elements are out of order. Thus, when each of these algorithms is to "sort" an already sorted list, there are no moves since each element is already in its correct position.

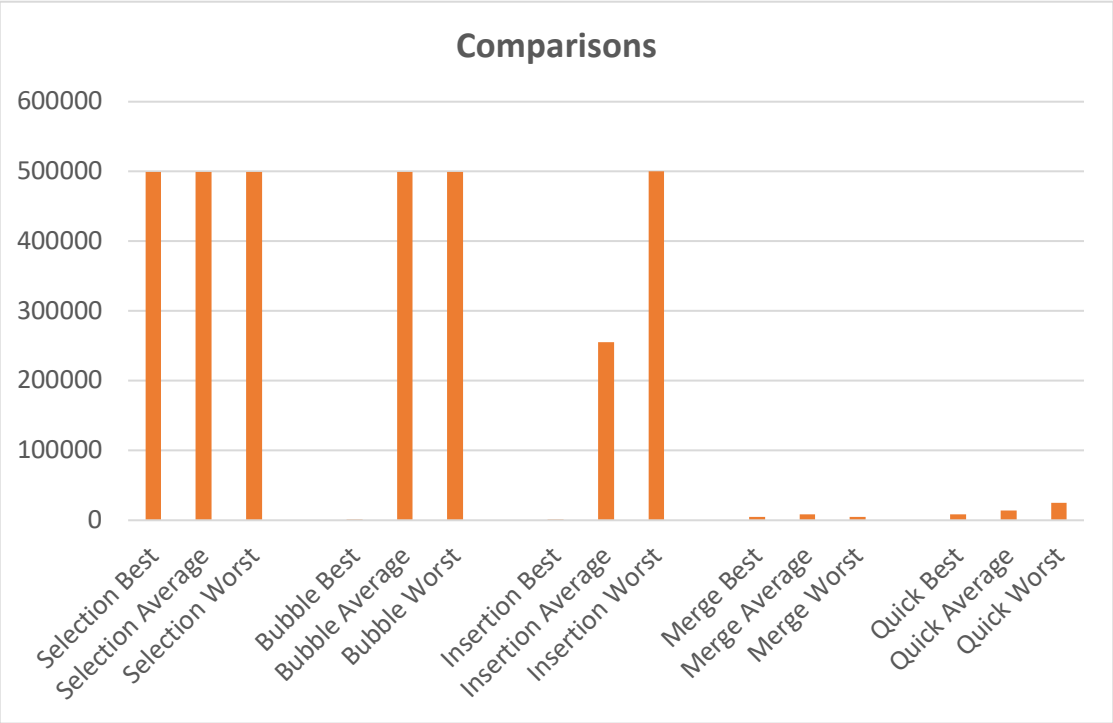
ii. Passing an already sorted list to the bubble sort function results in 999 comparisons since the algorithm only has to make one pass through the list. Since no swaps will be made, the algorithm only needs to compare all the numbers one time, which will be the length of the list minus one. Similarly, for insertion sort, for each iteration of the algorithm, each element is already in place, so each element is only compared once to the element before it.

iii. For the worst case for selection sort, there are 1500 moves since starting from the first element, the algorithm swaps it with the last, so both of these are now in place. This continues until reaching the halfway point of the list, and now the entire list is sorted. In total it takes 500 swaps to get all of the positions in the correct order, which is 1500 moves.

iv. Merge sort results in the same number of moves regardless of how the array is sorted because the algorithm calls itself the same times without checking how the array is sorted. It doesn't matter if the array is sorted or in reverse order since the same amount of calls will be made either way.

v.



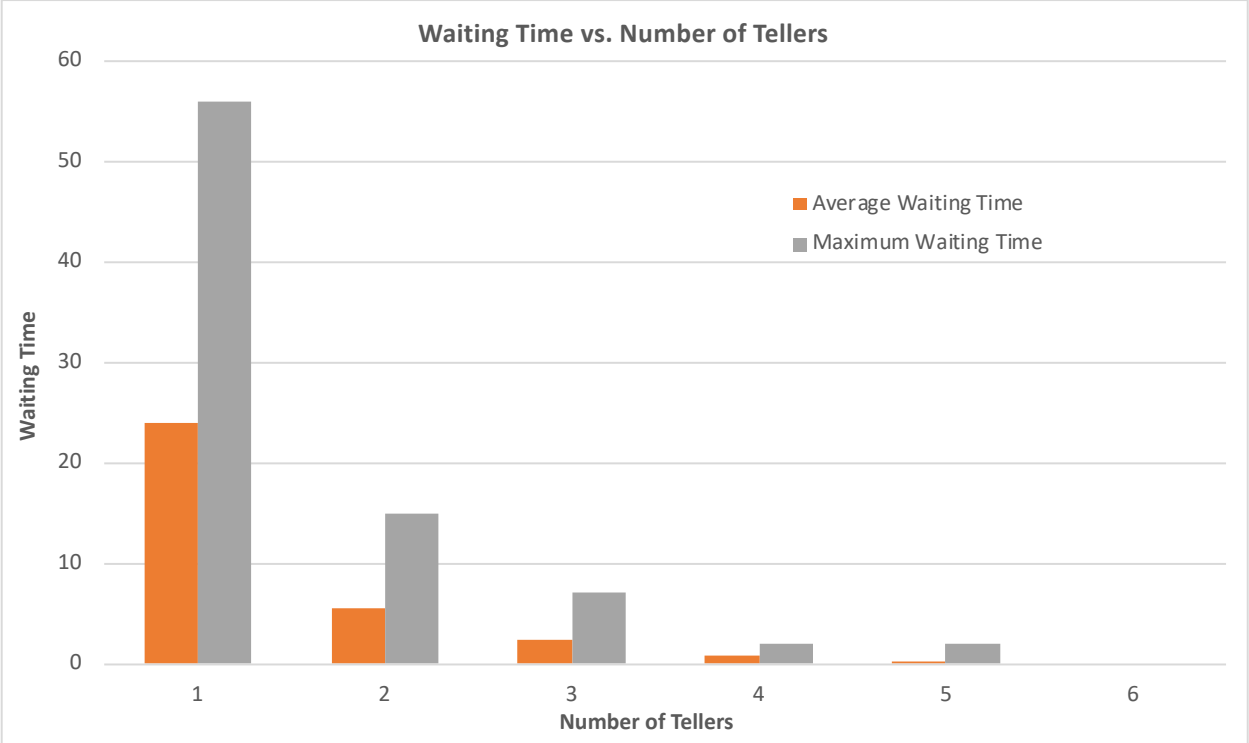


It is clear from these graphs that when taking both the number of moves and number of comparisons needed to sort a list that merge sort and quick sort are the best performing algorithms. Selection sort doesn't take that many moves but takes a large number of comparisons for all cases. Bubble sort performs well only on an already sorted list but performs quite poorly in other cases. Insertion sort overall is better on average than the first two algorithms discussed but still isn't the best. Merge sort and quick sort perform much better than the other algorithms, where performance for the worst case does not deteriorate nearly as badly compared to the other algorithms.

Alex Sadler

Problem 2

For the extra credit component of this assignment, I ran simulations with the same input example data from the assignment, but instead with two arrival events for every event (i.e. two customers arrived at time 1 with a 5 minute transaction time and so on).



- Below are the inputs for the BankTellerService function.
  - 1 5
  - 1 5
  - 2 5
  - 2 5
  - 4 5
  - 4 5
  - 20 5
  - 20 5
  - 22 5
  - 22 5
  - 24 5
  - 24 5
  - 26 5
  - 26 5
  - 28 5
  - 28 5

- 30 5
- 30 5
- 88 3
- 88 3

- Below is a sample output from using the above inputs with five tellers:
  -

```
Number of tellers: 5

Processing an arrival event at time  <-- 1
Processing an arrival event at time  <-- 1
Processing an arrival event at time  <-- 2
Processing an arrival event at time  <-- 2
Processing an arrival event at time  <-- 4
Processing an arrival event at time  <-- 4
Processing a departure event at time --> 6
Processing a departure event at time --> 6
Processing a departure event at time --> 7
Processing a departure event at time --> 7
Processing a departure event at time --> 9
Processing a departure event at time --> 11
Processing an arrival event at time  <-- 20
Processing an arrival event at time  <-- 20
Processing an arrival event at time  <-- 22
Processing an arrival event at time  <-- 22
Processing an arrival event at time  <-- 24
Processing an arrival event at time  <-- 24
Processing a departure event at time --> 25
Processing a departure event at time --> 25
Processing an arrival event at time  <-- 26
Processing an arrival event at time  <-- 26
Processing a departure event at time --> 27
Processing a departure event at time --> 27
Processing an arrival event at time  <-- 28
Processing an arrival event at time  <-- 28
Processing a departure event at time --> 29
Processing an arrival event at time  <-- 30
Processing an arrival event at time  <-- 30
Processing a departure event at time --> 30
Processing a departure event at time --> 31
Processing a departure event at time --> 32
Processing a departure event at time --> 33
Processing a departure event at time --> 34
Processing a departure event at time --> 35
Processing a departure event at time --> 36
Processing an arrival event at time  <-- 88
Processing an arrival event at time  <-- 88
Processing a departure event at time --> 91
Processing a departure event at time --> 91

Final Statistics:
Total number of customers processed: 20
Average waiting time = 0.3      Maximum waiting time = 2
Maximum waiting queue length = 2
```