1a)

Text

Description automatically generated

1b) The selected element is bold; the sorted elements are blue

*Selection Sort*

22**,** 8, **49**, 25, 18, 30, 20, 15, 35, 27

22**,** 8,27, 25, 18, 30, 20, 15, **35**, **49**

22**,** 8,27, 25, 18, **30**, 20, 15, **35**, **49**

22**,** 8, **27**, 25, 18, 15, 20, **30, 35, 49**

22**,** 8,20, **25**, 18, 15, **27**, **30, 35, 49**

**22,** 8,20, 15, 18, **25, 27, 30, 35, 49**

18**,** 8, **20**, 15, **22, 25, 27, 30, 35, 49**

**18,** 8,15, **20**, **22, 25, 27, 30, 35, 49**

**15,** 8, **18, 20, 22, 25, 27, 30, 35, 49**

8**, 15, 18, 20, 22, 25, 27, 30, 35, 49**

**8, 15, 18, 20, 22, 25, 27, 30, 35, 49**

*Bubble Sort*

**Pass 1: Pass 2:**

**22, 8,** 49, 25, 18, 30, 20, 15, 35, 27 **8, 22,** 25, 18, 30,20, 15, 35, 27, **49**

8**, 22, 49**, 25, 18, 30, 20, 15, 35, 27 8, **22, 25**, 18, 30,20, 15, 35, 27, **49**

8**,** 22**, 49**, **25**, 18, 30, 20, 15, 35, 27 8,22**, 25**, **18**, 30,20, 15, 35, 27, **49**

8**,** 22**,** 25, **49**, **18**, 30, 20, 15, 35, 27 8, 22,18, **25**, **30**,20, 15, 35, 27, **49**

8**,** 22**,** 25, 18, **49**, **30**, 20, 15, 35, 27 8, 22,18, 25, **30**, **20**, 15, 35, 27, **49**

8**,** 22**,** 25, 18, 30, **49**, **20**, 15, 35, 27 8, 22,18, 25, 20, **30**, **15**, 35, 27, **49**

8**,** 22**,** 25, 18, 30,20, **49**, **15**, 35, 27 8, 22,18, 25, 20,15, **30**, **35**, 27, **49**

8**,** 22**,** 25, 18, 30,20, 15, **49**, **35**, 27 8, 22,18, 25, 20,15, 30, **35**, **27**, **49**

8**,** 22**,** 25, 18, 30,20, 15, 35, **49**, **27** 8, 22,18, 25, 20,15, 30, 27, **35**, **49**

8**,** 22**,** 25, 18, 30,20, 15, 35, 27, **49**

**Pass 3: Pass 4:**

**8**, **22**,18, 25, 20,15, 30, 27, **35**, **49** **8**, **18**,22, 20, 15,25, 27, **30**, **35**, **49**

8, **22, 18,** 25, 20,15, 30, 27, **35**, **49** 8, **18**, **22**, 20, 15,25, 27, **30**, **35**, **49**

8, 18, **22**, **25**, 20,15, 30, 27, **35**, **49** 8, 18, **22**, **20**, 15,25, 27, **30**, **35**, **49**

8, 18,22, **25**, **20**,15, 30, 27, **35**, **49** 8, 18,20, **22, 15**,25, 27, **30**, **35**, **49**

8, 18,22, 20, **25**, **15**, 30, 27, **35**, **49** 8, 18,20, 15**, 22**, **25**, 27, **30**, **35**, **49**

8, 18,22, 20, 15, **25**, **30**, 27, **35**, **49** 8, 18,20, 15**,** 22, **25**, **27**, **30**, **35**, **49**

8, 18,22, 20, 15,25, **30**, **27**, **35**, **49** 8, 18,20, 15**,** 22,25, **27**, **30**, **35**, **49**

8, 18,22, 20, 15,25, 27, **30**, **35**, **49**

**Pass 5: Pass 6:**

**8, 18**, 20, 15, 22, 25, **27, 30, 35, 49 8, 18**, 15, 20, 22, **25, 27, 30, 35, 49**

8, **18, 20**, 15, 22, 25, **27, 30, 35, 49** 8, **18, 15**, 20, 22, **25, 27, 30, 35, 49**

8, 18, **20, 15**, 22, 25, **27, 30, 35, 49** 8, 15, **18, 20**, 22, **25, 27, 30, 35, 49**

8, 18, 15, **20, 22**, 25, **27, 30, 35, 49** 8, 15, 18, **20, 22**, **25, 27, 30, 35, 49**

8, 18, 15, 20, **22, 25**, **27, 30, 35, 49** 8, 15, 18, 20, **22, 25, 27, 30, 35, 49**

8, 18, 15, 20, 22, **25,** **27,** **30, 35, 49**

**Pass 7:**

There are no exchanges, so the algorithm will terminate.

Graphical user interface, text, application

Description automatically generated2c)

2d)

Random Arrays

Table

Description automatically generated

Almost Unsorted Arrays

Table

Description automatically generated

Almost Sorted Arrays

Table

Description automatically generated

3)

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

The three arrays used in the homework, *Random Arrays*, *Almost Unsorted Arrays*, *Almost Sorted Arrays*, represent average, worst, and best cases. Using these arrays, we can basically test the algorithms best case, worst case and average case. Since we already know the theoretical results of algorithms in these three cases, we can compare them without results.

The result shows that merge sort is the fastest algorithm on average of all four algorithms. The empirical result for all three arrays is near to O(nlogn) and in line with the theoretical results.

Quick Sort is the second fastest and almost comparable to merge sort. However, because of the way quick sort works, the time complexity of quick sort theoretically is O(n2 ) in the best case and O(nlogn) in the rest of the cases. From our quick sort graph, it can be seen that in almost sorted arrays, the time complexity is quite higher than the other arrays which follow O(nlogn) whereas this is in line with O(n2 ).

Insertion sort is the third slowest of the algorithms compared. The results of insertion sort are in line with theoretical results. In the case of random array, it performs near to O(n2 ), and for almost unsorted array, which can be taken as worst case more or less, it also performs at O(n2 ).

With almost sorted array it performs at O(n) which is also inline with the theoretical time complexity for best case in insertion sort.

Bubble Sort is the slowest of the algorithms compared. It always runs are O(n2 ) even if the array is sorted. This is also reflected in the graph of results obtained for bubble sort.