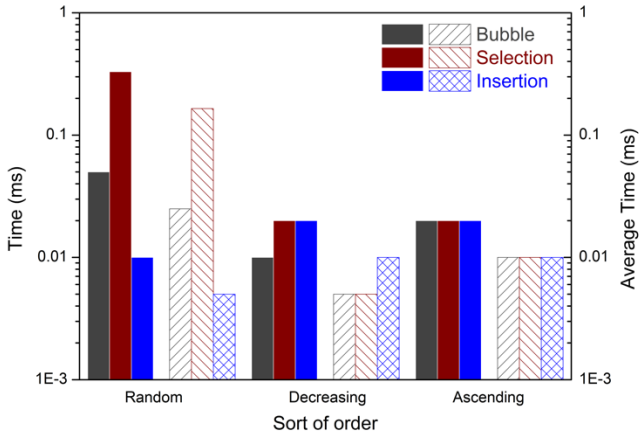
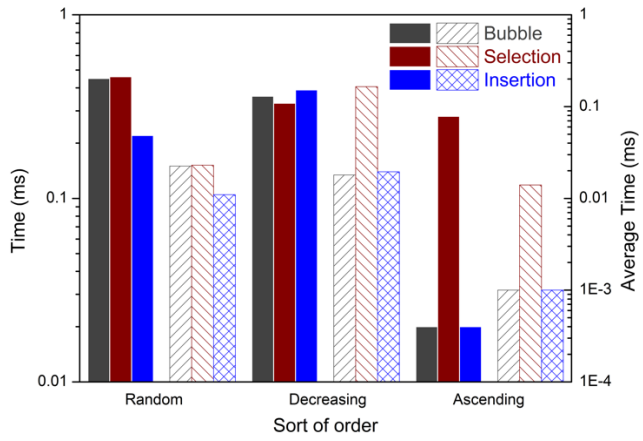
Report of Lab1

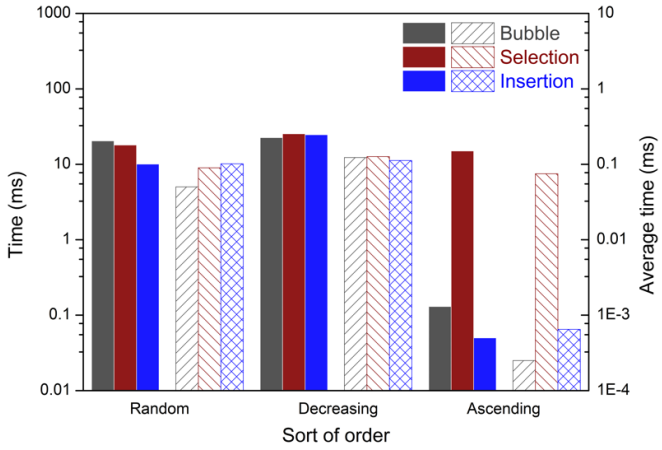
The experiment focuses on sorting different types of data using three slow sorting algorithms and then determining the total and average time for each algorithm to run.
In the original experiment the random and descending series were sorted using a slow sorting algorithm, However, to better test the efficiency of the three different slow sorting algorithms, I have included a test on the ascending series. And I found that the original bubble sorting algorithm will judge the numbers already sorted again, which will cause a lot of wasted time and resources, so I optimized the original bubble sorting algorithm by adding flag variable to eliminate the redundant comparison and thus improve the efficiency of the algorithm.
The following are statistical charts of the three algorithms for sorting different orders of magnitude and different types of series, which can clearly reflect the efficiency of the three algorithms.
10^1:



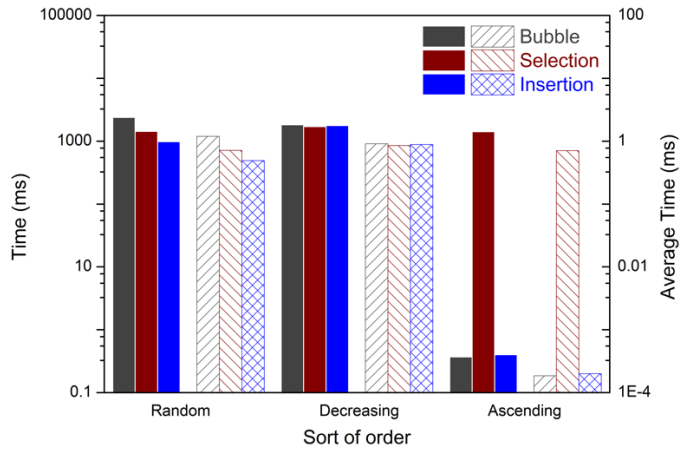
10^2:



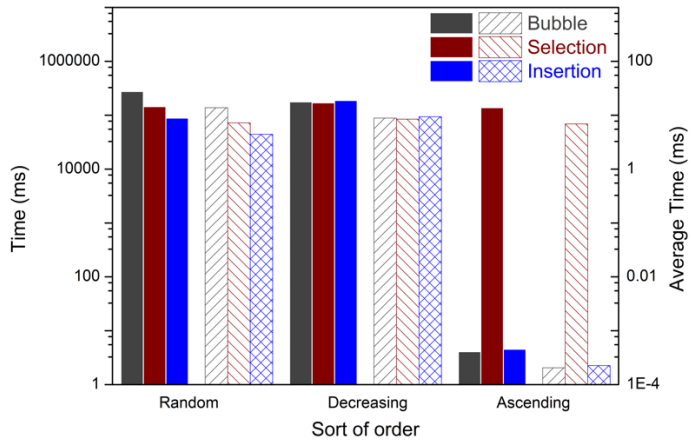
10^3:


Bubble
Selection
Insertion
Time (ms)
Average time (ms)
1000
100
10
1
0.1
0.01
10
1
0.1
0.01
1E-3
1E-4
Random
Decreasing
Ascending
Sort of order

10^4:


Bubble
Selection
Insertion
Time (ms)
Average Time (ms)
100000
1000
10
0.1
100
1
0.01
1E-4
Random
Decreasing
Ascending
Sort of order

10^5:


Bubble
Selection
Insertion
Time (ms)
Average Time (ms)
1000000
10000
100
1
100
1
0.01
1E-4
Random
Decreasing
Ascending
Sort of order

According to the graphs, we can know that bubble sort takes the longest time in the case of many random columns, followed by selection sort, and finally insertion sort, but they are all the same in time level, and the time complexity of all three sorting algorithms is O(n^2) for the random case.

For descending arrays, i.e., for the worst case of the sorting algorithm, the time of the three algorithms is about the same, and in the worst case, the time complexity of the three slow sorting algorithms is also O(n^2).

Because I added an ascending array, i.e., a completely sorted array, the insertion sort in this case shows a very high speed because it does not bother with redundant comparisons, while for the traditional bubble sort algorithm, the time complexity in this case should be also O(n^2), but after I optimized the algorithm to achieve the same O(n) as the insertion sort in this case, the time complexity of the selection sort in this case is still O(n^2).

So, all three slow sorts have a time complexity of O(n^2) for the general case, which is not a good choice, while the insertion sort has a higher efficiency in special cases, as does the optimized bubble sort.