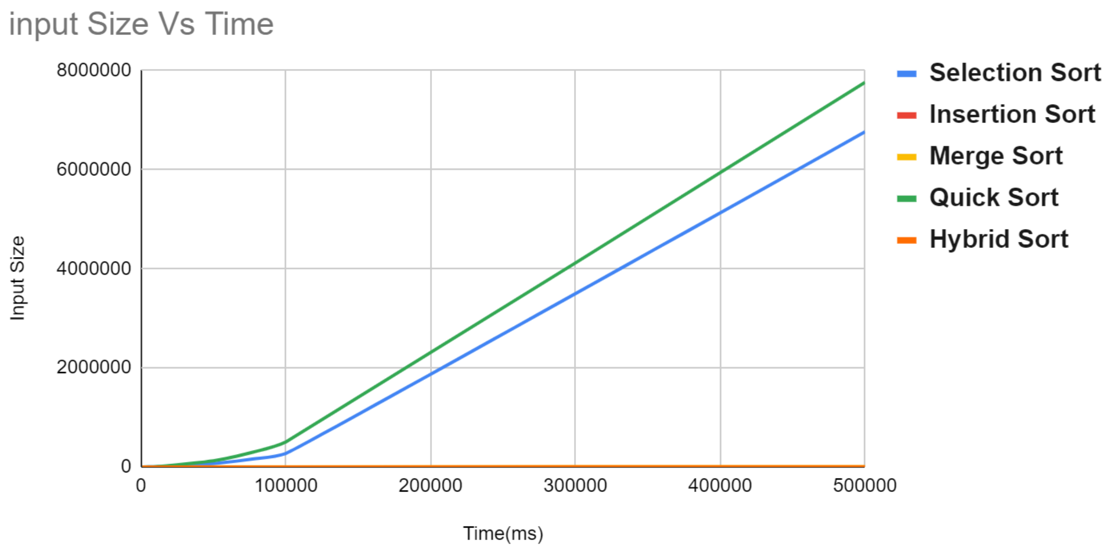
Cairo University
Faculty of Engineering

Assignment HW2

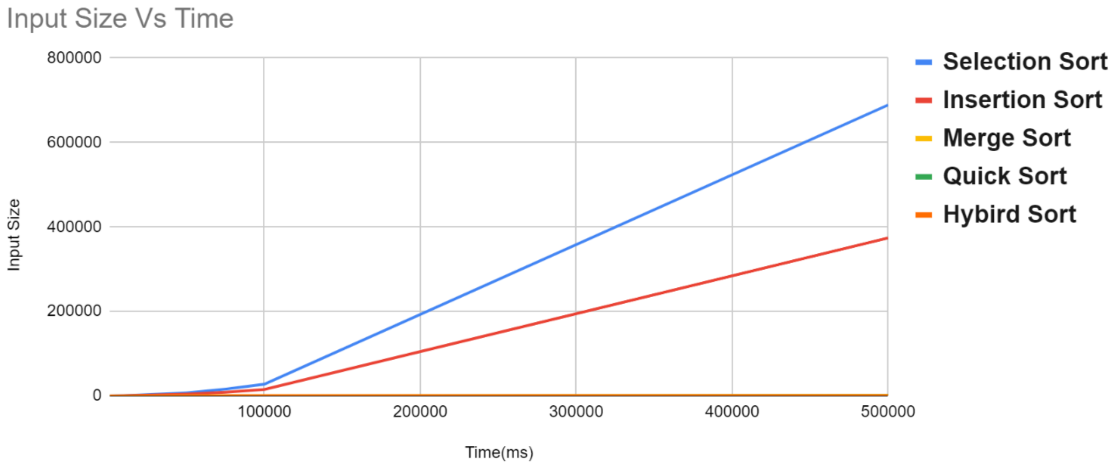
Name	Section	B.N
Aly Ramzy Hassan	1	36

SORTED



	Selection Sort	Insertion Sort	Merge Sort	Quick Sort	Hybrid Sort
1000	3.511	0.008	1.597	5.028	0.181
5000	70.904	0.097	7.886	126.543	1.179
10000	276.1	0.104	16.008	505.498	2.474
50000	6778.94	0.357	81.621	12506.9	14.241
75000	15264.7	0.557	125.365	28178.3	24.73
100000	27217.6	0.75	161.081	50274.5	29.848
500000	676336	3.529	845.656	776336	198.672

UN SORTED



	Selection Sort	Insertion Sort	Merge Sort	Quick Sort	Hybrid Sort
1000	3.768	1.532	1.677	0.231	0.299
5000	71.563	38.486	8.127	0.882	2.279
10000	274.909	149.457	17.438	1.618	4.3
50000	6805.1	3730.05	85.329	10.013	27.332
75000	15536.4	8485.21	142.039	15.834	41.524
100000	27203.5	14741.4	172.769	21.364	56.274
500000	688966	373914	898.538	124.33	301.632

Hybrid Sort:

I used combination of Merge sort and insertion sort because Merge sort is O(nlogn) and insertion sort is O(n^2),and at small size (100 as example) O(n^2) is better than O(nlogn) ,so in code I start with merge sort if size of the array is very large and when dividing into small array if the size of the small array is less than 100 I start using insertion sort ,and this lead to a sorting algorithm better than the original merge sort (look at the time in the table).

```
Hybrid Sort =>
If(size>100)
        Merge sort
Else
        Insertion Sort
```

Notes:

1-my code outputs the Sorting Algorithm name and Inputs Size

And time for UN Sorted and time for Sorted for each run of code

2-i used bash script to run all input sizes for all sorting algorithms

(attached with the files : script.sh)