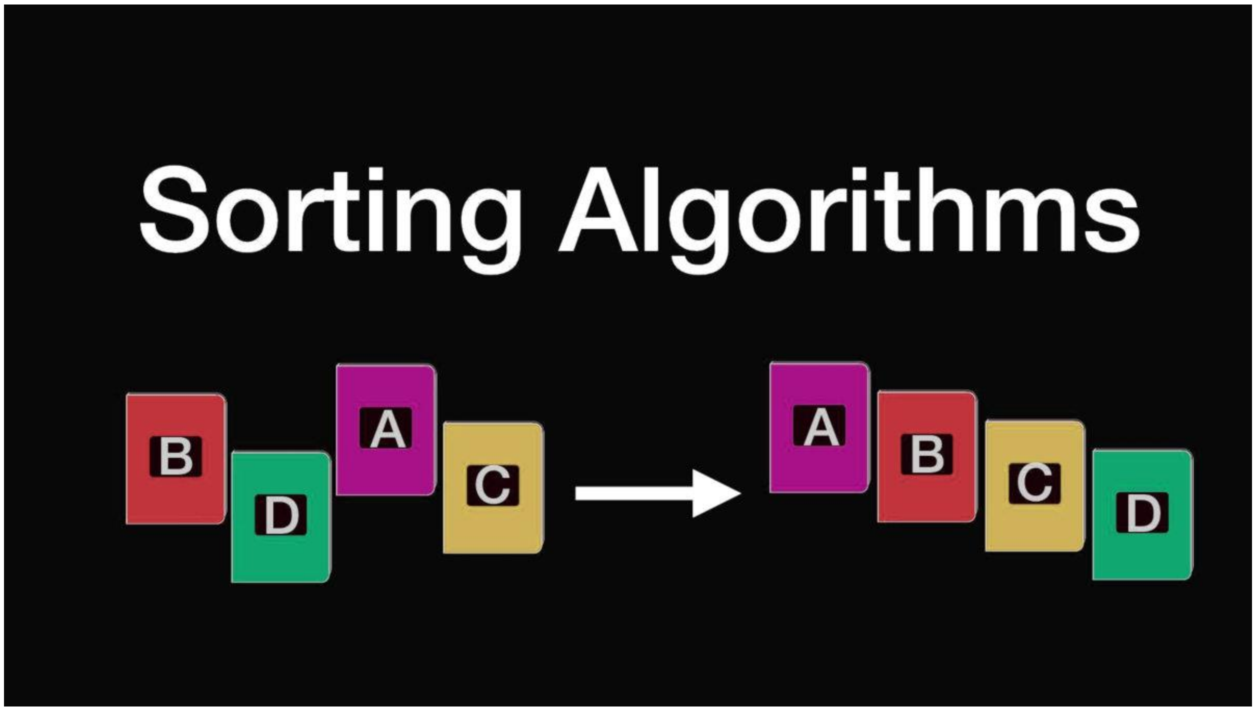
جامعة الإسكندرية
ALEXANDRIA
UNIVERSITY

DATA structures 2 Report

Alaa Hossam 6750 Group 4 section 2
Manar Amgad 7113 Group 4 section 2
Nadeen Ayman 7130 Group 1

Sorting Algorithms

A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure.



1)Quick sort:

QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

We took a random pivot and start partioning randomly by pick up the index of the pivot randomly then swap it with the last element in the array and start our portioning and sorting by last element as pivot algorithm.

The average time complexity of quick sort is O(N log(N)).As we are not creating any container other then given array therefore Space complexity will be in order of N ,O(N).

Best Case Time Complexity: **O(n*log n)**

Worst Case Time Complexity: **O(n^2)**

Average Time Complexity: **O(n*log n)**

2) Merge sort:

Like QuickSort, Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves. the worst-case run time of this algorithm is O(nLogn)

Auxiliary Space: **O(n)**
Sorting In Place: No

Best Case Time Complexity: **O(n*log n)**

Worst Case Time Complexity: **O(n*log n)**

Average Time Complexity: **O(n*log n)**

3)Selection sort

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array. Since the algorithm performs 2 loops iterating over the array, it has a time complexity of **O(n^2)**.This algorithm is not suitable for large data sets as its average and worst case complexities are of O(n^2), where **n** is the number of items.

Best Case Time Complexity: **O(n^2)**.

Worst Case Time Complexity: **O(n^2)**.

Average Time Complexity: **O(n^2)**.

4)Insertion sort

It is very similar to sorting cards. Divide the vector into two parts, one sorted and the other one unsorted. Now, we insert cards(elements) from the unsorted part into the sorted part at their correct position. This method is also quadratic with N.

Best Case Time Complexity: **O(n^2)**.

Worst Case Time Complexity: **O(n^2)**.

Average Time Complexity: **O(n^2)**.

Hybrid sorting :

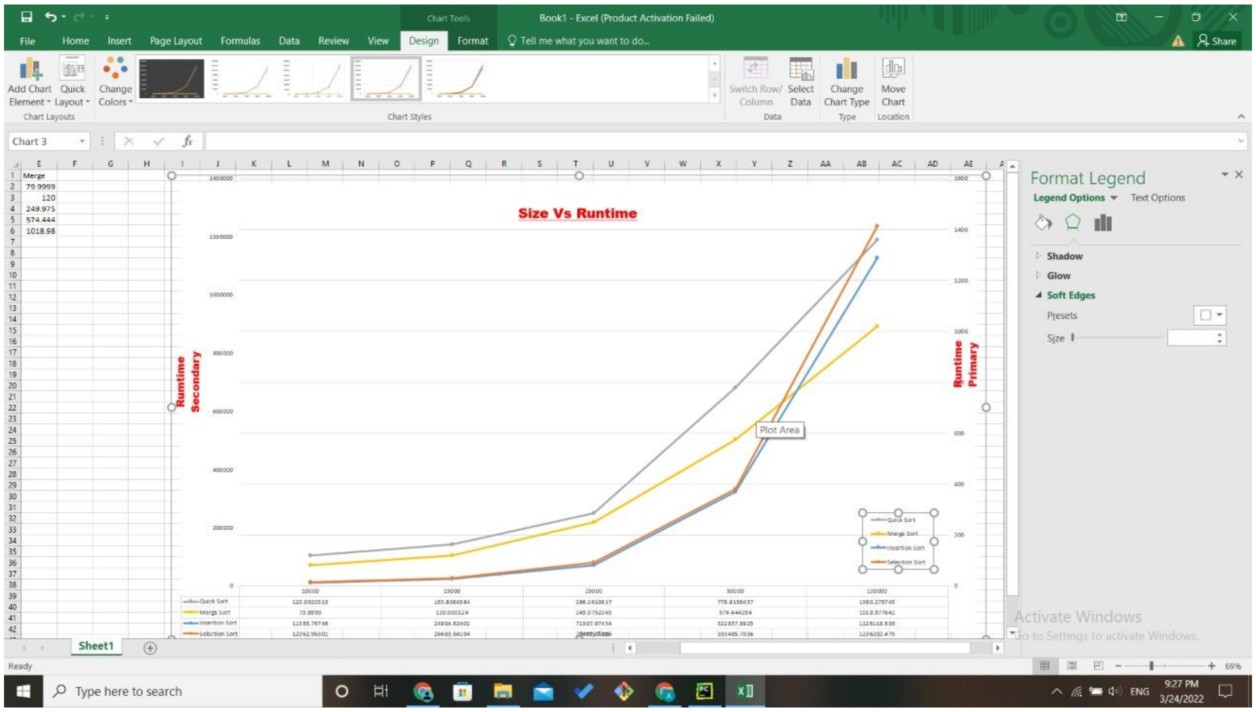
In this project, a Hybrid algorithm with the combination of merge sort and selection sort is implemented. As the name suggests, the Hybrid algorithm combines more than one algorithm.

Merge algorithm is efficient if the size of the input is very large. But, selection sort is more efficient than quick sort in case of small arrays as the number of comparisons and swaps are less compared to quicksort.

Kth Smallest Element:

In this method the user will enter the order [k] of the smallest element he wants to find, then we will compare this value with the index of the pivot [i] in the array. If k>i then the number that we want is less than the pivot so it will be found in the left subarray. If k<i then the number we want to find is greater than the pivot and it will be found in the right subarray.

Graph to indicate the difference in the runtime between each sorting technique:



	E
1	Merge
2	79.9999
3	120
4	249.975
5	574.444
6	1018.98

	10000	15000	25000	50000	100000
Quick Sort	120.0020513	163.8064584	286.2610617	778.9156457	1560.275745
Merge Sort	79.9999	120.000124	249.9752045	574.444294	1018.977642
Insertion Sort	11385.76746	24904.82402	71507.97454	322857.8925	1126118.536
Selection Sort	12062.96301	26683.84194	[illegible]	333485.7936	1236292.475

<u>Comment :</u>

Merge sort and quick sort are very close to each other **but** The worst case complexity of quick sort is O(n2) as there is need of lot of comparisons in the worst condition.
whereas
In merge sort, worst case and average case has same complexities O(n log n),Also Quick sort is preferred for arrays. whereas Merge sort is preferred for linked lists.

The average time complexity of selection and insertion algorithms is **O(n^2)** but as the size of input data increases, insertion sort performs slightly better than selection sort.

ERRORS we have faced:

1-identation errors.

2- syntax errors.

3-using randrange method instead of randint.

4-Recursion error in quick sort : The main problem we faced was that when we try to put the random pivot as first element in the array and maximizing the size of the array we work on it , we had Recursion Error which said that the recursion calls in python is limited so we cannot exceed it.