DATA STRUCTURES II

LAB 1

Names:

1- Amr Yasser Imam – 6772

2- Marwan Khaled Mohamed – 7020

3- Begad Wael – 6718

Repository address:

https://github.com/AMR-21/Sorting-Algorithms

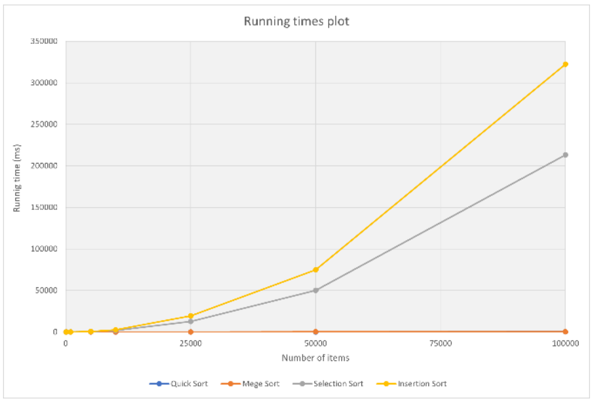
1- Time comparison

On testing different arrays of unique integers with different size (100 – 1000 – 5000 – 10000 - 25000 – 50000 – 100000), the following results for (Quick Sort – Merge Sort – Selection Sort – Insertion Sort) was obtained as shown in TAB.1.1.

Algorithm – Size	100	1000	5000	10000	25000	50000	100000	Time (ms)
Quick Sort	0	0.99	7.99	18.99	46	101.01	211	
Merge Sort	1	2.99	14	33.01	81	179.99	365.99	
Selection Sort	0	19	486.26	1906.08	12687.45	50194.72	213170	
Insertion Sort	0	25	623.02	2726.11	19392.05	74761.92	322379.9	

TAB.1.1. Running times for different algorithms

The following FIG.1.1. and FIG.1.2. represent the plots of the running time for each algorithm versus the array size



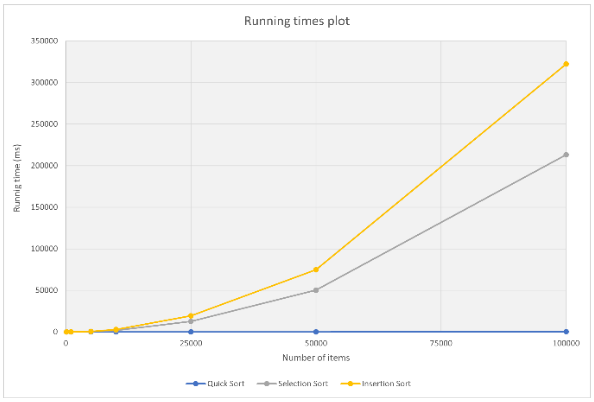
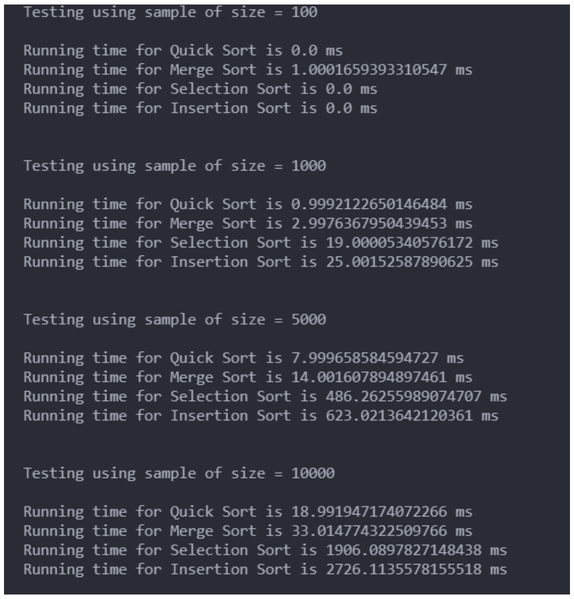


FIG.2.1. Plots for Merge, Quick, Selection, and Insertion sorting algorithms

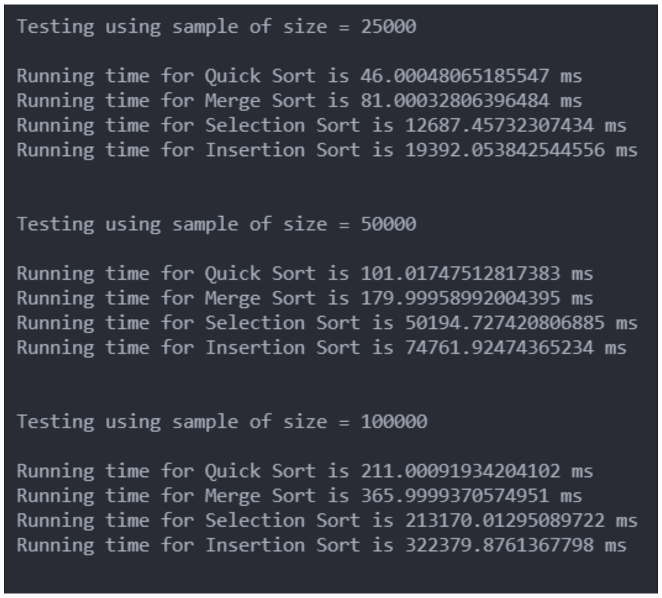
FIG.3.2. Plot showing Quick sort algorithm curve

The following FIG.1.3. and FIG.1.4. are the results captured inside the program



```
 Testing using sample of size = 100

 Running time for Quick Sort is 0.0 ms
 Running time for Merge Sort is 1.0001659393310547 ms
 Running time for Selection Sort is 0.0 ms
 Running time for Insertion Sort is 0.0 ms


 Testing using sample of size = 1000

 Running time for Quick Sort is 0.9992122650146484 ms
 Running time for Merge Sort is 2.9976367950439453 ms
 Running time for Selection Sort is 19.00005340576172 ms
 Running time for Insertion Sort is 25.00152587890625 ms


 Testing using sample of size = 5000

 Running time for Quick Sort is 7.999658584594727 ms
 Running time for Merge Sort is 14.001607894897461 ms
 Running time for Selection Sort is 486.26255989074707 ms
 Running time for Insertion Sort is 623.0213642120361 ms


 Testing using sample of size = 10000

 Running time for Quick Sort is 18.991947174072266 ms
 Running time for Merge Sort is 33.014774322509766 ms
 Running time for Selection Sort is 1906.0897827148438 ms
 Running time for Insertion Sort is 2726.1135578155518 ms
```

FIG.4.3. Test cases (100 – 1000 – 5000 – 10000)



```
Testing using sample of size = 25000

Running time for Quick Sort is 46.00048065185547 ms
Running time for Merge Sort is 81.00032806396484 ms
Running time for Selection Sort is 12687.45732307434 ms
Running time for Insertion Sort is 19392.053842544556 ms


Testing using sample of size = 50000

Running time for Quick Sort is 101.01747512817383 ms
Running time for Merge Sort is 179.99958992004395 ms
Running time for Selection Sort is 50194.727420806885 ms
Running time for Insertion Sort is 74761.92474365234 ms


Testing using sample of size = 100000

Running time for Quick Sort is 211.00091934204102 ms
Running time for Merge Sort is 365.9999370574951 ms
Running time for Selection Sort is 213170.01295089722 ms
Running time for Insertion Sort is 322379.8761367798 ms
```

FIG.5.4. Test cases (25000 – 50000 - 100000)

2- Quick Select Algorithm

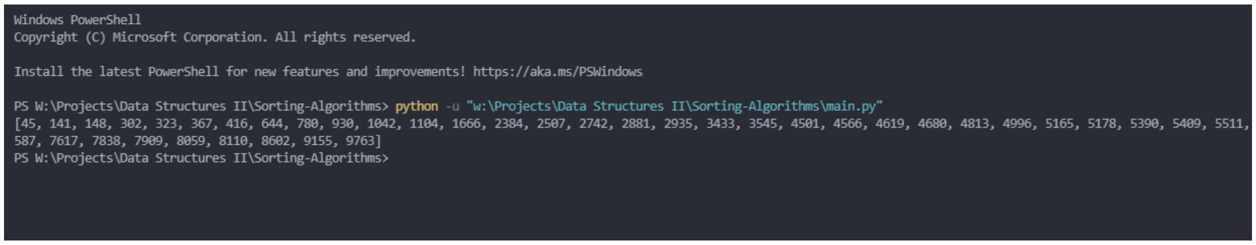
We implemented a function called quickSelect(list,k,left,right) that take the array and the order of item to be found, afterwards, we generated a random array with (name arr and size = 50).

```
arr = [7838, 5511, 7617, 2935, 4619, 5409, 8602, 1042, 5390, 45, 8110, 3545,
5761, 1666, 5165, 7909, 5178, 2881, 7013, 1104, 9763, 4813, 7043, 7048, 4996,
416, 7076, 5565, 4501, 4680, 5827,
     644, 4566, 3433, 8059, 7132, 780, 7587, 9155, 302, 2742, 141, 7106, 2507,
367, 2384, 6272, 323, 148, 930]
```

2.1 Test case

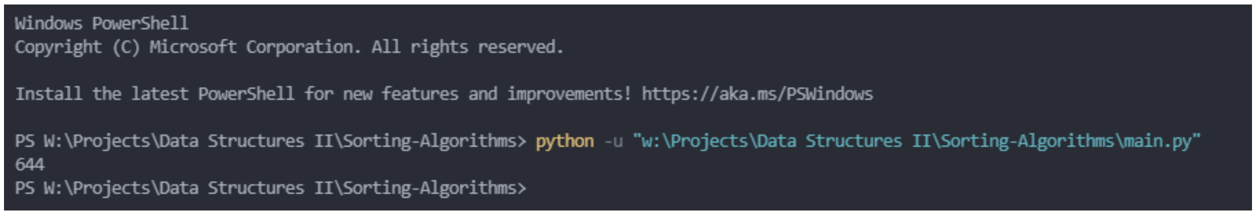
We tested the function by searching for the 8th smallest element by calling it with the following call quickSelect(arr,8,0,len(arr)-1). The result was 644 and to make sure it's the correct element we sorted the array as shown in FIG.2.1.1. and indeed, we found that the 8th smallest element is 644 as shown in FIG.2.1.2..

```
quickSelect(arr,8,0,len(arr)-1)
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS W:\Projects\Data Structures II\Sorting-Algorithms> python -u "w:\Projects\Data Structures II\Sorting-Algorithms\main.py"
[45, 141, 148, 302, 323, 367, 416, 644, 780, 930, 1042, 1104, 1666, 2384, 2507, 2742, 2881, 2935, 3433, 3545, 4501, 4566, 4619, 4680, 4813, 4996, 5165, 5178, 5390, 5409, 5511,
587, 7617, 7838, 7909, 8059, 8110, 8602, 9155, 9763]
PS W:\Projects\Data Structures II\Sorting-Algorithms>
```

FIG.2.1.1. The sorted array shows that 644 is the 8th smallest element



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS W:\Projects\Data Structures II\Sorting-Algorithms> python -u "w:\Projects\Data Structures II\Sorting-Algorithms\main.py"
644
PS W:\Projects\Data Structures II\Sorting-Algorithms>
```

FIG.2.1.2. 8th smallest element in the array

3- Hybrid Merge and Selection Algorithm

On testing different arrays of unique integers with different size (50 - 100 – 1000 – 10000 – 25000) with threshold = 10, the following results for (Hybrid Merge and Selection – Merge Sort – Selection Sort) was obtained as shown in TAB.3.1.

```
testHybrid([50,100,1000,10000,25000],10)
```

Algorithm – Size	50	100	1000	10000	25000	Time (ms)
Hybrid Sort	0	0	1.99	23.99	63	
Merge Sort	0	0	1.99	30.99	88	
Selection Sort	0	0	21	1897.64	11891.81	

TAB.3.1. Running times for the three algorithms

The following FIG.3.1. represents the plots of the running time for each algorithm versus the array size

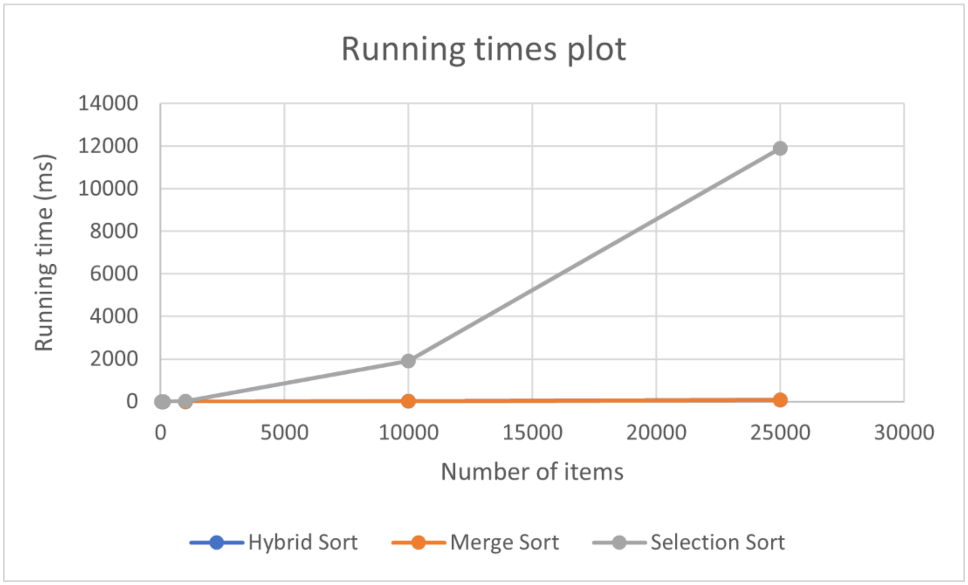
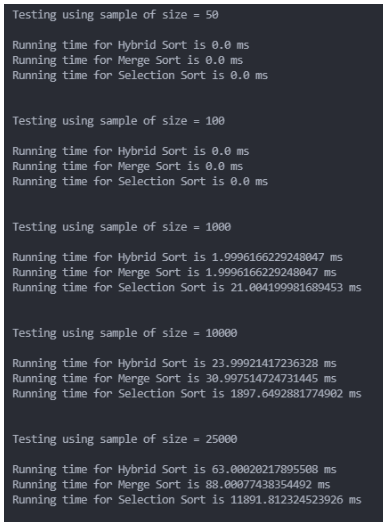


FIG.3.1. Plots for the three algorithms

The following FIG.3.2. is the results captured inside the program



```
Testing using sample of size = 50

Running time for Hybrid Sort is 0.0 ms
Running time for Merge Sort is 0.0 ms
Running time for Selection Sort is 0.0 ms


Testing using sample of size = 100

Running time for Hybrid Sort is 0.0 ms
Running time for Merge Sort is 0.0 ms
Running time for Selection Sort is 0.0 ms


Testing using sample of size = 1000

Running time for Hybrid Sort is 1.9996166229248047 ms
Running time for Merge Sort is 1.9996166229248047 ms
Running time for Selection Sort is 21.004199981689453 ms


Testing using sample of size = 10000

Running time for Hybrid Sort is 23.99921417236328 ms
Running time for Merge Sort is 30.997514724731445 ms
Running time for Selection Sort is 1897.6492881774902 ms


Testing using sample of size = 25000

Running time for Hybrid Sort is 63.00020217895508 ms
Running time for Merge Sort is 88.00077438354492 ms
Running time for Selection Sort is 11891.812324523926 ms
```

FIG.3.2. Test cases (50 - 100 – 1000 – 10000 – 25000)