CS 202 Homework 1

Eren Şenoğlu
21702079

Question 1-

A -> f3 > f1 > f7 > f2 > f8 > f6 = f9 >f10 > f4 > f5

B ->

It is an recursive algorithm so, let's assume T(n) = Run time for input n.
If n <=0, then T(n) = 1, as only one comparison is required.
If n >= 0, then with the first comparison, the else part is executed. Input random value i is generated by random() function and the return value is added by (n-1-i).
Now we have T(0) = 1, If we expand T(n) = T(n-1) + 1 , n >= 0 like this we will get
T(n) = T(1) + (n-1) = 1 + (n-1)
Hence, it can be said that time taken is T(n) = n.

C ->

Bubble Sort
Pass 0
607,1896,1165,2217,675,2492,2706,894,743,568
Pass 1
607,1165,1896,675,2217,2492,894,743,568,2706
Pass 2
607,1165,675,1896,2217,894,743,568,2492,2706
Pass 3
607,675,1165,1896,894,743,568,2217,2492,2706
Pass 4
607,675,1165,894,743,568,1896,2217,2492,2706
Pass 5
607,675,894,743,568,1165,1896,2217,2492,2706
Pass 6
607,675,743,568,894,1165,1896,2217,2492,2706
Pass 7
607,675,568,743,894,1165,1896,2217,2492,2706
Pass 8
607,568,675,743,894,1165,1896,2217,2492,2706
Pass 9
568,607,675,743,894,1165,1896,2217,2492,2706

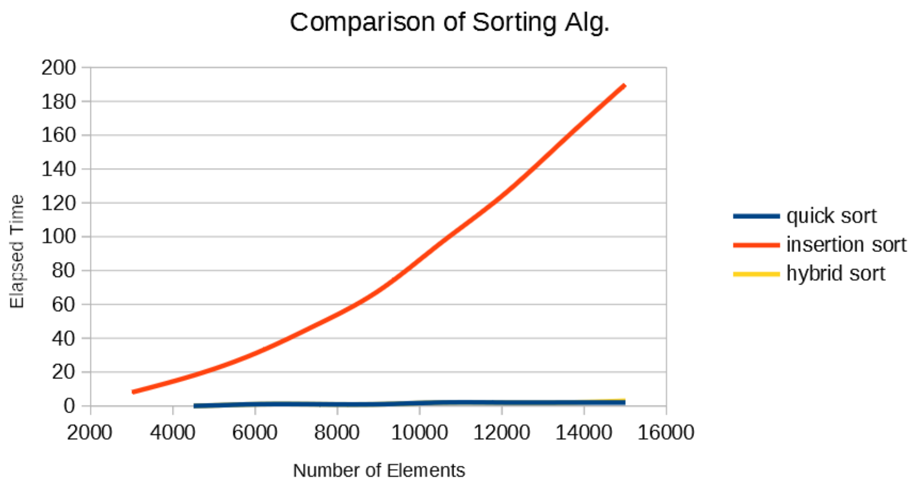
Radix Sort
0607,1896,1165,2217,0675,2492,2706,0894,0743,0568

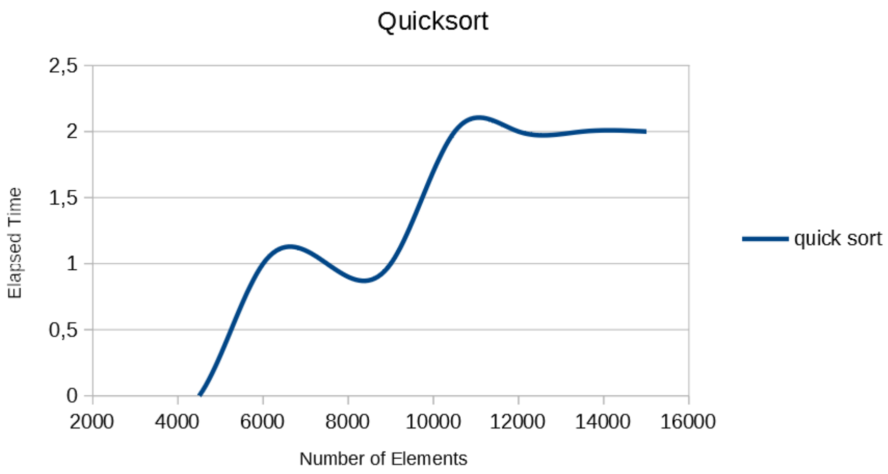
(2492),(0743),(0894),(0675,1165),(2706,1896),(2217,0607),(0568)	Grouped by four digit
2492 , 0743 , 0894 , 0675 , 1165 , 2706 , 1896 , 2217 , 0607 , 0568	Combined
(2706,0607),(2217),(0743),(1165,0568),(0675),(2492,0894,1896)	Grouped by third digit
2706,0607,2217,0743,1165,0568,0675,2492,0894,1896	Combined
(1165),(2217),(2492),(0568),(0607,0675),(2706,0743),(0894,1896)	Grouped by second digit
1165,2217,2492,0568,0607,0675,2706,0743,0894,1896	Combined
(0568,0607,0675,0743,0894),(1165,1896),(2217,2492,2706)	Grouped by first digit
0568,0607,0675,0743,0894,1165,1896,2217,2492,2706	Combined

Question 3-

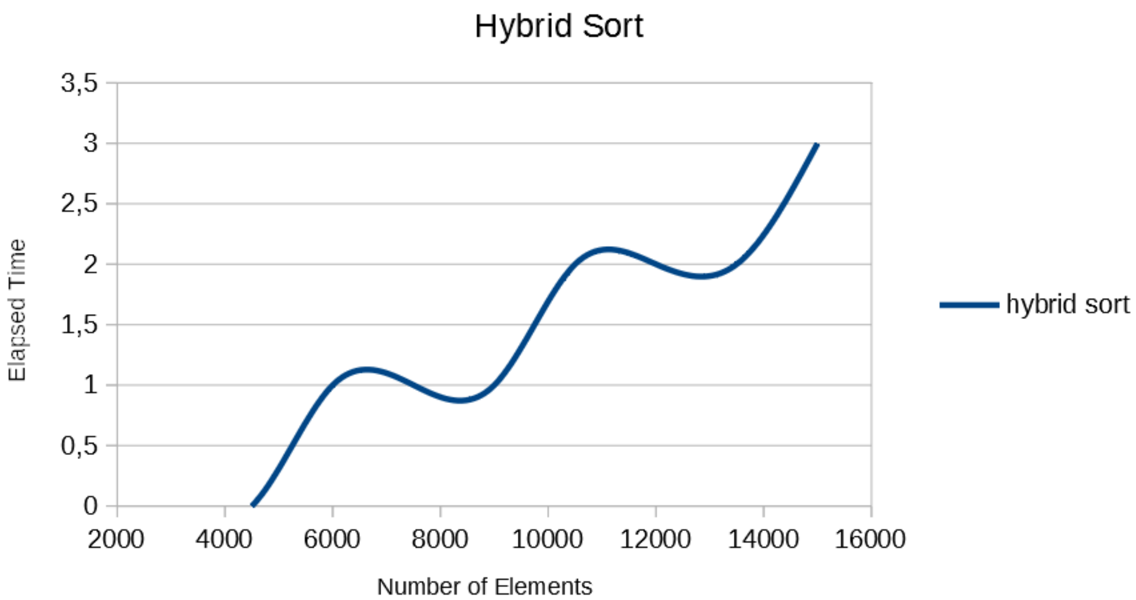
Size	Quick Sort	Insertion Sort	Hybrid Sort
3000	0 ms	8 ms	0 ms
4500	1 ms	18 ms	0 ms
6000	1 ms	31 ms	1 ms
7500	1 ms	48 ms	0 ms
9000	1 ms	68 ms	1 ms
10500	2 ms	96 ms	2 ms
12000	2 ms	124 ms	1 ms
13500	2 ms	157 ms	1 ms
15000	2 ms	190 ms	3 ms

In this study, I used three different algorithms which implement sorting solutions with different approaches. As a result of their different approaches, each of them has different complexities. The first algorithm, which theoretically has $O(N^2)$ complexity, was using two nested for loops to solve the problem. I measured the taken time by the algorithm. As we know, Big-O notation shows the upper bound and results that are found in the run was upper bounded by Big-O notation curve as it is expected. The second algorithm was using a faster algorithm which has O(N*log(N)) complexity for the average case. As expected time results were under the Big-O notation curve. The remaining algorithm was an faster algorithm which have O(k*N) (k = threshold) complexity and again run-time results were upper bounded by the worst-case curve. From graphs we can see that the worst efficient algorithm is the Insertion sort, as it should be in this way theoretically. The most efficient one is the Hybrid Sort. Also from graphs we can't see the difference of hybrid sort and quick sort for our sample input sizes, however we can see it from table. As the sample input size increases, run-time differrence of hybrid sort and quick sort gets observable. As hybrid sort is faster theoretically, this comparison holds the theoretical values. Combining two algorithms creates a faster algorithm than themselves. Hence, our theoretical values holds our experimental results, as expected.




Quicksort
2,5
2
1,5
1
0,5
0
Elapsed Time
2000
4000
6000
8000
10000
12000
14000
16000
Number of Elements
quick sort


Insertion Sort
200
180
160
140
120
100
80
60
40
20
0
Elapsed Time
2000
4000
6000
8000
10000
12000
14000
16000
Number of Elements
insertion sort


Hybrid Sort
3,5
3
2,5
2
1,5
1
0,5
0
Elapsed Time
2000
4000
6000
8000
10000
12000
14000
16000
Number of Elements
hybrid sort

eren.senoglu@dijkstra:~

```
[eren.senoglu@dijkstra ~]$ make
g++ main.cpp sorting.cpp sorting.h -o hw1
[eren.senoglu@dijkstra ~]$ ./hw1
1 2 6 7 9 11 17 20 22 24 27 30
1 2 6 7 9 11 17 20 22 24 27 30
1 2 6 7 9 11 17 20 22 24 27 30
-----------------------
Part a - Time analysis of Quick Sort
-----------------------
Array Size     Time elapsed   compCount      moveCount
3000           0         ms    43948          74097
4500           0         ms    63738          111281
6000           0         ms    86665          122752
7500           0         ms    106713         190334
9000           0         ms    137124         214001
10500          10        ms    156555         257498
12000          10        ms    195657         330629
13500          10        ms    225456         399515
15000          0         ms    238478         370523
-----------------------
Part b - Time analysis of Insertion Sort
-----------------------
Array Size     Time elapsed   compCount      moveCount
3000           10        ms    2252403        2255407
4500           40        ms    5082700        5087206
6000           60        ms    8953284        8959294
7500           100       ms    14095432       14102941
9000           140       ms    20301545       20310551
10500          180       ms    27570312       27580821
12000          240       ms    35841654       35853660
13500          300       ms    45480573       45494076
15000          390       ms    56935912       56950920
-----------------------
Part c - Time analysis of Hybrid Sort
-----------------------
Array Size     Time elapsed   compCount      moveCount
3000           0         ms    45128          69271
4500           0         ms    65538          104067
6000           0         ms    88635          112660
7500           0         ms    109187         178013
9000           0         ms    139991         199311
10500          0         ms    160078         241358
12000          0         ms    199400         311723
13500          10        ms    229798         378479
```