Question 1

Question - 1 Utku Boran Torun

a) $f(n) = 3n^4 + 4n^3 + n \leq c \cdot n^5 \Rightarrow (3n^3 + 4n^2 + 1) \leq c \cdot n^4$

$\Rightarrow 3n^3 + 4n^2 + 1 \leq cn^4$

$c = 3$ and $n_0 = 2$ hold this equation

$\Rightarrow 3 \cdot 8 + 4 \cdot 4 + 1 \leq 3 \cdot 16 \Rightarrow 41 \leq 48$ is true ✓

When we set c to 3, all $n_0 \geq 2$ holds this equation.

---

b) At first arr = [31, 17, 58, 35, 45, 27, 36, 28, 24, 34]

Selection Sort

1st iteration => swap 58 with 34

arr = [31, 17, 34, 35, 45, 27, 36, 28, 24, 58]

2nd iteration => swap 45 with 24

arr = [31, 17, 34, 35, 24, 27, 36, 28, 45, 58]

3rd iteration => swap 36 with 28

arr = [31, 17, 34, 35, 24, 27, 28, 36, 45, 58]

4th iteration => swap 35 with 28

arr = [31, 17, 34, 28, 24, 27, 35, 36, 45, 58]

5th iteration => swap 34 with 27

arr = [31, 17, 27, 28, 24, 34, 35, 36, 45, 58]

6th iteration => swap 31 with 24

arr = [24, 17, 27, 28, 31, 34, 35, 36, 45, 58]

7th iteration => swap 28 with nothing (array stays same)

8th iteration => swap 27 with nothing (array stays same)

9th iteration => swap 24 with 17

arr = [17, 24, 27, 28, 31, 34, 35, 36, 45, 58] => Sorted array ✓

Insertion Sort

Utku Baran Torun

We will have sorted and unsorted sublist. At first the first element (31) will be in sorted sublist, the rest is in unsorted sublist.

1st iteration => Key = 17

Sorted sublist = {17, 31}, Unsorted sublist = {58, 35, 45, 27, 36, 28, 24, 34}

2nd iteration => Key = 58

Sorted sublist = {17, 31, 58}, Unsorted sublist = {35, 45, 27, 36, 28, 24, 34}

3rd iteration => Key = 35

Sorted sublist = {17, 31, 35, 58}, Unsorted sublist = {45, 27, 36, 28, 24, 34}

4th iteration => Key = 45

Sorted sublist = {17, 31, 35, 45, 58}, Unsorted sublist = {27, 36, 28, 24, 34}

5th iteration => Key = 27

Sorted sublist = {17, 27, 31, 35, 45, 58}, Unsorted sublist = {36, 28, 24, 34}

6th iteration => Key = 36

Sorted sublist = {17, 27, 31, 35, 36, 45, 58}, Unsorted sublist = {28, 24, 34}

7th iteration => Key = 28

Sorted sublist = {17, 27, 28, 31, 35, 36, 45, 58}, Unsorted sublist = {24, 34}

8th iteration => Key = 24

Sorted sublist = {17, 24, 27, 28, 31, 35, 36, 45, 58}, Unsorted sublist = {34}

9th iteration => Key = 34

Sorted sublist = {17, 24, 27, 28, 31, 34, 35, 36, 45, 58}, Unsorted sublist = $\emptyset$

When unsorted sublist is empty, sorted sublist is our sorted array.

arr = [17, 24, 27, 28, 31, 34, 35, 36, 45, 58] //

Question-2

c)

```
--Bubble Sort Algorithm--
11, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29
Bubble sort comparison count: 114
Bubble sort move count: 180

--Merge Sort Algorithm--
11, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29
Merge sort comparison count: 77
Merge sort move count: 128

--Quick Sort Algorithm--
11, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29
Quick sort comparison count: 64
Quick sort move count: 102
```

d) Analysis

Random Array Analysis

```
------------------------------------------------------------------
Random Array Bubble Sort Analysis
Array Size        Elapsed Time        compCount       moveCount
4000              44 ms               7981347          11745744
8000              175 ms              31974472          47930808
12000             418 ms              71993985          108179841
16000             744 ms              127970472          192243045
20000             1181 ms              199981089          301323513
24000             1725 ms             287985722          434665155
28000             2298 ms             391965900         584333328
32000             3029 ms             511930372         764815902
36000             3815 ms             647956349         968482698
40000             4709 ms             799977074         1193825970

------------------------------------------------------------------
Random Array Merge Sort Analysis
Array Size        Elapsed Time        compCount       moveCount
4000              1 ms                50817          95808
8000              2 ms               109641          207616
12000             3 ms               171682          327232
16000             4 ms               235401          447232
20000             5 ms               301006          574464
24000             7 ms              367493          702464
28000             7 ms              435018         830464
32000             8 ms              502517         958464
36000             9 ms              571962         1092928
40000             11 ms              641724          1228928

------------------------------------------------------------------
Random Array Quick Sort Analysis
Array Size        Elapsed Time        compCount       moveCount
4000              0 ms                59100          81871
8000              0 ms               131820          199369
12000             1 ms               215760          325527
16000             1 ms               275810          406163
20000             2 ms               378066          507669
24000             3 ms              450831          693508
28000             3 ms              536933         836353
32000             3 ms              603210         884391
36000             4 ms              719579         1033594
40000             5 ms              795091         1287080
```

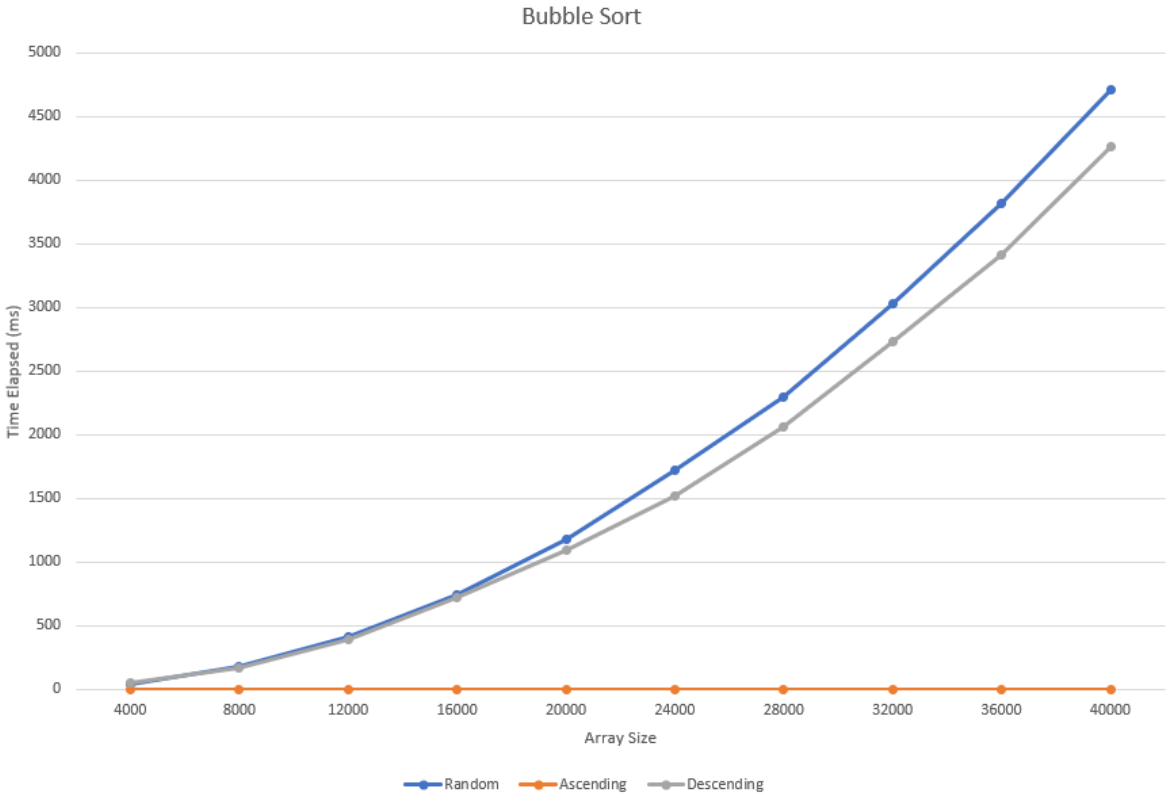
Ascending Array Analysis

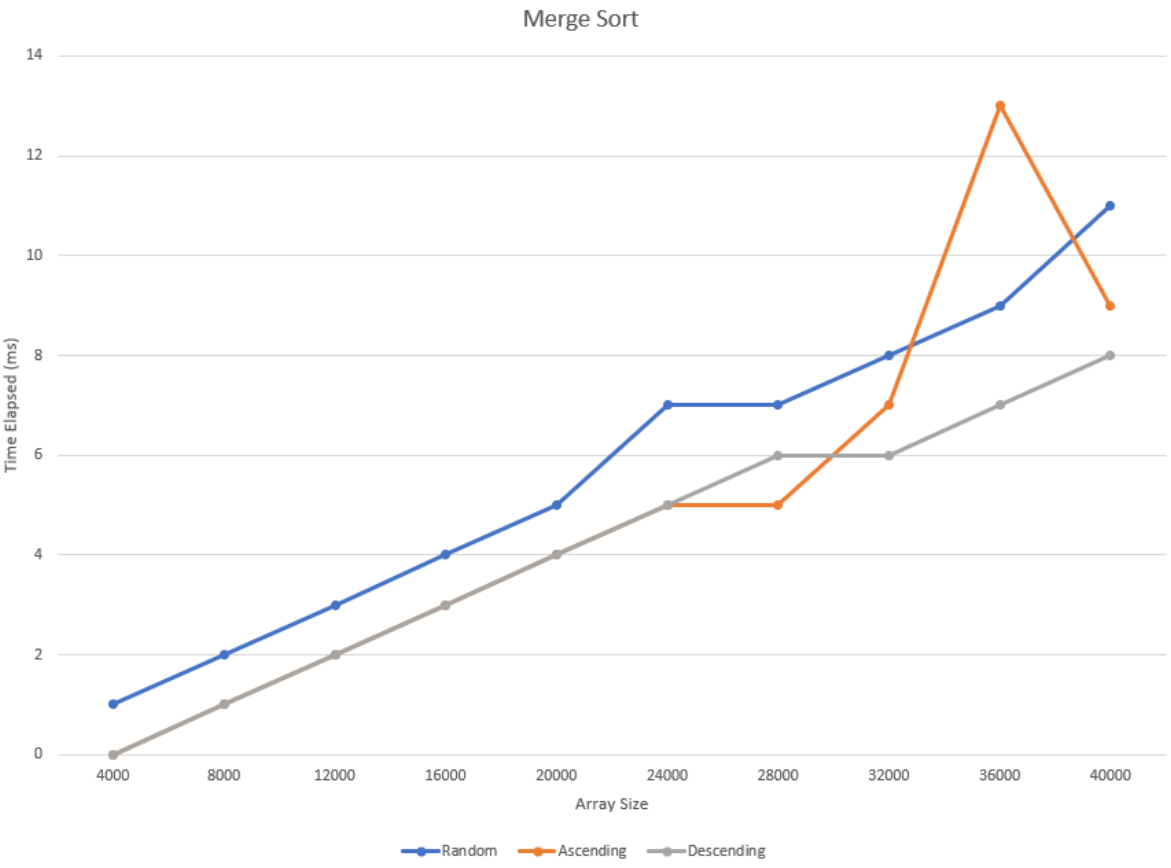
```
------------------------------------------------------------
Ascending Array Bubble Sort Analysis
Array Size        Elapsed Time        compCount       moveCount
4000              0 ms                3999            0
8000              0 ms                7999            0
12000             0 ms                11999           0
16000             0 ms                15999           0
20000             0 ms                19999           0
24000             0 ms                23999           0
28000             0 ms                27999           0
32000             0 ms                31999           0
36000             0 ms                35999           0
40000             0 ms                39999           0

------------------------------------------------------------
Ascending Array Merge Sort Analysis
Array Size        Elapsed Time        compCount       moveCount
4000              0 ms                32175           95808
8000              1 ms                68351           207616
12000             2 ms                108303          327232
16000             3 ms                144703          447232
20000             4 ms                188015          574464
24000             5 ms                228607          702464
28000             5 ms                268719          830464
32000             7 ms                305407          958464
36000             13 ms               351183          1092928
40000             9 ms                396031          1228928

------------------------------------------------------------
Ascending Array Quick Sort Analysis
Array Size        Elapsed Time        compCount       moveCount
4000              17 ms               8005999         15996
8000              59 ms               32011999        31996
12000             149 ms              72017999        47996
16000             230 ms              128023999       63996
20000             356 ms              200029999       79996
24000             513 ms              288035999       95996
28000             716 ms              392041999       111996
32000             951 ms              512047999       127996
36000             1257 ms             648053999       143996
40000             1507 ms             800059999       159996
```

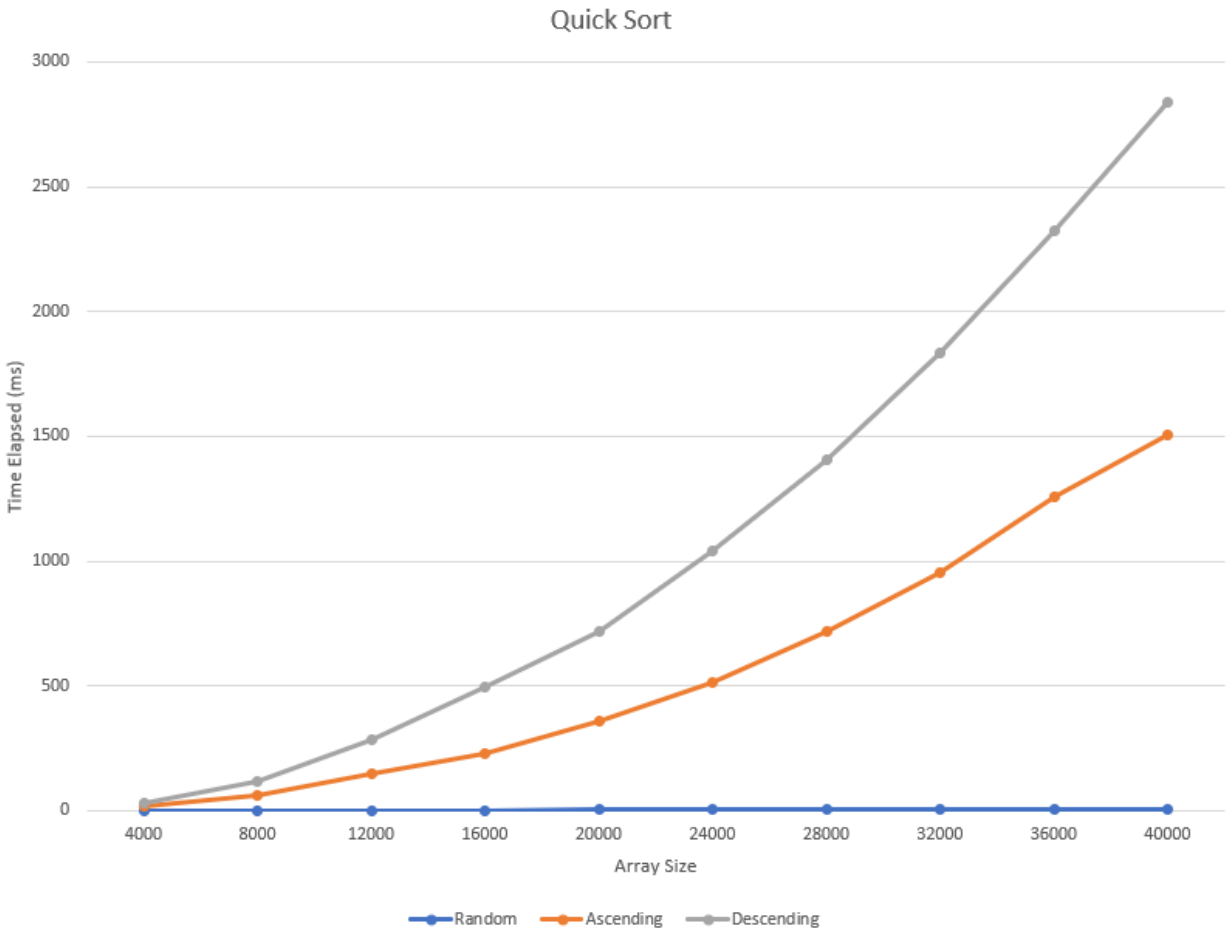
Descending Array Analysis

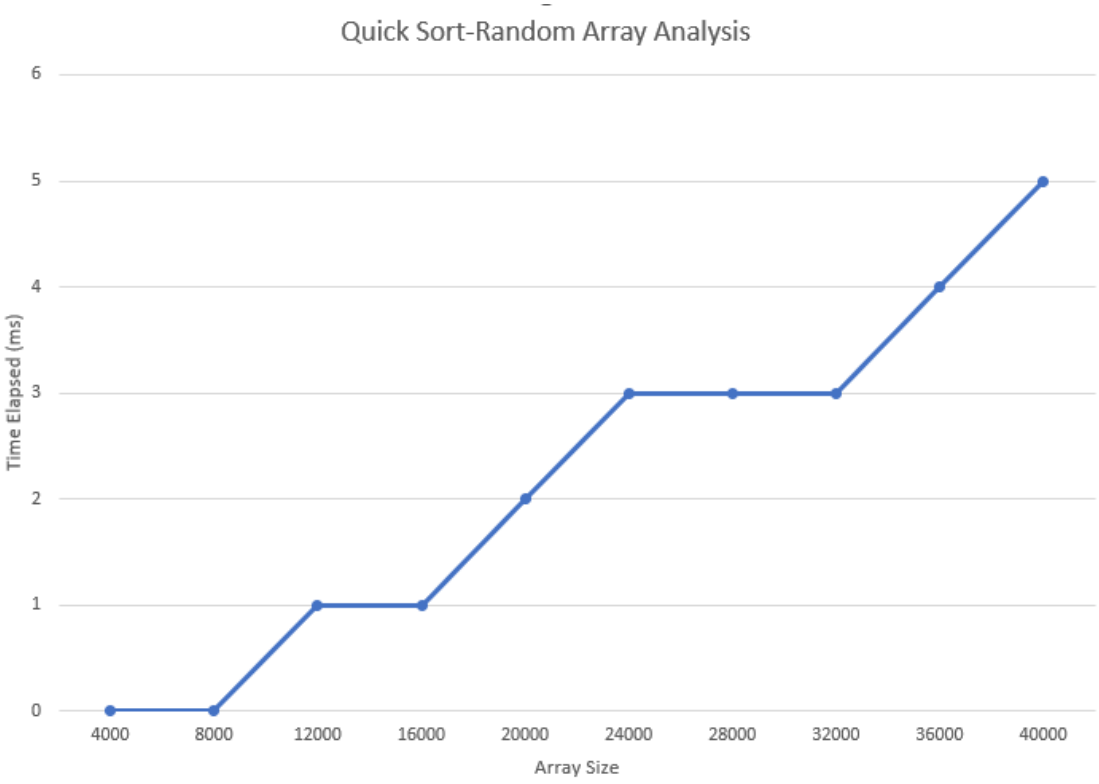
```
------------------------------------------------------------
Descending Array Bubble Sort Analysis
Array Size        Elapsed Time        compCount       moveCount
4000              46 ms               7998000          23994000
8000              172 ms              31996000          95988000
12000             395 ms              71994000          215982000
16000             721 ms              127992000          383976000
20000             1091 ms              199990000          599970000
24000             1520 ms             287988000          863964000
28000             2067 ms             391986000         1175958000
32000             2734 ms             511984000         1535952000
36000             3415 ms             647982000         1943946000
40000             4267 ms             799980000        -1895027296

------------------------------------------------------------
Descending Array Merge Sort Analysis
Array Size        Elapsed Time        compCount       moveCount
4000              0 ms               31727          95808
8000              1 ms              67455          207616
12000             2 ms              103311          327232
16000             3 ms              142911          447232
20000             4 ms              179215          574464
24000             5 ms             218623          702464
28000             6 ms             258511         830464
32000             6 ms             301823         958464
36000             7 ms             339279         1092928
40000             8 ms             378431         1228928

------------------------------------------------------------
Descending Array Quick Sort Analysis
Array Size        Elapsed Time        compCount       moveCount
4000              29 ms               8005999          12015996
8000              114 ms              32011999          48031996
12000             282 ms              72017999          108047996
16000             496 ms              128023999          192063996
20000             719 ms              200029999          300079996
24000             1043 ms             288035999          432095996
28000             1404 ms             392041999         588111996
32000             1835 ms             512047999         768127996
36000             2324 ms             648053999         972143996
40000             2838 ms             800059999         1200159996
```

Question 3 (Graphics with line chart and conclusion)






Quick Sort
3000
2500
2000
1500
1000
500
0
Time Elapsed (ms)
4000
8000
12000
16000
20000
24000
28000
32000
36000
40000
Array Size
Random
Ascending
Descending


Quick Sort-Random Array Analysis
6
5
4
3
2
1
0
Time Elapsed (ms)
4000
8000
12000
16000
20000
24000
28000
32000
36000
40000
Array Size

Conclusion

In theory, bubble sort's time complexity is O(n^2) in the average and worst cases and O(n) in the best case. As we can see from graphics that our experimental result matches for best case which is ascending (already sorted) array. In addition, we may expect descending (reverse ordered) array should be the one takes more time than random array (can be seen as a average case), however it is not the case for experimental results. On the other hand, both cases fit O(n^2) time complexity. This error may be caused of machine's RAM or compiler type.

Merge sort's algorithm time complexity is O($n * log_2 n$) in all (best, average and worst) cases. As we can see from the graph, all array types' (random, ascending, descending) complexities are quite close and they are equal for certain array sizes. Our experimental results match with theoretical results as we can obtain from graphs. Except random array, size of 4000 may seem to took 0 ms on table, but actually it is not the case. Merge sort is so fast that when we run it, it seems as 0 ms for ascending and descending array at size 4000. This error may be because of again machine's RAM or compiler type.

For quick sort we took first element of array as pivot in this experiment. Quick sort's time complexity is O(n^2) in worst case which are already sorted array and reverse ordered array, in adittion for best case and average case its time complexity is O($n * log_2 n$) . As we can see from graphs ascending and descending arrays are the one which took more time than random array. From this result we can say that, ascending and descending arrays are worst and random arrays are average cases for quick sort. Again for random array in size 4000 and 8000 time elapsed may seem as 0 ms, but actually it is not the case. It is so fast that it seems as 0 ms and again this error may be caused by machine's RAM or compiler type. Ascending and descending arrays took order of O(n^2) and random arrays took order of O($n * log_2 n$). To conclude, our experimental results match with theoretical ones for quick sort algorithm.