Sorting and Algorithm Efficiency
Erdem Ege Eroğlu
21601636
Section: 3
Assignment: 1

Question 1

a) Show that f(n) = 5n^3 + 4n^2 + 10 is O(n^4) by specifying appropriate c and n_0 values in Big-O definition.

We need to find two positive constants: c and n0 such that:

0 <= 5n^3 + 4n^2 + 10 <= cn^4 for all n >= n_0

0 <= 5/n + 4/n^2 + 10/n^4 <= cn^4 for all n >= n_0

Chose c = 19 and n_0 = 1

0 <= 5n^3 + 4n^2 + 10 <= 19n^4 for all n >= 1

b) Trace the following sorting algorithms to sort the array [24, 8, 51, 28, 20, 29, 21, 17, 38, 27] in ascending order. Use the array implementation of the algorithms as described in the textbook and show all major steps. (Insertion sort, Bubble sort)

Insertion Sort

24	8	51	28	20	29	21	17	38	27	Copy 8	(Initial array)
24	24	51	28	20	29	21	17	38	27	Shift 24	
8	24	51	28	20	29	21	17	38	27	Insert 8; Copy 51, Insert 51 on top of itself	
8	24	51	28	20	29	21	17	38	27	Copy 28	
8	24	51	51	20	29	21	17	38	27	Shift 51	
8	24	28	51	20	29	21	17	38	27	Insert 28	
8	24	28	51	20	29	21	17	38	27	Insert 28; Copy 20	
8	24	24	28	51	29	21	17	38	27	Shift 24, 28, 51	
8	20	24	28	51	29	21	17	38	27	Insert 20; Copy 29	
8	20	24	28	51	51	21	17	38	27	Shift 51	
8	20	24	28	29	51	21	17	38	27	Insert 29	
8	20	24	28	29	51	21	17	38	27	Copy 21	
8	20	24	24	28	29	51	17	38	27	Shift 24, 28, 29, 51	
8	20	21	24	28	29	51	17	38	27	Insert 21; Copy 17	
8	20	20	21	24	28	29	51	38	27	Shift 20, 21, 24, 28, 29, 51	
8	17	20	21	24	28	29	51	38	27	Insert 17; Copy 38	
8	17	20	21	24	28	29	51	51	27	Shift 51	
8	17	20	21	24	28	29	38	51	27	Insert 38	
8	17	20	21	24	28	29	38	51	27	Copy 27	
8	17	20	21	24	28	28	29	38	51	Shift 28, 29, 38, 51	

Sorted Array:

8	17	20	21	24	27	28	29	38	51	Insert 27

Bubble Sort

Initial array:

Pass 1

24	8	51	28	20	29	21	17	38	27
8	24	51	28	20	29	21	17	38	27
8	24	51	28	20	29	21	17	38	27
8	24	28	51	20	29	21	17	38	27
8	24	28	20	51	29	21	17	38	27
8	24	28	20	29	51	21	17	38	27
8	24	28	20	29	21	51	17	38	27
8	24	28	20	29	21	17	51	38	27
8	24	28	20	29	21	17	38	51	27
8	24	28	20	29	21	17	38	27	51

Pass 2

8	24	28	20	29	21	17	38	27	51
8	24	28	20	29	21	17	38	27	51
8	24	28	20	29	21	17	38	27	51
8	24	20	28	29	21	17	38	27	51
8	24	20	28	29	21	17	38	27	51
8	24	20	28	21	29	17	38	27	51
8	24	20	28	21	17	29	38	27	51
8	24	20	28	21	17	29	38	27	51
8	24	20	28	21	17	29	27	38	51

Pass 3

8	24	20	28	21	17	29	27	38	51
8	24	20	28	21	17	29	27	38	51
8	20	24	28	21	17	29	27	38	51
8	20	24	28	21	17	29	27	38	51
8	20	24	21	28	17	29	27	38	51
8	20	24	21	17	28	29	27	38	51
8	20	24	21	17	28	29	27	38	51
8	20	24	21	17	28	27	29	38	51

Pass 4

8	20	24	21	17	28	27	29	38	51
8	20	24	21	17	28	27	29	38	51
8	20	24	21	17	28	27	29	38	51
8	20	21	24	17	28	27	29	38	51
8	20	21	17	24	28	27	29	38	51
8	20	21	17	24	28	27	29	38	51
8	20	21	17	24	27	28	29	38	51

Pass 5

8	20	21	17	24	27	28	29	38	51
8	20	21	17	24	27	28	29	38	51
8	20	21	17	24	27	28	29	38	51
8	20	17	21	24	27	28	29	38	51
8	20	17	21	24	27	28	29	38	51
8	20	17	21	24	27	28	29	38	51

Pass 6

8	20	17	21	24	27	28	29	38	51
8	20	17	21	24	27	28	29	38	51
8	17	20	21	24	27	28	29	38	51
8	17	20	21	24	27	28	29	38	51
8	20	17	21	24	27	28	29	38	51

Pass 7

8	20	17	21	24	27	28	29	38	51
8	20	17	21	24	27	28	29	38	51
8	17	20	21	24	27	28	29	38	51
8	20	17	21	24	27	28	29	38	51

Pass 8

8	20	17	21	24	27	28	29	38	51
8	20	17	21	24	27	28	29	38	51
8	17	20	21	24	27	28	29	38	51

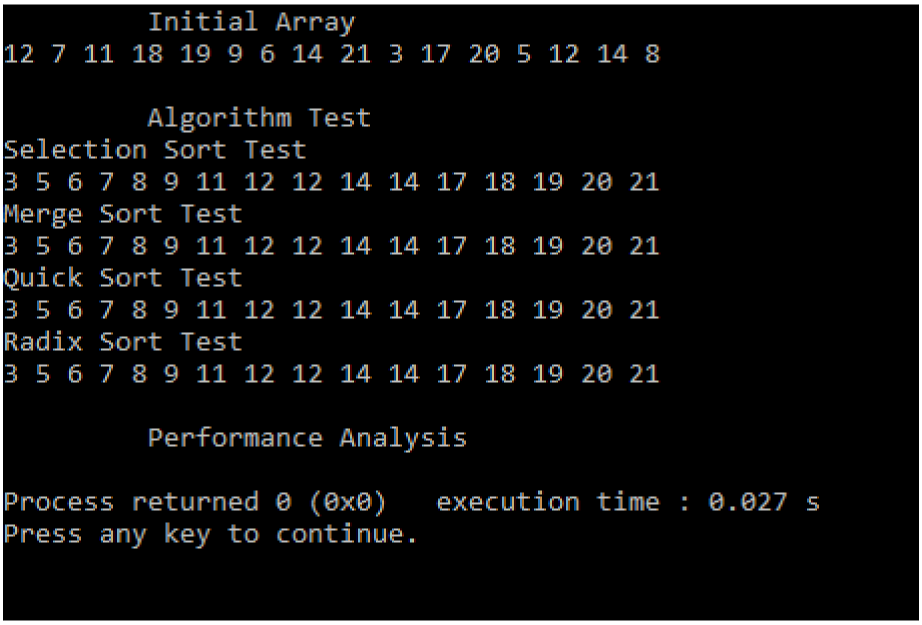
Pass 9

8	17	20	21	24	27	28	29	38	51
8	17	20	21	24	27	28	29	38	51

Pass 10

8	17	20	21	24	27	28	29	38	51

Question 2

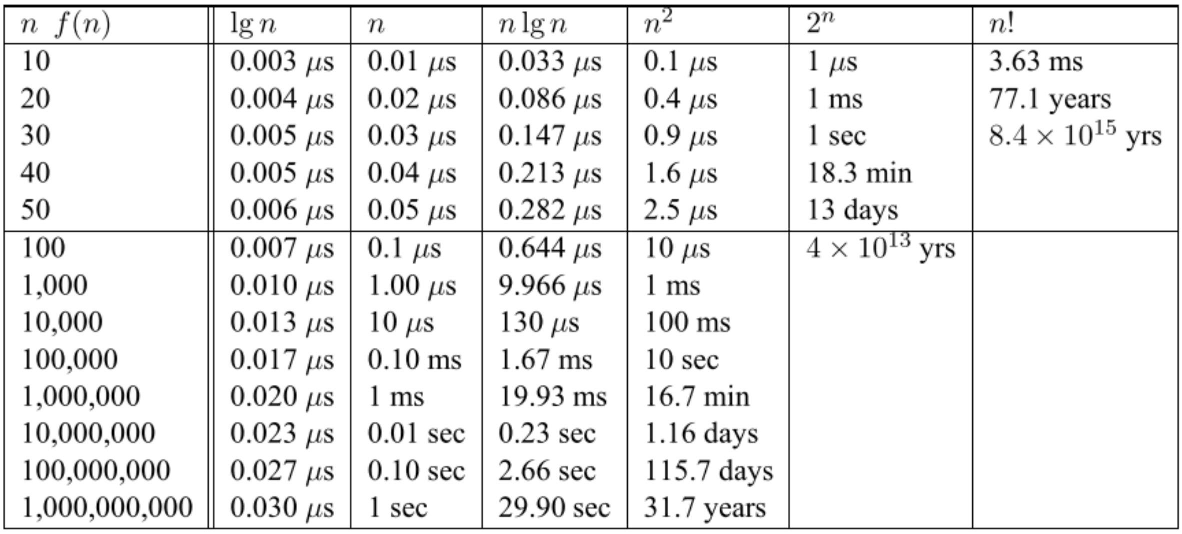


```
         Initial Array
12 7 11 18 19 9 6 14 21 3 17 20 5 12 14 8

         Algorithm Test
Selection Sort Test
3 5 6 7 8 9 11 12 12 14 14 17 18 19 20 21
Merge Sort Test
3 5 6 7 8 9 11 12 12 14 14 17 18 19 20 21
Quick Sort Test
3 5 6 7 8 9 11 12 12 14 14 17 18 19 20 21
Radix Sort Test
3 5 6 7 8 9 11 12 12 14 14 17 18 19 20 21

         Performance Analysis

Process returned 0 (0x0)   execution time : 0.027 s
Press any key to continue.
```

Question 3

In this homework we are supposed to observe four sorting algorithms. Those are Selection-sort, Merge-sort, Quick-sort, Radix-sort. They are examined under different conditions to find out wort, best, and average cases. In order to come up with those three observations, we were supposed to generate random, ascending, and descending arrays. Random arrays are for average cases, descending arrays are for the worst cases, and eventually, ascending arrays are for the best cases.

The first eye-caching observation is that Selection-sort is the less effective way to sort an array. It has O(n^2) growth rate; thus, this makes it the slowest one. Secondly, the growth rate of Merge-sort algorithm is O(n*logn) for all arrays. Therefore, Merge-sort algorithm has the same rate of growth for each case (best, average, worst). Moreover, the worst case of Quick-sort is O(n^2) and average case is O(n*logn). Finally, we can detect that Radix-sort is the best way for sorting. Because, Radix-sort's growth rate is O(n) for all three cases (random, ascending, descending). This makes Radix-sort much preferable that the other three algorithms.



n $f(n)$	$\lg n$	n	$n \lg n$	n^2	2^n	$n!$
10	0.003 μs	0.01 μs	0.033 μs	0.1 μs	1 μs	3.63 ms
20	0.004 μs	0.02 μs	0.086 μs	0.4 μs	1 ms	77.1 years
30	0.005 μs	0.03 μs	0.147 μs	0.9 μs	1 sec	8.4×10^{15} yrs
40	0.005 μs	0.04 μs	0.213 μs	1.6 μs	18.3 min	
50	0.006 μs	0.05 μs	0.282 μs	2.5 μs	13 days	
100	0.007 μs	0.1 μs	0.644 μs	10 μs	4×10^{13} yrs	
1,000	0.010 μs	1.00 μs	9.966 μs	1 ms		
10,000	0.013 μs	10 μs	130 μs	100 ms		
100,000	0.017 μs	0.10 ms	1.67 ms	10 sec		
1,000,000	0.020 μs	1 ms	19.93 ms	16.7 min		
10,000,000	0.023 μs	0.01 sec	0.23 sec	1.16 days		
100,000,000	0.027 μs	0.10 sec	2.66 sec	115.7 days		
1,000,000,000	0.030 μs	1 sec	29.90 sec	31.7 years		

This is taken from slides