CS-202

Fundamental Structures of Computer Science II

Section: 1

Homework 1

Mehmet Hasat Serinkan

21901649

22.06.2022

Question 1:

Question 1)

a) $0 \leq 3n^4 + 4n^3 + n \leq cn^5$ for all $n \geq n_0$

(divide by n^5

$0 \leq \frac{3}{n} + \frac{4}{n^2} + \frac{1}{n^4} \leq c$ for all $n \geq n_0$

($n_0 = 1$

$0 \leq 3 + 4 + 1 \leq c$ for all $n \geq 1$

(then we can choose $c = 8$ and $n_0 = 1$

Answer: $3n^4 4n^3 + n \leq 8n^5$ for all $n \geq 1$

Question 2:

Selection Sort

(unsorted | sorted)

Initial array: 31, 17, 58, 35, 45, 27, 36, 28, 24, 34

After 1st swap: 31, 17, 34, 35, 45, 27, 36, 28, 24, 58

After 2nd swap: 31, 17, 34, 35, 24, 27, 36, 28, 45, 58

After 3rd swap: 31, 17, 34, 35, 24, 27, 28, 36, 45, 58

After 4th swap: 31, 17, 34, 28, 24, 27, 35, 36, 45, 58

After 5th swap: 31, 17, 27, 28, 24, 34, 35, 36, 45, 58

After 6th swap: 24, 17, 27, 28, 31, 34, 35, 36, 45, 58

After 7th swap: 24, 17, 27, 28, 31, 34, 35, 36, 45, 58

After 8th swap: 24, 17, 27, 28, 31, 34, 35, 36, 45, 58

After 9th swap: 17, 24, 27, 28, 31, 34, 35, 36, 45, 58

Array is sorted!

Insertion Sort

(sorted | unsorted)

original Array: 31, 17, 58, 35, 45, 27, 36, 28, 24, 34

After pass1: 17, 31, 58, 35, 45, 27, 36, 28, 24, 34

After pass2: 17, 31, 58, 35, 45, 27, 36, 28, 24, 34

After pass3: 17, 31, 35, 58, 45, 27, 36, 28, 24, 34

After pass4: 17, 31, 35, 45, 58, 27, 36, 28, 24, 34

After pass5: 17, 27, 31, 35, 45, 58, 36, 28, 24, 34

After pass6: 17, 27, 31, 35, 36, 45, 58, 28, 24, 34

After pass7: 17, 27, 28, 31, 35, 36, 45, 58, 24, 34

After pass8: 17, 24, 27, 28, 31, 35, 36, 45, 58, 34

After pass9: 17, 24, 27, 28, 31, 34, 35, 36, 45, 58

Array is sorted!

Question 2-C:

```
CompCount and MoveCounts of bubble sort are 114 and 180
11 13 14 15 16 17 19 20 21 22 23 25 26 27 28 29

CompCount and MoveCounts of merge sort are 46 and 128
11 13 14 15 16 17 19 20 21 22 23 25 26 27 28 29

CompCount and MoveCounts of quick sort are 45 and 93
11 13 14 15 16 17 19 20 21 22 23 25 26 27 28 29


Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

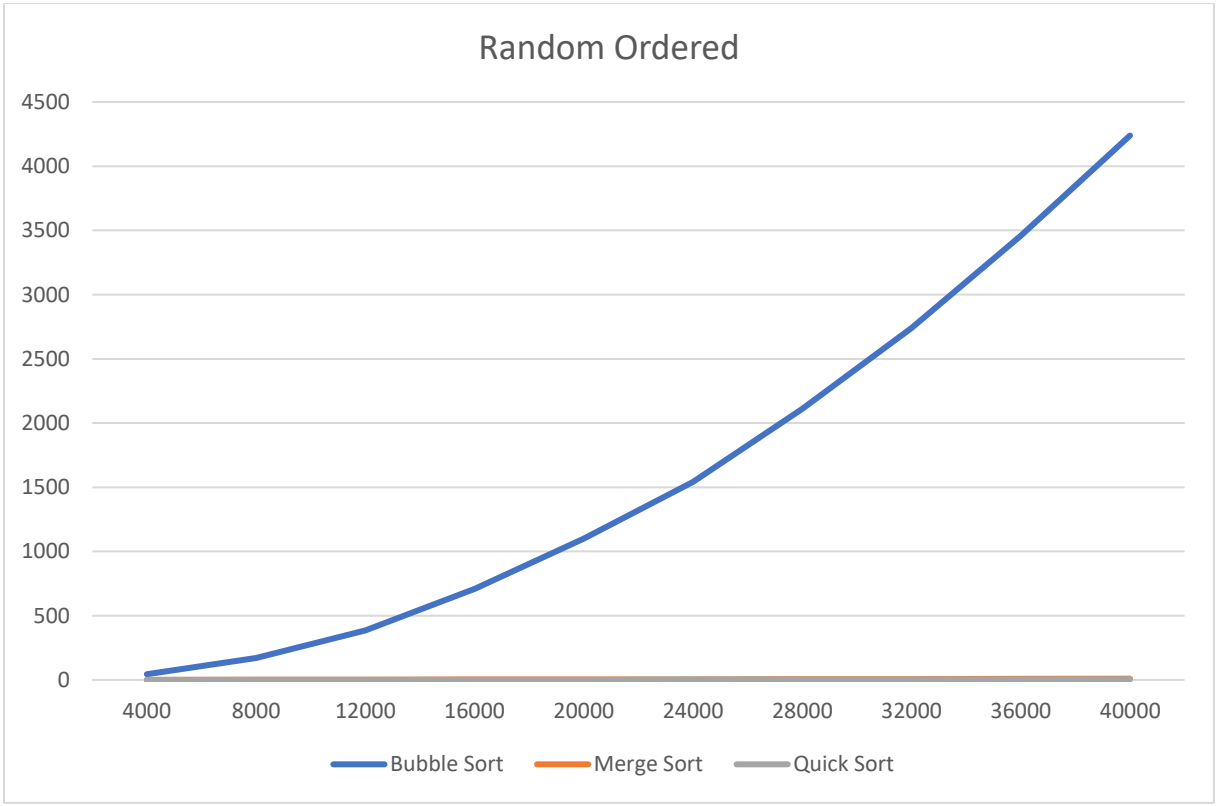
Question 2-D:

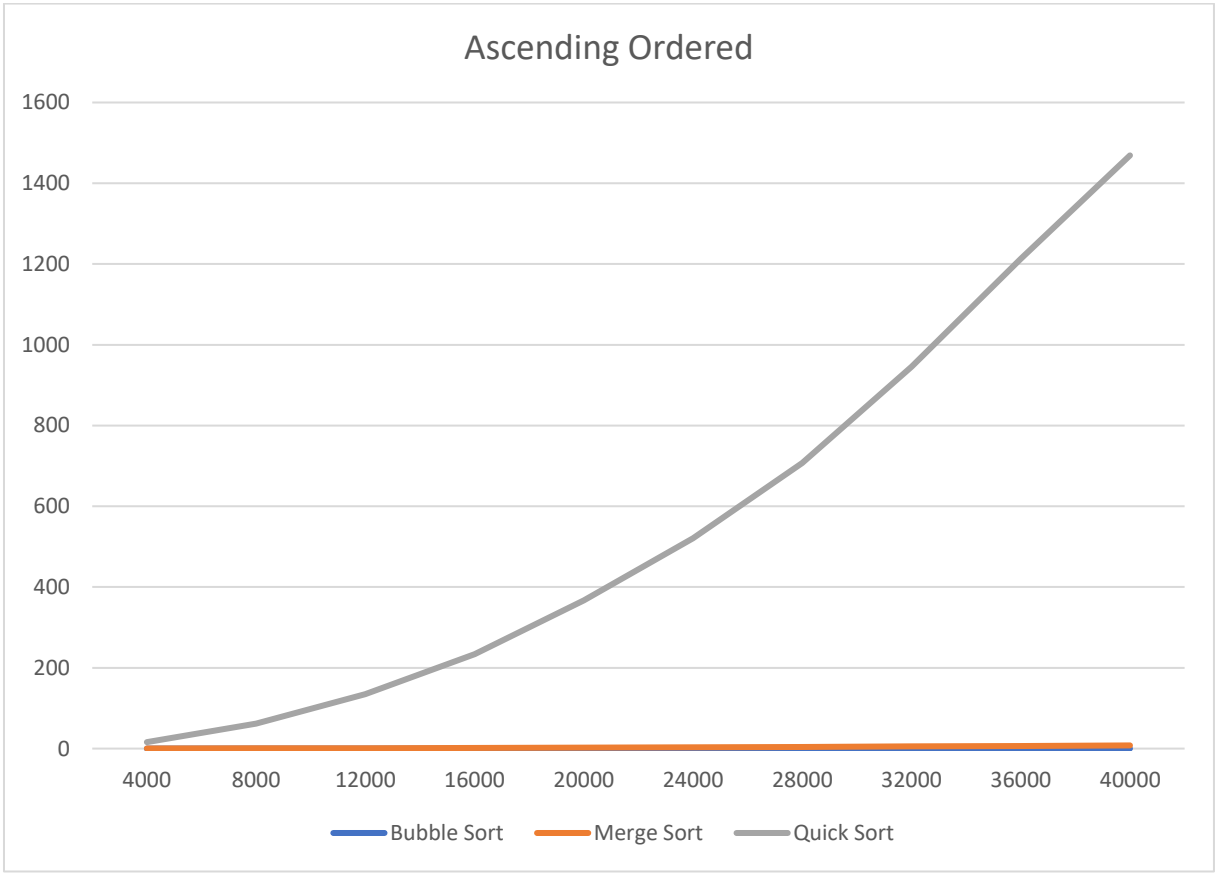
```
RANDOM ARRAYS
---------------------------------
Analysis of Bubble Sort
Array SizeElapsed   timecompCount      moveCount
4000   43.631 ms        7991330         11800752
8000   191.081 ms       31985847        47740428
12000   448.481 ms      71991654        106724217
16000   813.171 ms      127991864        192136425
20000   1295.87 ms      199960839        298648782
24000   1845.58 ms      287979485        432343086
28000   2479.16 ms      391972470        585888090
32000   3182.08 ms      511978435        766969419
36000   4072.61 ms      647918454        971019897
40000   5046.97 ms      799958055        1200261435
---------------------------------
Analysis of Merge Sort
Array SizeElapsed   timecompCount   moveCount
4000   0.47 ms          42749          95808
8000   1.052 ms         93611          207616
12000   1.717 ms         147777        327232
16000   2.965 ms         203229        447232
20000   3.467 ms         260805        574464
24000   4.436 ms         319421        702464
28000   5.731 ms         378888        830464
32000   6.966 ms         438653        958464
36000   8.427 ms         499795        1092928
40000   9.877 ms         561892        1228928
---------------------------------
Analysis of Quick Sort
Array SizeElapsed   timecompCount   moveCount
4000   0.379 ms         54935         82434
8000   0.865 ms         128197        211005
12000   1.266 ms         187098        288915
16000   1.839 ms         293001        472743
20000   2.28 ms          347508        542730
24000   2.737 ms         393787        566628
28000   3.25 ms          515693        747768
32000   3.736 ms         555776        865770
36000   4.301 ms         639201        971397
40000   4.732 ms         699840        1108170
```

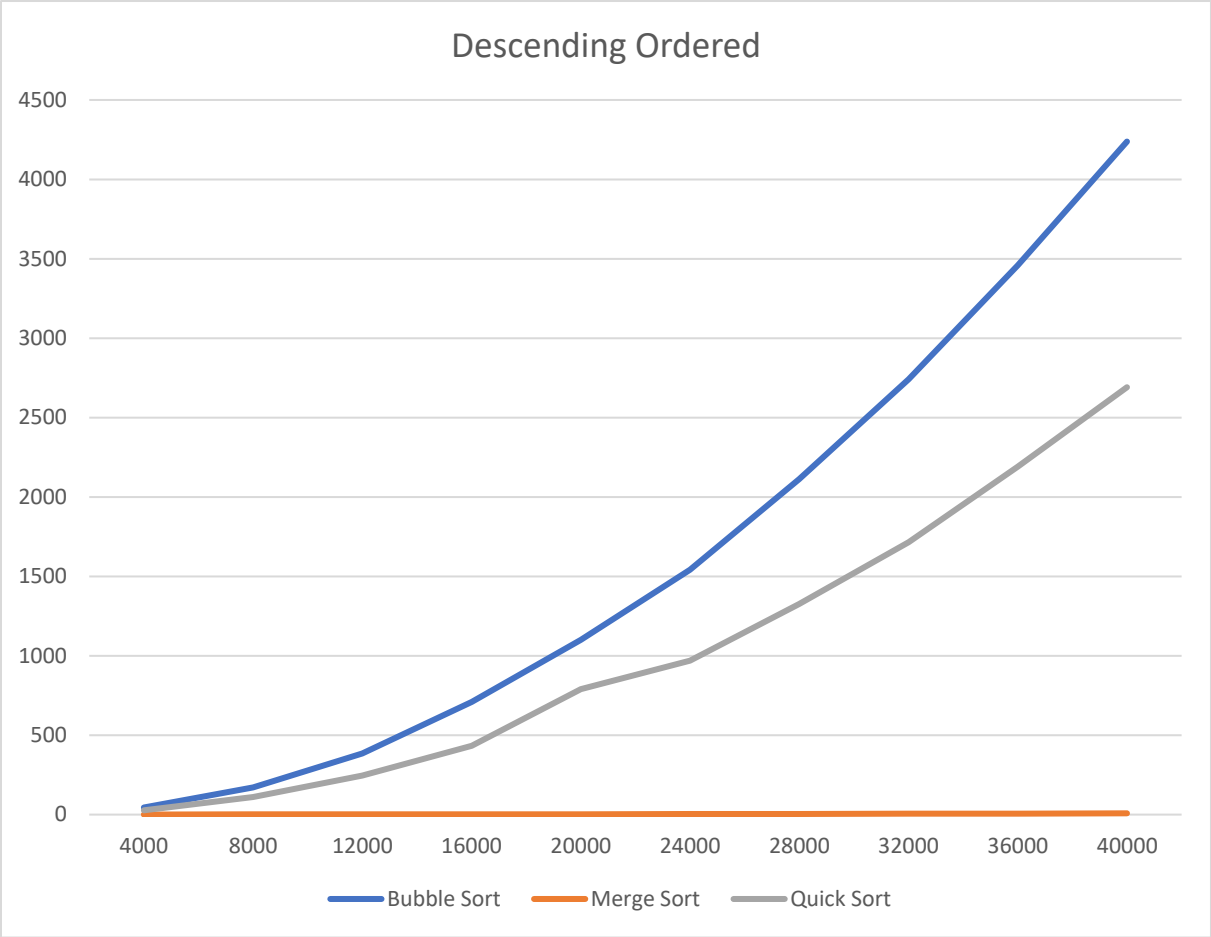
```
ASCENDING ARRAYS
---------------------------------
Analysis of Bubble Sort
Array SizeElapsed   timecompCount   moveCount
4000   0.009 ms        3999         0
8000   0.016 ms       7999          0
12000   0.028 ms      11999         0
16000   0.033 ms      15999         0
20000   0.041 ms      19999         0
24000   0.051 ms      23999         0
28000   0.059 ms      27999         0
32000   0.068 ms      31999         0
36000   0.079 ms      35999         0
40000   0.082 ms      39999         0
---------------------------------
Analysis of Merge Sort
Array SizeElapsed   timecompCount   moveCount
4000   0.285 ms         24176          95808
8000   0.763 ms         52352          207616
12000   1.195 ms         84304        327232
16000   1.786 ms         112704         447232
20000   2.509 ms         148016         574464
24000   3.377 ms         180608         702464
28000   4.337 ms         212720         830464
32000   5.309 ms         241408         958464
36000   6.477 ms         279184         1092928
40000   7.866 ms         316032         1228928
---------------------------------
Analysis of Quick Sort
Array SizeElapsed   timecompCount       moveCount
4000   16.091 ms         7998000          11997
8000   61.609 ms         31996000          23997
12000   135.007 ms         71994000         35997
16000   234.312 ms         127992000        47997
20000   366.702 ms         199990000        59997
24000   521.264 ms         287988000        71997
28000   707.844 ms         391986000        83997
32000   946.588 ms         511984000        95997
36000   1213.03 ms         647982000        107997
40000   1469.96 ms         799980000        119997
```

```
DESCENDING ARRAYS
---------------------------------
Analysis of Bubble Sort
Array SizeElapsed   timecompCount       moveCount
4000   44.619 ms         7998000          23994000
8000   172.005 ms        31996000         95988000
12000   384.638 ms       71994000         215982000
16000   709.402 ms       127992000         383976000
20000   1100.22 ms       199990000         599970000
24000   1541.35 ms       287988000         863964000
28000   2111.96 ms       391986000         1175958000
32000   2741.03 ms       511984000         1535952000
36000   3459.38 ms       647982000         1943946000
40000   4238.91 ms       799980000         2399940000
---------------------------------
Analysis of Merge Sort
Array SizeElapsed   timecompCount   moveCount
4000   0.274 ms          23728          95808
8000   0.835 ms          51456          207616
12000   1.241 ms          79312        327232
16000   1.936 ms          110912         447232
20000   2.376 ms          139216         574464
24000   3.809 ms          170624         702464
28000   4.581 ms          202512         830464
32000   5.429 ms          237824         958464
36000   6.591 ms          267280         1092928
40000   7.996 ms          298432         1228928
---------------------------------
Analysis of Quick Sort
Array SizeElapsed   timecompCount           moveCount
4000   27.474 ms          7998000           12011997
8000   110.53 ms          31996000          48023997
12000   245.225 ms          71994000          108035997
16000   434.396 ms          127992000         192047997
20000   689.778 ms          199990000         300059997
24000   969.706 ms          287988000         432071997
28000   1327.91 ms          391986000         588083997
32000   1716.59 ms          511984000         768095997
36000   2190.68 ms          647982000          972107997
40000   2691.11 ms          799980000         1200119997
```

Question 3:


Random Ordered
4500
4000
3500
3000
2500
2000
1500
1000
500
0
4000
8000
12000
16000
20000
24000
28000
32000
36000
40000
Bubble Sort
Merge Sort
Quick Sort


Ascending Ordered
1600
1400
1200
1000
800
600
400
200
0
4000
8000
12000
16000
20000
24000
28000
32000
36000
40000
Bubble Sort
Merge Sort
Quick Sort



ALGORITHM	TIME COMPEXITY		
	Worst case	Average Case	Best Case
Bubble Sort	O(n^2)	O(n^2)	O(n)
Merge Sort	O(nlogn)	O(nlogn)	O(nlogn)
Quick Sort	O(n^2)	O(nlogn)	O(nlogn)

From the graphs and performance analysis table, it can be observed that merge sort is working best. Merge Sort's time complexity is O(nlogn) for every case. This can be seen from the graphs. Therefore, it can be stated that the theoretical and empirical results are the same for the merge sort. Moreover, it is the fastest among these algorithms.

One can say that bubble sort is one of the worst algorithms for sorting. Bubble sort's worst case happens when one tries to sort a descending ordered array. The main reason is that descending ordered arrays cause the maximum amount of data moves and key comparisons. This causes O(n^2) time complexity. It can be easily seen from the graph that bubble sort has

O(n^2) time complexity for descending ordered array. For a random array, which is an average case, the execution times go parabolically. It means it has O(n^2) time complexity on the average case. In the best case, the bubble sort's execution time is almost 0, but it can be seen that it is increasing linearly. Hence, its time complexity is O(n). It shows that the empirical and theoretical results are the same for bubble sort.

Quick Sort's worst case happens when the array is sorted and the first element of the array is chosen as the pivot. It causes O(n^2) time complexity. In the sortings, the pivot has always been the first element of the array. Therefore, quick sorting of ascending or descending arrays is slow. It can be observed from the graph that quick Sort has O(n^2)time complexity for ascending or descending arrays. Quick Sort has O(nlogn) time complexity for the average and best case. This time complexity happens when quick sorting a randomly ordered array. It is almost 0; however, from the execution timetable, it can be seen that it is growing a little bit. Henceforth, the empirical and theoretical results are almost identical for quick Sort.

- Bubble sort's time complexity changes when sorting an ascending ordered array. The main reason for this is array is already sorted. The algorithm checks for n items, and it stops. Therefore, it is O(n).
- Merge sort's time complexity isn't changing for any case.
- Quick Sort's time complexity changes when sorting ordered arrays, and the pivot is the first element. The logical reason for this it makes comparisons.