QUESTION #1

CS 202 - HW 01

Emre Karataş
22001641
CS 202-01

Question 1:

a) According to big-O definition, $T(n) = O(f(n))$

$T(n) \leq c \cdot f(n)$ when $n \geq n_0$

$0 \leq 3n^3 + 4n^2 + 2n \leq c \cdot n^3$

$0 \leq 3 + \frac{4}{n} + \frac{2}{n^2} \leq c$

select $c = 4$ $n_0 = 5$

$3n^3 + 4n^2 + 2n \leq 4n^3$ for $n \geq 5$

b)

$T(1) = \Theta(1) = 1$

$T(n) = T(n-1) + n^2 + \Theta(1)$

$T(n) = [T(n-2) + (n-1)^2 + \Theta(1)] + n^2 + \Theta(1)$

$\vdots$

after k steps

$T(n) = T(n-k) + (n-k-1)^2 + (n-k-2)^2 + \dots + (n-1)^2 + n^2 + \Theta^k(1)$

set $k = n-1$

$T(n) = \underbrace{T(1)}_{1} + 2^2 + 3^2 + \dots + n^2 + \underbrace{\Theta^{n-1}(1)}_{1^{n-1} = 1}$

$T(n) = 2 + \sum_{k=2}^{n} k^2$

$T(n) = 2 + \sum_{k=1}^{n} k^2 - 1 = \underbrace{1}_{} + \frac{n(n+1)(2n+1)}{6}$

$\boxed{\Theta(1) + \frac{2n^3 + 2n^2 + n}{6}}$

Scanned with CamScanner

Set $\theta(1) = T(1) = 1$

$$T(n) = 2T(n/2) + n/2 + \theta(1)$$

$$T(n) = 2\left[2T(n/4) + n/4 + \theta(1)\right] + n/2 + \theta(1)$$

$\therefore$ after k steps

$$T(n) = 2^k T(n/2^k) + 2^{k-1}\frac{n}{2^k} + 2^k\theta(1) + n/2 + \theta(1)$$

$$T(n) = 2^k T(n/2^k) + \frac{n}{2} + 2^k\theta(1) + \frac{n}{2} + \theta(1)$$

$$T(n) = 2^k T(n/2^k) + n + \theta(1)(2^k + 1)$$

set $n = 2^k$

$$T(n) = n\underbrace{T(1)}_{\theta(1)} + n + \theta(1)(n+1)$$

$$T(n) = \theta(1)(n + n + 1) + n$$

$$\boxed{T(n) = (2n+1)\,\theta(1) + n}$$

CS Scanned with CamScanner

c) Tracing the Sorting Algorithms

1) Selection Sort:

Original array is: [21, 9, 58, 28, 36, 18, 27, 19, 4, 25]

Select the biggest element from unsorted list and add to the sorted list.

① Array: [21, 9, 28, 36, 18, 27, 19, 4, 25, 58]
unsorted | sorted

② Array: [21, 9, 28, 18, 27, 19, 4, 25, 36, 58]

③ Array: [21, 9, 18, 27, 19, 4, 25, 28, 36, 58]

④ Array: [21, 9, 18, 19, 4, 25, 27, 28, 36, 58]

⑤ Array: [9, 18, 19, 4, 21, 25, 27, 28, 36, 58]

⑥ Array: [9, 18, 4, 19, 21, 25, 27, 28, 36, 58]

⑦ Array: [9, 4, 18, 19, 21, 25, 27, 28, 36, 58]

⑧ Array: [4, 9, 18, 19, 21, 25, 27, 28, 36, 58]

– Array is sorted ascending order –

2) Insertion Sort

Original Array: [21, 9, 58, 28, 36, 18, 27, 19, 4, 25]
sorted | unsorted

① Array: [9, 21 | 58, 28, 36, 18, 27, 19, 4, 25]

② Array: [9, 21, 58 | 28, 36, 18, 27, 19, 4, 25]

③ Array: [9, 21, 28, 58 | 36, 18, 27, 19, 4, 25]

④ Array: [9, 21, 28, 36, 58 | 18, 27, 19, 4, 25]

⑤ Array: [9, 18, 21, 28, 36, 58 | 27, 19, 4, 25]

⑥ Array: [9, 18, 21, 27, 28, 36, 58 | 19, 4, 25]

⑦ Array: [9, 18, 19, 21, 27, 28, 36, 58 | 4, 25]

⑧ Array: [4, 9, 18, 19, 21, 27, 28, 36, 58 | 25]

⑨ Array: [4, 9, 18, 19, 21, 25, 27, 28, 36, 58 |]

– Array is sorted in ascending order.

CS Scanned with CamScanner

QUESTION #2

Check Dijkstra server fort part A and part B. Code is uploaded there with executable hw1 makefile.

PART C:

emre.karatas@dijkstra:~

```
Array given in the lab instruction:
40 25 29 56 37 27 24 32 79 12 35 38 23 31 33 26
---- BUBBLE SORT ----
Number of comparison: 126
Number of moves: 204
12 23 24 25 26 27 29 31 32 33 35 37 38 40 56 79
---- MERGE SORT ----
Number of comparison: 46
Number of moves: 128
12 23 24 25 26 27 29 31 32 33 35 37 38 40 56 79
---- QUICK SORT ----
Number of comparison: 62
Number of moves: 42
12 23 24 25 26 27 29 31 32 33 35 37 38 40 56 79
```

In the next page; array size, elapsed time, CompCount and moveCount can be seen for different type of arrays with different type of sorting algorithms (bubble sort, merge sort and quick sort).

PART D:

For Random Arrays:

emre.karatas@dijkstra:~

```
RANDOM ARRAYS
-----ANALYSIS OF BUBBLE SORT-----
Array Size          Time Elapsed        CompCount           MoveCount
4000                96    ms            7999984             11960313
8000                349   ms            31997559            35063799
12000               693   ms            71994974            59393994
16000               1122  ms            127936747           83933256
20000               1630  ms            199975284           105855156
24000               2231  ms            287939610           130399392
28000               2912  ms            391967640           154786026
32000               3686  ms            511926747           177603222
36000               4544  ms            647863154           203997276
40000               5501  ms            799912584           229703727
44000               6513  ms            967900229           247219680
48000               7630  ms            1151841894          272895573
-----------------------------------
-----ANALYSIS OF MERGE SORT-----
Array Size          Time Elapsed        CompCount           MoveCount

4000                1     ms            42798               95808
8000                1     ms            78157               207616
12000               2     ms            112353              327232
16000               3     ms            147930              447232
20000               4     ms            181770              574464
24000               5     ms            217348              702464
28000               6     ms            254121              830464
32000               7     ms            293990              958464
36000               8     ms            330468              1092928
40000               9     ms            365579              1228928
44000               10    ms            400487              1364928
48000               11    ms            438334              1500928
-----------------------------------
-----ANALYSIS OF QUICK SORT-----
Array Size          Time Elapsed        CompCount           MoveCount

4000                1     ms            117670              21813
8000                4     ms            831845              34927
12000               12    ms            2558001             44746
16000               26    ms            5897758             60433
20000               49    ms            11220896            72771
24000               82    ms            18830387            81214
28000               118   ms            27453876            89311
32000               174   ms            40588241            92337
36000               265   ms            61900966            102866
40000               307   ms            71782508            121487
44000               451   ms            105717520           124544
48000               588   ms            138014666           130983
```

For Ascending Random Arrays:

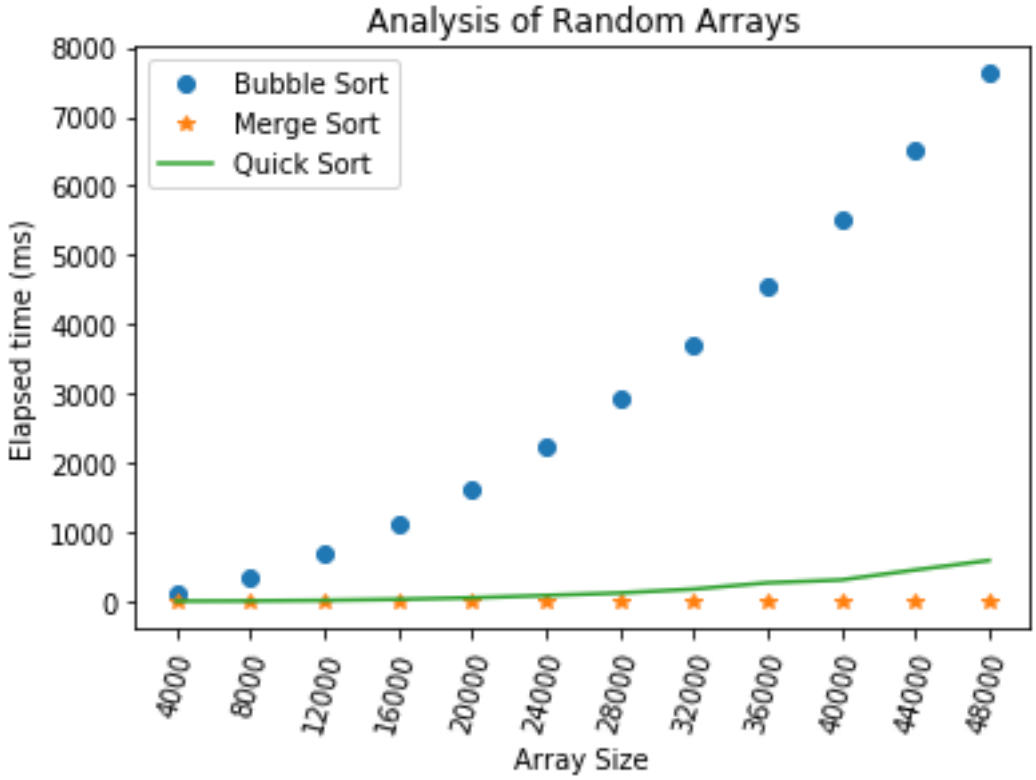
emre.karatas@dijkstra:~

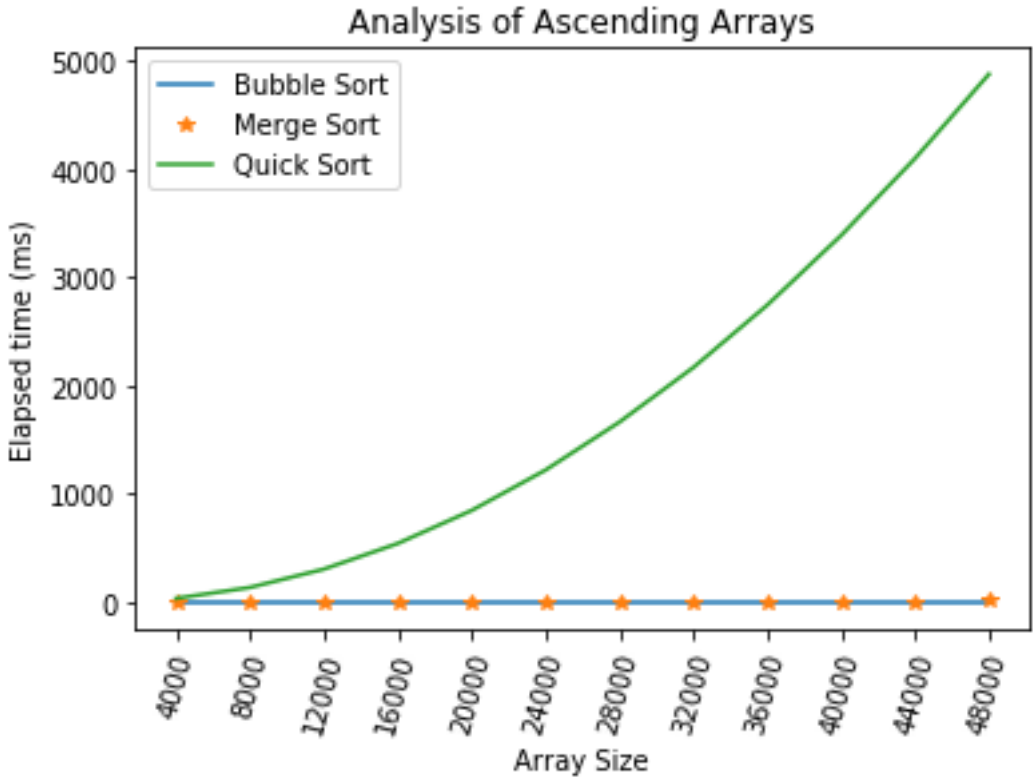
```
ASCENDING RANDOM ARRAYS
-----ANALYSIS OF BUBBLE SORT-----
Array Size          Time Elapsed        CompCount           MoveCoun
4000                0     ms            4000                0
8000                0     ms            8000                0
12000               0     ms            12000               0
16000               0     ms            16000               0
20000               0     ms            20000               0
24000               0     ms            24000               0
28000               0     ms            28000               0
32000               0     ms            32000               0
36000               0     ms            36000               0
40000               0     ms            40000               0
44000               0     ms            44000               0
48000               0     ms            48000               0
-----------------------------------
-----ANALYSIS OF MERGE SORT-----
Array Size          Time Elapsed        CompCount           MoveCoun
4000                0     ms            24386               95808
8000                1     ms            52791               207616
12000               2     ms            84682               327232
16000               3     ms            113515              447232
20000               4     ms            148827              574464
24000               5     ms            181448              702464
28000               6     ms            213883              830464
32000               6     ms            243056              958464
36000               7     ms            280844              1092928
40000               8     ms            317700              1228928
44000               9     ms            353691              1364928
48000               10    ms            386867              1500928
-----------------------------------
-----ANALYSIS OF QUICK SORT-----
Array Size          Time Elapsed        CompCount           MoveCoun
4000                34    ms            7998000             3999
8000                135   ms            31996000            7999
12000               304   ms            71994000            11999
16000               541   ms            127992000           15999
20000               846   ms            199990000           19999
24000               1219  ms            287988000           23999
28000               1662  ms            391986000           27999
32000               2166  ms            511984000           31999
36000               2742  ms            647982000           35999
40000               3385  ms            799980000           39999
44000               4096  ms            967978000           43999
48000               4874  ms            1151976000          47999
```

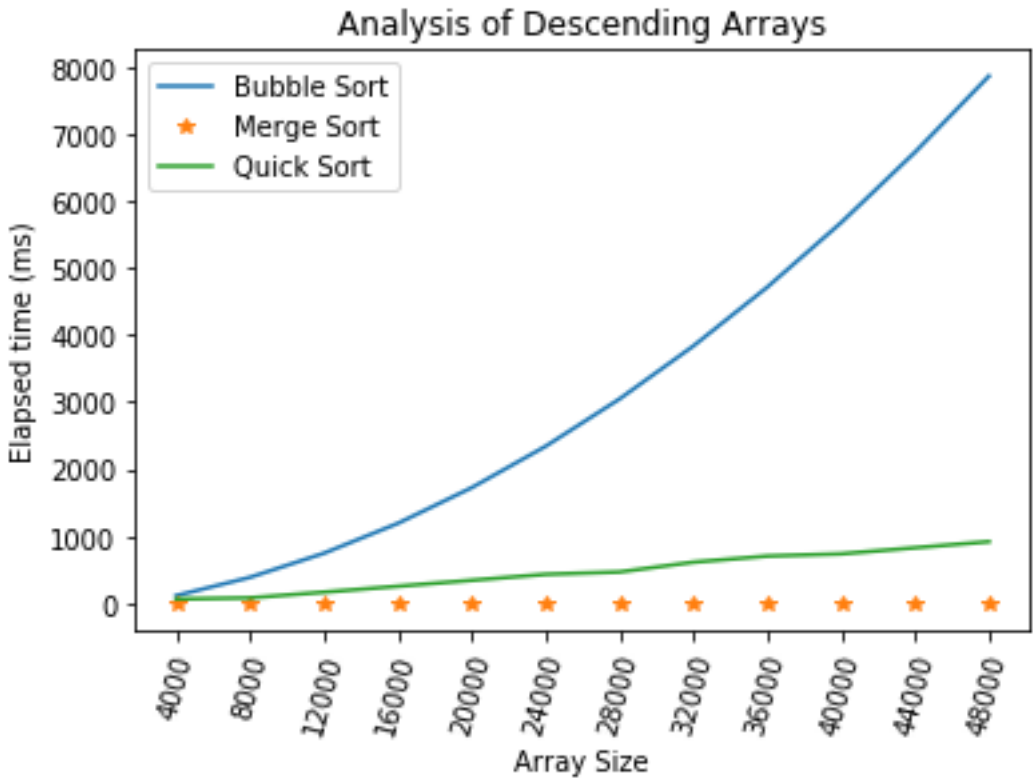
For Descending Random Arrays:

```
DESCENDING RANDOM ARRAYS
-----ANALYSIS OF BUBBLE SORT-----
Array Size          Time Elapsed        CompCount           MoveCount
4000                114   ms            8001999             23992614
8000                387   ms            32003999            71992572
12000               746   ms            72005999            119992689
16000               1191  ms            128007999           167992749
20000               1723  ms            200009999           215992587
24000               2340  ms            288011999           263992641
28000               3045  ms            392013999           311992635
32000               3835  ms            512015999           359992692
36000               4712  ms            648017999           407992530
40000               5674  ms            800019999           455992668
44000               6724  ms            968021999           503992623
48000               7859  ms            1152023999          551992602
-----------------------------------
-----ANALYSIS OF MERGE SORT-----
Array Size          Time Elapsed        CompCount           MoveCount
4000                0     ms            23728               95808
8000                1     ms            52102               207616
12000               2     ms            88268               327232
16000               3     ms            116872              447232
20000               4     ms            155443              574464
24000               5     ms            188994              702464
28000               5     ms            221424              830464
32000               6     ms            250425              958464
36000               7     ms            291553              1092928
40000               8     ms            328330              1228928
44000               9     ms            364925              1364928
48000               10    ms            398438              1500928
-----------------------------------
-----ANALYSIS OF QUICK SORT-----
Array Size          Time Elapsed        CompCount           MoveCount
4000                64    ms            7987536             5298311
8000                81    ms            19744474            9082104
12000               166   ms            37250483            20023544
16000               254   ms            61290330            38183197
20000               343   ms            92050534            63531161
24000               432   ms            129727845           96019041
28000               467   ms            175494429           128562438
32000               613   ms            227300916           175441249
36000               704   ms            286293768           229514023
40000               737   ms            353185521           283543226
44000               828   ms            427300137           344760301
48000               919   ms            508528591           413207656
[emre.karatas@dijkstra ~]$
```

QUESTION #3


Analysis of Random Arrays
Bubble Sort
Merge Sort
Quick Sort
Elapsed time (ms)
0
1000
2000
3000
4000
5000
6000
7000
8000
4000
8000
12000
16000
20000
24000
28000
32000
36000
40000
44000
48000
Array Size


Analysis of Ascending Arrays
Bubble Sort
Merge Sort
Quick Sort
Elapsed time (ms)
0
1000
2000
3000
4000
5000
4000
8000
12000
16000
20000
24000
28000
32000
36000
40000
44000
48000
Array Size



Graphs are made with Spyder IDE by using Python.

Bubble sort algorithm **O(n^2)** for worst (array in reverse order, that is; descending array) and for average (random generated array) case. Bubble sort works **O(n)** for the best case, which the array is already sorted ascending order. In the best case there will be no move, since array is already sorted, therefore it is faster than other cases. There will be n-1 comparisons in this case.

Merge sort algorithm works similar within all cases, which is the fastest one. Its time complexity is **O(nlogn)**. Even though it is the fastest, it requires an extra-array. Divide-conquer algorithm works fast in this sense.

Quick sort algoritm works **O(nlogn)** for best (randomly generated array) and for the average (array is reversed) case (like merge sort). However, quicksort is slow when the array is already sorted (ascending array) and if we select pivot as first element. In this case its time complexity is **O(n^2)**. In this case we move everything from unsorted list to sorted list, which requires a lot of moves, therefore slower.