Title: Algorithm Efficiency and Sorting, İsmet Alp Eren, 21703786, Sec-01, Assigment-01

Question 1

a-)

T(n) = 5T(n/3) + n.logn, where T(1) = 1 and n is an exact power of 3.

$T(n/3) = 5T(n/3^2) + (n/3).\log(n/3)$

$T(n/3^2) = 5T(n/3^3) + (n/3^2).\log(n/3^2)$

$= 5(5(5T(n/3^2) + (n/3^2). \log(n/3^2)) + (n/3).\log(n/3)) + n.\log n$

$=5^k T(n/3^k) + \sum_{i=0}^{k-1} 5^i . \left(\frac{n}{3^i}\right) . \log\left(\frac{n}{3^i}\right)$

because n is power of 3 we can say $3^k = n$ than

$=5^{\log n} . O(1) + (5^0 + 5^1 + 5^k). \sum_{i=0}^{k-1} \left(\frac{n}{3^i}\right). \sum_{i=0}^{k-1} \log\left(\frac{n}{3^i}\right)$

$=5^{\log n} + 5^{k-1}.n(1-3^k / 1-3). \sum_{i=0}^{k-1} \log\left(\frac{n}{3^i}\right)$

$=5^{\log n} + 5^{k-1}.n(n/2). \sum_{i=0}^{k-1} \log\left(\frac{n}{3^i}\right)$

$=5^{\log n} + 5^{k-1}.n^2. \sum_{i=0}^{k-1} \log\left(\frac{n}{3^i}\right)$ I couldn't find $\log(n/3^i)$ part

=O(n.logn)

$T(n) = T(n - 1) + n^2$, where T(1) = 1

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

$T(n-2) = T(n-3) + (n-2)^2 + (n^2-2n + 1) + n^2$

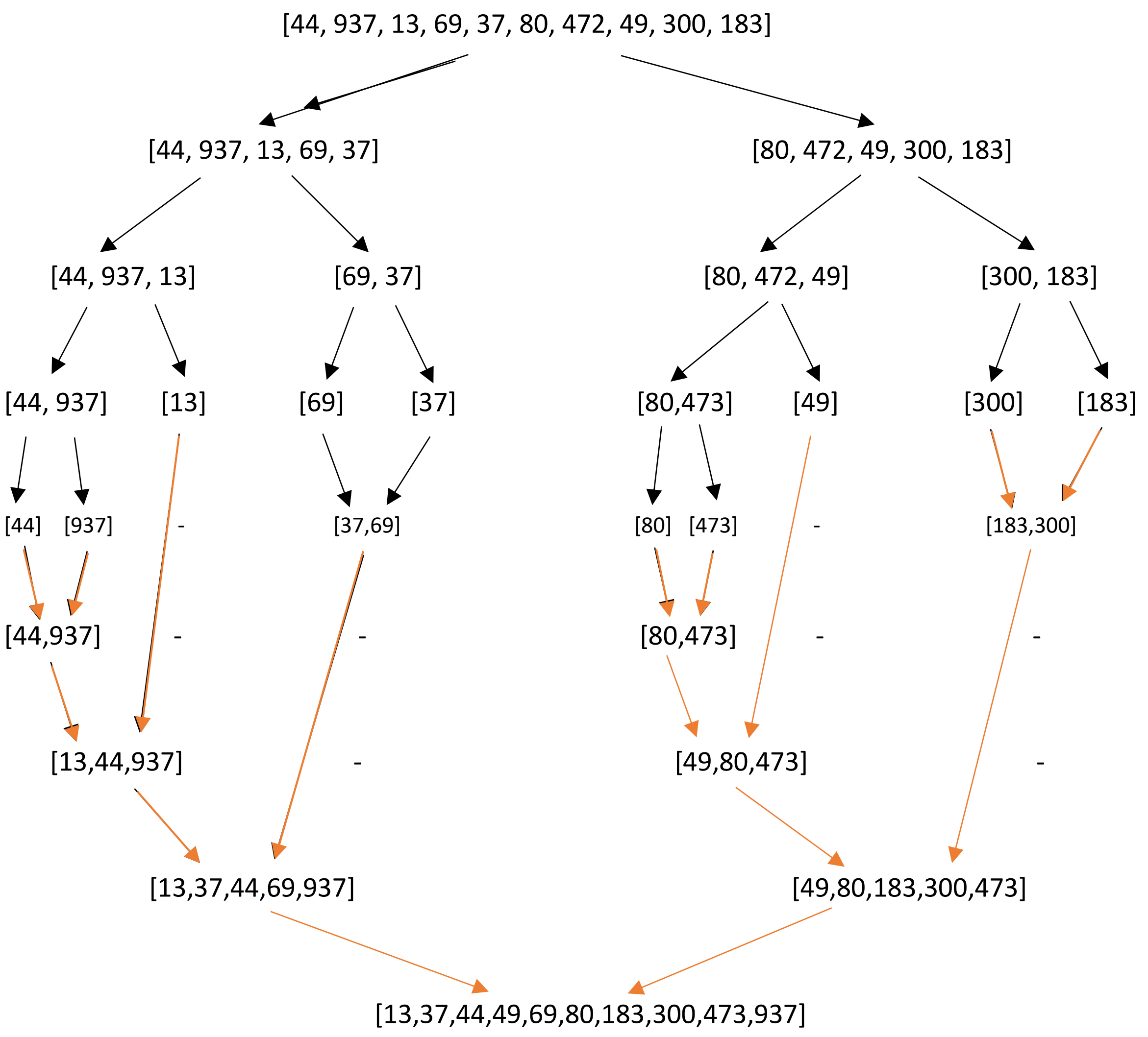
To get T(1) we should repeat this step (n-1) times → T(n-(n-1)) = T(1)

$= T(1) + \sum_{i=0}^{n-1} n^\wedge 2 - 2n.\sum_{i=0}^{n-1} i + \sum_{i=0}^{n-1} i^\wedge 2$

$= 1 + (n-1)n^2 - 2n(n^2/2) + n^3/3$

$= \Theta(n^3)$

b-)



Orange rows are merging

Black rows are dividing

[44, 937, 13, 69, 37, 80, 472, 49, 300, 183]

- If 44 > key (937) (false) [44, 937, 13, 69, 37, 80, 472, 49, 300, 183]

- If 937 > key (13) (true), if 44 > 13 (true) [13,44, 937, 69, 37, 80, 472, 49, 300, 183]

- If 937 > key (69) (true), If 44 > key (69) (false) [13,44,69,937, 37, 80, 472, 49, 300, 183]

- If 937 > key (37) (true), If 69 > key (37) (true), If 44 > key (37) (true), If 13 > key (37) (false)

[13,37,44,69,937, 80, 472, 49, 300, 183]

- If 937 > key (80) (true), If 69 > key (80) (false) [13,37,44,69,80,937, 472, 49, 300, 183]

- If 937 > key (472) (true), If 80 > key (472) (false) [13,37,44,69,80,472,937, 49, 300, 183]

- If 937 > key (49) (true), If 472 > key (49) (true), if 80 > 49 (true) , if 69 > 49 (true) , if 44 > 49 (false)

[13,37,44,49,69,80,472,937, 300, 183]

- If 937 > key (300) (true), If 472 > key (300) (true), if 80 > 300 (false)

[13,37,44,49,69,80,300,472,937,183]

- If 937 > key (183) (true), If 472 > key (183) (true), if 300 > 183 (true), if 80> 183 (false)

[13,37,44,49,69,80,183,300,472,937]

Sorted array is [13,37,44,49,69,80,183,300,473,937]

c-)

T(n) = T(n - 1) + cn

T(n -1) = T(n - 2) + c(n - 1)

T(n - 2) = T(n - 3) + c(n - 2)

= T(n -2) + c(n-2) + c(n - 1) + cn

= T(n-3) + c(n-3) + c(n-2) + c(n - 1) + cn

= T(1) + c(1) + … + c(n-3) + c(n-2) + c(n - 1) + cn

T(1) is constant, $\sum_{i=2}^{n} i$ is equal to n^2

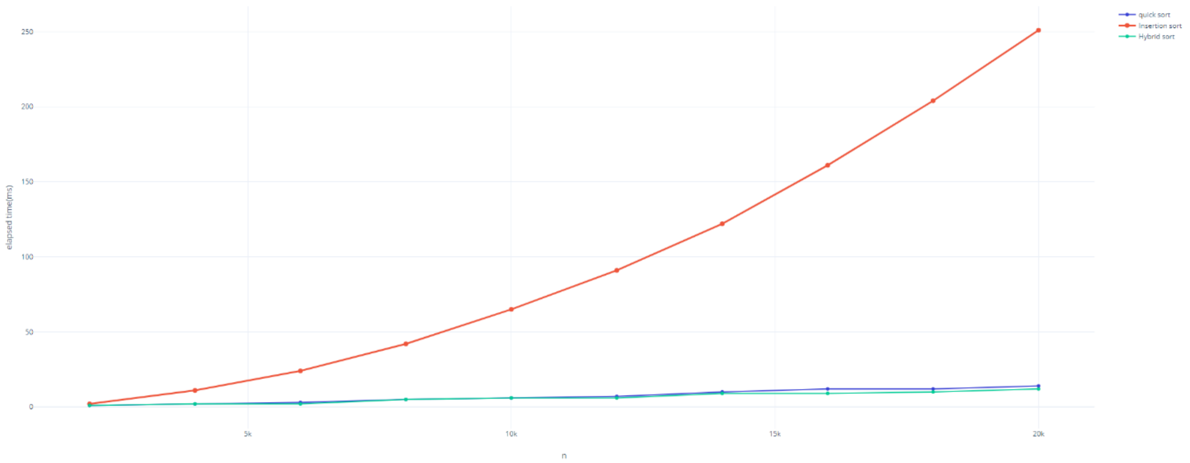
T(n) = T(1) + c $\sum_{i=2}^{n} i$ = $O(n^2)$

Question-2

```
Microsoft Visual Studio Debug Console
1 17 20 43 57 58 92 93 99 100
1 17 20 43 57 58 92 93 99 100
1 17 20 43 57 58 92 93 99 100
-----------------------------------------------------------------------------------------------
Part a - Time analysis of Quick Sort
Array Size     Time Elapsed(ms)      compCount         moveCount
2000            0                       12239             42189
4000            1                       37017             127275
6000            1                       76509             262147
8000            2                       148155            498721
10000           2                       224996            756228
12000           4                       320308            1074776
14000           5                       451775            1507177
16000           5                       597877            1988735
18000           5                       763661            2534895
20000           6                       951462            3152722
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
Part b - Time analysis of Insertion Sort
Array Size     Time Elapsed(ms)         compCount         moveCount
2000            3                       976988              980986
4000            10                      4936134             4948130
6000            24                      13913870            13937864
8000            43                      30127941            30167933
10000           65                      55488636            55548626
12000           92                      91825493            91909481
14000           122                     140605825           140717811
16000           159                     204372420           204516404
18000           210                     284798209           284978191
20000           251                     384351481           384571461
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
Part c - Time analysis of Hybrit Sort
Array Size    Time Elapsed(ms)       compCount        moveCount
2000            1                       11059             37207
4000            0                       33553             112825
6000            1                       69382             232478
8000            2                       136251            449313
10000           2                       207103            682169
12000           2                       295169            970777
14000           3                       418086            1367982
16000           3                       554401            1809433
18000           4                       709320            2310742
20000           4                       884940            2878180
-----------------------------------------------------------------------------------------------

C:\Users\ismet\source\repos\Project2\Debug\Project2.exe (process 6516) exited with code 0.
Press any key to close this window . . .
```

Question-3



1-) Insertion sort average and worst-case time complexity is $O(n^2)$ and I get the same result with my program, insertion sort has $O(n^2)$ time complexity. For Quick sort average case O(n.logn) and worst case $O(n^2)$. Quick sort in my program worked as average case as it should be and with the graph wee can see quick sort also has n.logn time complexity. (it looks like n or logn because of the values but it's n.logn) Hybrid sort has same time complexity but also has slither less move and comparison number that is why there is little time difference. There is no error in my values.

2-) There is no big difference between hybrid and quick sort because of sizes of arrays.

Since array sizes are not large, we can't see huge difference but still there is a few milliseconds between these two sorting algorithms. The reason why that different occur is, hybrid sort is switch to insertion sort when subarray size less than 10 and that gives advantage to hybrid sort but this advantage is valid for small sized array. (20.000 also small size, millions I assume for large sized)

The disadvantage of hybrid sort is, hybrid sort's move count is larger than quick sort's and that cause memory problems for large sized array.