BLG 335E – ANALYSIS OF ALGORITHMS I
ASSIGNMENT 1
Student Name: MUHAMMET ASLAN
No: 150160031
Deadline : 10/12/2020

PART 2:

Part A Quicksort Analysis:

Worst Case: The worst case is when the input is completely sequential or reverse sequential or all same elements. Because we want to divide the input by 2 each time, but when it is completely ordered, all elements will remain either to the right or left of the pivot. In other words, for N inputs, N depths will be reached, that is, N times pivot will be selected. So ;

T(N) = T(N-1) + N
= T(N-2) + (N-1) + N
= T(N-3) + (N-2) + (N-1) + N
= T(N-4) + (N-3) + (N-2) + (N-1) + N
= . . .
T(N) = T(1) + 2 + 3 + . . . +(N-1) + N

$$\mathbf{T(N) = \frac{N*(N+1)}{2} = \frac{N^2 + N}{2} = O(N^2)}$$

Best Case : Quicksort's best case occurs when the partitions are as evenly balanced as possible: their sizes either are equal or are within 1 of each other. The former case occurs if the subarray has an odd number of elements and the pivot is right in the middle after partitioning, and each partition has (n-1)/2 elements. The latter case occurs if the subarray has an even number nn of elements and one partition has n/2 elements with the other having n/2-1.

T(n) = 2*T(n/2) + n
= 2*[2*T(n/4) + n/2] + n
= 2^2*T(n/4) + n + n
= 2^2*T(n/4) + 2n
= 2^2*[2*T(n/8) + (n/4)] + 2n
= 2^3*T(n/8) + 2^2*(n/4) + 2n

$= 2^3*T(n/8) + n + 2n$
$= 2^3*T(n/8) + 3n$
$= 2^4*T(n/16) + 4n$
$= 2^k*T(n/(2^k)) + k*n$
$= 2^k*T(1) + k*n$
$= 2^k*1 + k*n$
$= 2^k + k*n$
$= n + k*n$
$= n + (\log(n))*n$
$= n*(\log(n) + 1)$

T(n)~= nlog(n) = O(nlog(n))

Average Case : When we pick the pivot, the worst we can do is 0 | n and the best we can do is n/2 | n/2. The average case will find we getting a split of something more like n/4 | 3n/4, assuming uniform randomness.
We get O(nlogn) once constants are eliminated.

T(n)=O(n)+T(0)+T(n-1)=O(n)+T(n-1)
T(N) = O(N) + 2T(N/2)

The master theorem for divide-and-conquer recurrences tells us that $\boldsymbol{T(n) = O(n \log n)}$.

PART B:

1- Yes, it will give us the output we want. Because after sorting according to profit conditions, we write to this file named sorted_by_profits.txt. Later, when we sort this file alphabetically, it will sort the countries with the same name in their order in the sorted_by_profits.txt file. This will give us both alphabetical and profit order output.

Example Input sales.txt:

South Africa	Fruits	443368995	1593	3839.13
Morocco	Clothes	667593514	4611	338631.84
Papua New Guinea	Meat	940995585	360	20592
Djibouti	Clothes	880811536	562	41273.28
Slovakia	Beverages	174590194	3973	62217.18
Sri Lanka	Fruits	830192887	1379	3323.39
Seychelles	Beverages	425793445	597	9349.02
Tanzania	Beverages	659878194	1476	23114.16
Ghana	Office Supplies	601245963	896	113120
Tanzania	Cosmetics	739008080	7768	1350622.16

Sorted by profit the sales.txt in sorted_by_profits.txt :

Tanzania	Cosmetics	739008080	7768	1350622.16
Morocco	Clothes	667593514	4611	338632
Ghana	Office Supplies	601245963	896	113120
Slovakia	Beverages	174590194	3973	62217.2
Djibouti	Clothes	880811536	562	41273.3
Tanzania	Beverages	659878194	1476	23114.2
Papua New Guinea	Meat	940995585	360	20592
Seychelles	Beverages	425793445	597	9349.02
South Africa	Fruits	443368995	1593	3839.13
Sri Lanka	Fruits	830192887	1379	3323.39

Sorted by according to country names the sorted_by_profits.txt;

Djibouti	Clothes	880811536	562	41273.3
Ghana	Office Supplies	601245963	896	113120
Morocco	Clothes	667593514	4611	338632
Papua New Guinea	Meat	940995585	360	20592
Seychelles	Beverages	425793445	597	9349.02
Slovakia	Beverages	174590194	3973	62217.2
South Africa	Fruits	443368995	1593	3839.13
Sri Lanka	Fruits	830192887	1379	3323.39
Tanzania	Cosmetics	739008080	7768	1350622.16
Tanzania	Beverages	659878194	1476	23114.2

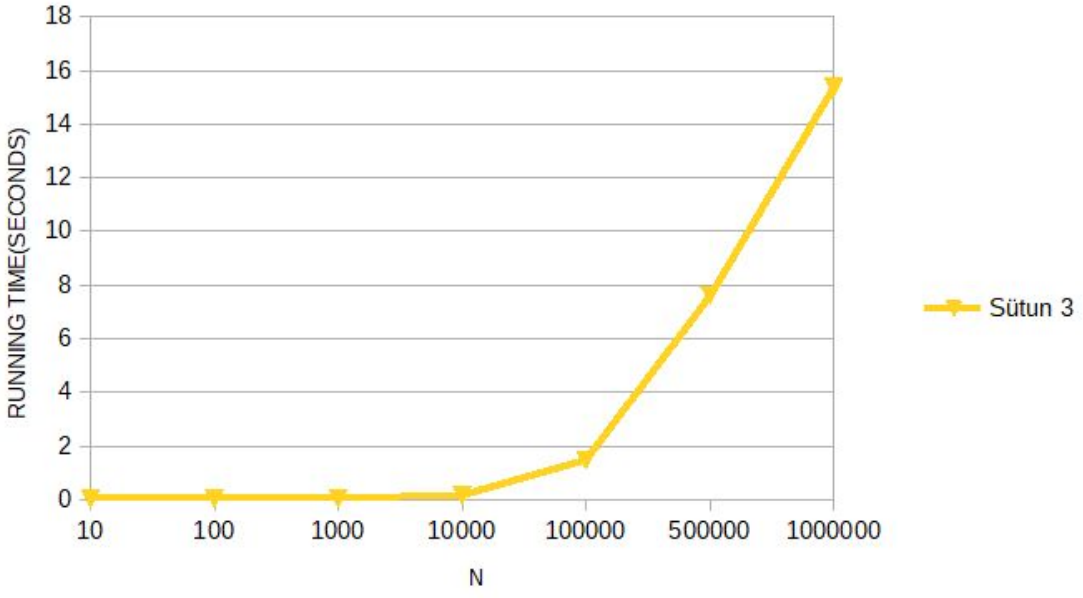
2-
-MERGE SORT
-BUBBLE SORT
-INSERTION SORT

PART C:

Run Time	1	2	3	4	5	6	7	8	9	10	Average Time
N											
10	0,04	0,042	0,04	0,04	0,04	0,042	0,041	0,041	0,041	0,04	0,0407
100	0,043	0,042	0,042	0,048	0,043	0,043	0,043	0,042	0,04	0,042	0,0428
1000	0,055	0,053	0,053	0,054	0,055	0,055	0,054	0,054	0,054	0,055	0,0542
10000	0,178	0,168	0,177	0,167	0,169	0,166	0,168	0,169	0,17	0,169	0,1701
100000	1,522	1,42	1,48	1,573	1,484	1,486	1,483	1,46	1,472	1,46	1,484
500000	7,842	7,479	7,546	7,552	7,589	7,624	7,49	7,593	7,681	7,779	7,6175
1000000	15,561	15,78	15,983	15,349	15,194	15,203	15,852	15,011	14,791	15,04	15,3764

Since the time complexity of the average case is NlogN, N and running time increased linearly. For example, the average running time for N = 500000 is 7,6175 while it is 15,3764 for N = 1000000. So when N doubled, running time also almost doubled. Since there was not much difference in low values, the difference was not very clear at those points. However, as N increased, the linear increase became more pronounced. Linear increases clear between N=100000 and N=1000000.

Graph of my running time:



PART D:

1-Worst case time complexity did not increase linearly. Because N^2 increases exponentially. The average time for N numbers is larger than the average times I get in part c. Because the time complexity for the worst case, N^2 and N^2 increase faster than NlogN. I could not get results for N values greater than 10000. However, as N = 1000 passed from N = 10000, running time increased excessively compared to the increase in part c, N increased 10 times, but running time increased approximately 68 times, which showed that it increased exponentially.

Run Time	1	2	3	4	5	6	7	8	9	10	Average Time
N											
10	0,042	0,041	0,042	0,04	0,042	0,041	0,041	0,04	0,041	0,041	0,0411
100	0,045	0,044	0,043	0,043	0,043	0,043	0,043	0,045	0,045	0,055	0,0449
1000	0,168	0,164	0,177	0,167	0,167	0,16	0,175	0,162	0,167	0,165	0,1672
10000	11,123	11,551	11,167	12,024	11,368	11,078	11,18	11,089	11,053	12,027	11,366

2-Reverse of sorted.txt or when all elements are the same in the input file.

3-We can update the quick sort algorithm. The leading or trailing pointers move continuously if there is no moving element, and when we scan the input once, we understand that the input is sequential, and we can understand that the input is sequential without selecting the pivot or partition the input again.