BLG-335E
Homework 1 - Report
Quicksort Algorithm

Muhammed Salih YILDIZ

150180012

Fall-2020

Part a.

Asymptotic equation for the worst case : f(n) = $c(\frac{n\times(n+1)}{2} - 1)$

Prove: If we choose the smallest or the largest number in the array in each iteration, then the array will split into two subarrays which one of them has 0 element and the other has (n-1) elements. So, in each iteration the algorithm has (n-k) subproblems where k is number of the step. If we assume each partition takes c time, then in each step takes c(n-k) time. Like:

$$cn + c(n-1) + c(n-2) + c(n-3) + \cdots + 2c = c(\frac{n \times (n+1)}{2} - 1)$$

T(0) = T(1) = 0

T(N) = N + T(N-1)

T(N-1) = (N-1) + T(N-2)

T(N-2) = (N-2) + T(N-3)

…

T(2) = 2 + T(1)

T(1) = 0

So, T(N) = N + (N-1) + (N-2) ... + 3 + 2 ≈ (N^2)/2

O(N^2)

Asymptotic equation for the best case : f(n) = $n \times \log(n)$

Prove: If the partitions are as evenly balanced as possible, then best case occurs. In every iteration the array divides into 2 mostly equal arrays. So, in each step we have to make n operation. Also, number of the iteration becomes log(n). So, T(N) = 2T(N/2) + N

The master theorem says:

$$\text{Suppose } T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d). \text{ Then}$$

$$T(n) = \begin{cases} O(n^d \log(n)) & \text{if } a = b^d \\ O(n^d) & \text{if } a < b^d \\ O(n^{\log_b(a)}) & \text{if } a > b^d \end{cases}$$

In quicksort recurrence equation a = 2, b = 2, d = 1. Thus, we can say that:

$T(n) = n \times \log(n)$. To sum up, in the quicksort it takes O(n x log(n)) times to sort the array.

Asymptotic equation for the average case : f(n) = $n \times \log(n)$

Prove: In the average case one branch of the division tree should be longer than the other one but there should be not too much difference between them. Let us assume that one branch takes n/5 size array and the other one 4n/5 in each iteration. Then, the short branch will take a step to the last node. Because we can say that it would reach to n through

multiplying with 5 in each step. On the other hand, the branch will take steps to the last node. Thus, the algorithm has steps at most. Then, the program will work in times. When we divide it into, we got which is constant. We can neglect this constant number and we can say that quicksort takes times in case of average.

Part b.

1) Yes, the solution gives the desired output. Because the quicksort algorithm does not change order of the element that have the same values. There are 2 cases for this situation. Firstly, the elements that have the same value are lower than the pivot. In partition, the algorithm checks them in order of how they are. So, the order does not change. The second case occurs if the pivot is one of the ordered elements. Also, in this case the algorithm checks elements from left to right and does not change the order. For example:

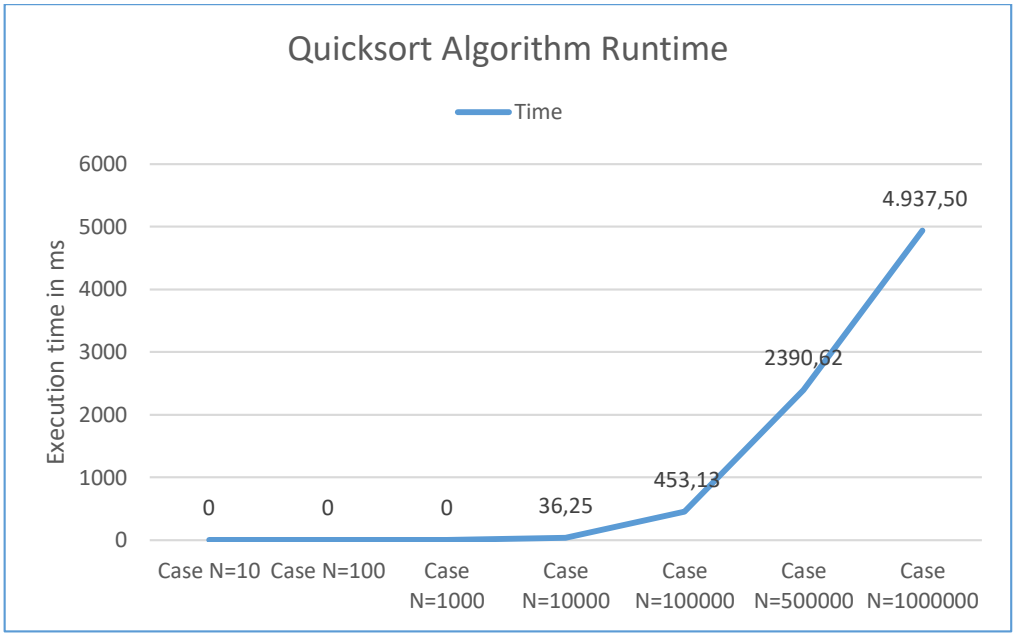
	Country	**Item Type**	**Order ID**	**Units Sold**	**Total Profit**
a.	Tanzania	Cosmetics	739008080	7768	1350622.16
b.	Ghana	Office Supplies	601245963	896	113120
c.	Tanzania	Beverages	659878194	1476	23114.16
d.	Taiwan	Fruits	732588374	8034	19361.94
e.	Seychelles	Beverages	425793445	597	9349.02

This dataset is sorted by profits. When we apply the quicksort, in the first step **e** object becomes pivot and **b** object places in the left array and rest of the list becomes right array. However, in the right array **a** element places on the left side of the **c** element. So, the order between **a** and **c** objects that have the same country name never changes. Therefore, the order becomes like:

	Country	**Item Type**	**Order ID**	**Units Sold**	**Total Profit**
a.	Ghana	Office Supplies	601245963	896	113120
b.	Taiwan	Fruits	732588374	8034	19361.94
c.	Tanzania	Cosmetics	739008080	7768	1350622.16
d.	Tanzania	Beverages	659878194	1476	23114.16
e.	Seychelles	Beverages	425793445	597	9349.02

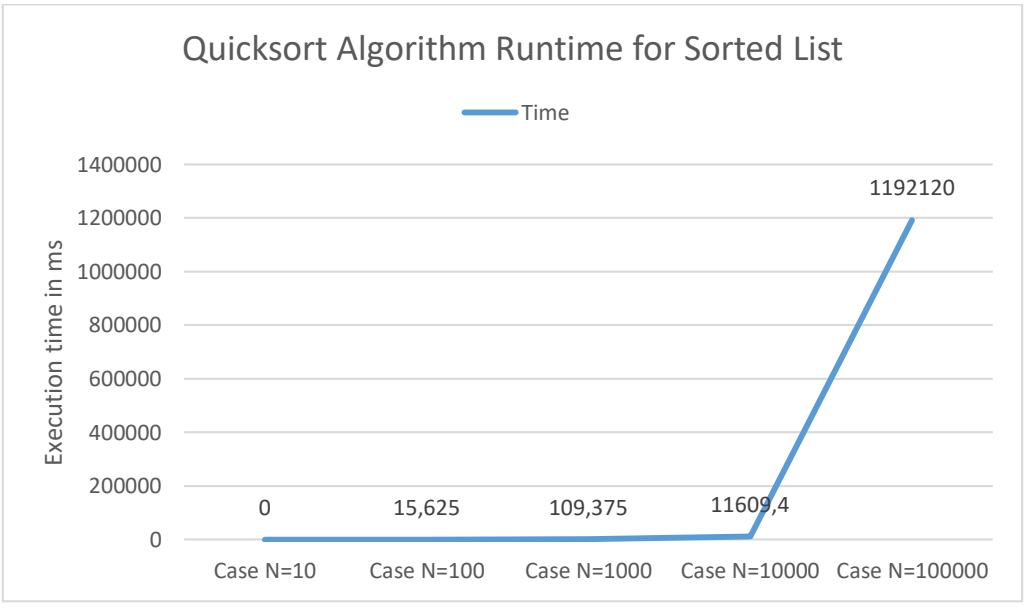
2) Merge Sort, Counting Sort, Insertion Sort

Part c.



In this table we can observe that execution time dependent on the number of input. However, there is no linear relation. It is more likely exponential growth. As we consider the asymptotic bound which is $n \times \log(n)$ in the best case and $\log(n^2)$ in the worst case, we can understand the relation. The relation is neither the best or worst case, it is between them.

Part d.



1) Sorting a sorted list is the worst case in the quicksort which picks the pivot as last or first element of the array. We can observe this situation at the table. Because runtime increased with exponential as the second power of increment rate. Also, the results are much higher than the part c results. This is the result of asymptotic

equations that found in **part a.** as $n \times \log(n)$ for the best case and $\log(n^2)$ for the worst case.

2) The reversed order of the sorted list will give the same result.
3) Selecting the pivot as the middle element of the array.