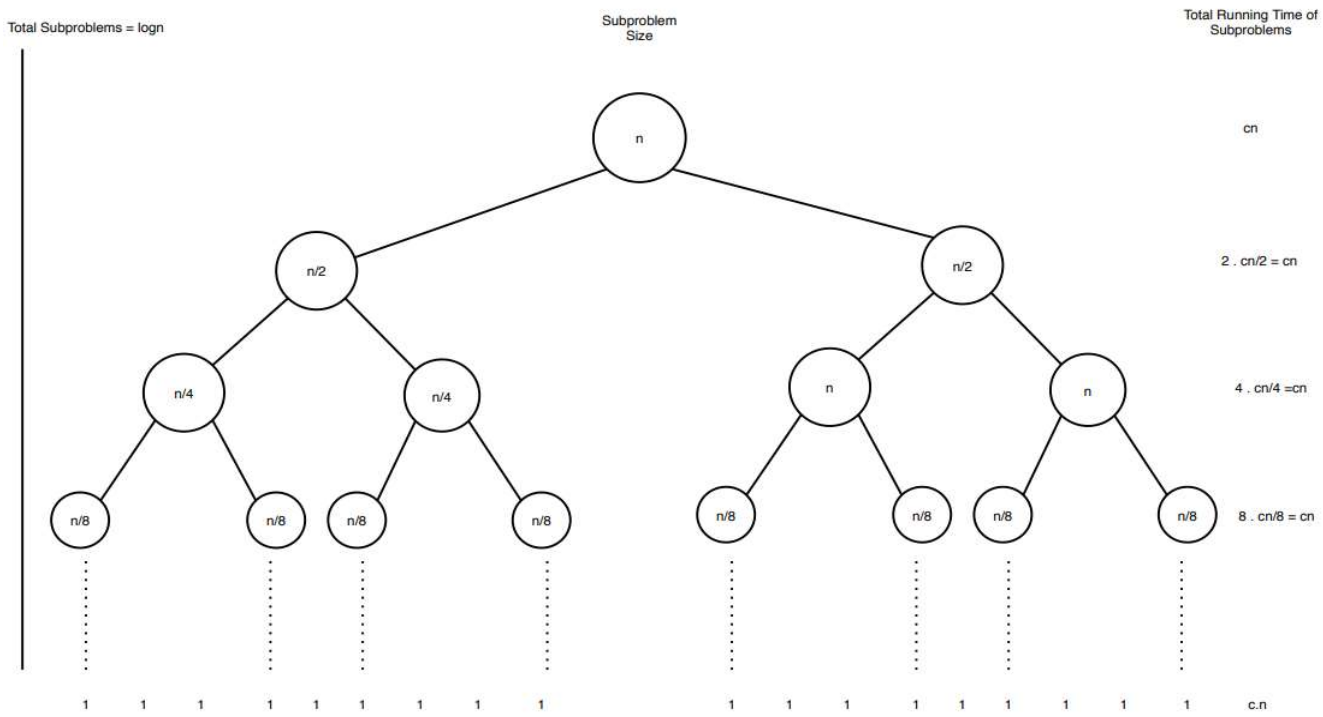
BLG 335E – Analysis of Algorithms I
Fall2020 Homework 1 Report
Oguzhan Karabacak 150170021

a)

Asymptotic Upper Bound for Best Case

The best case in QuickSort occurs when the number of elements in partitions is equal, so the Pivot is in the middle of the order.



$$T(n) = \begin{cases} c & , \quad n = 1 \\ 2T\left(\frac{n}{2}\right) + cn, & n > 1 \end{cases}$$

$$T(n) = 2\left[2T\left(\frac{n}{4}\right) + \left(\frac{cn}{2}\right)\right] + cn$$

$$T(n) = 2\left[2\left[2T\left(\frac{n}{8}\right) + \frac{cn}{4}\right] + \frac{cn}{2}\right] + cn$$

...

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + c\sum_{i=0}^{k-1} \frac{n}{2^i}$$

$$\sum_{i=0}^{k-1} \frac{n}{2^i} = k.n \quad (c \; is \; ignored)$$

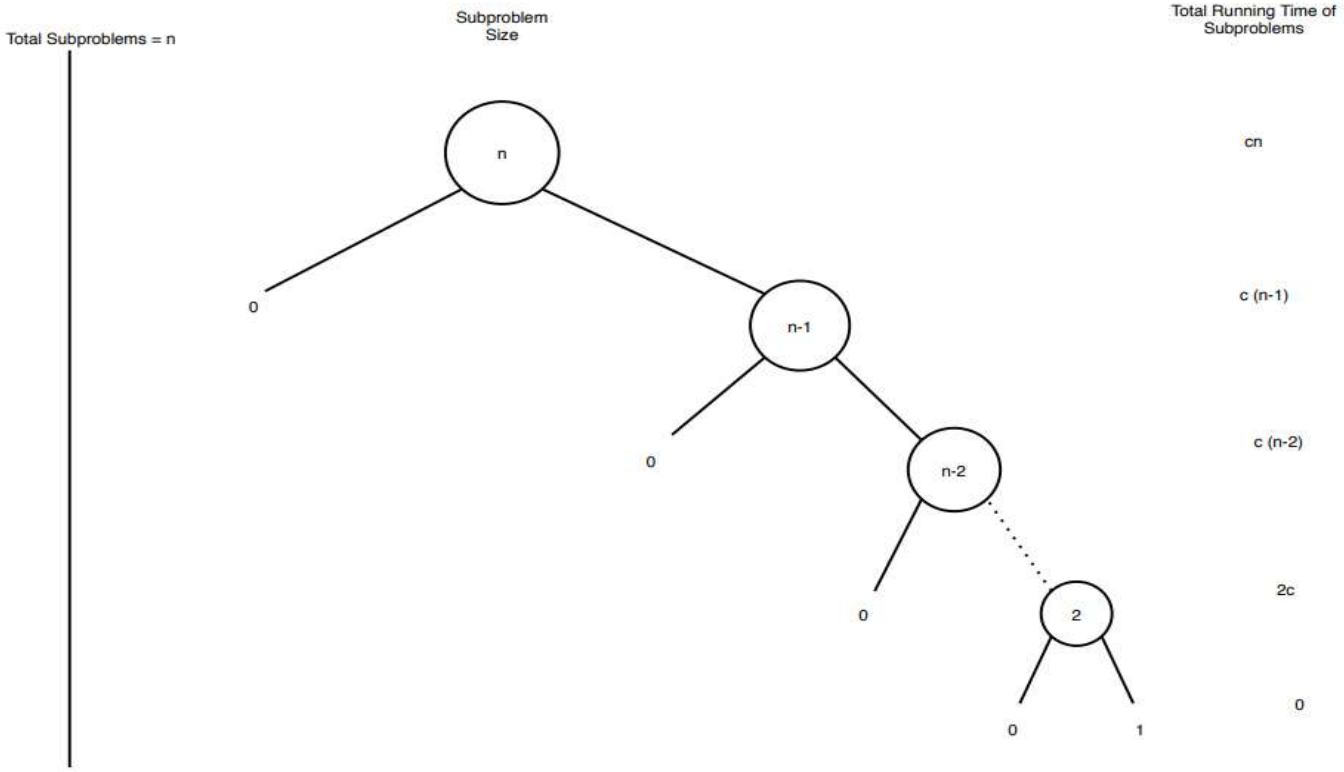
$$T(1) = 0\,, 2^k = n\,, k = \log_2 n$$

$$T(n) = nT(1) + nlog_2 n$$

$$so\ \Theta(n) = nlog_2 n$$

Asymptotic Upper Bound for Worst Case

Quick Sort's worst case is when the chosen pivot is either the largest or smallest element of the list. When this happens, one of the two sublists will be empty, so Quick Sort is only called on one list during the Sort step.



$$T(n) = \begin{cases} c, & n = 1 \\ T(n-1) + cn, & n > 1 \end{cases}$$

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

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

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

$$\cdots$$

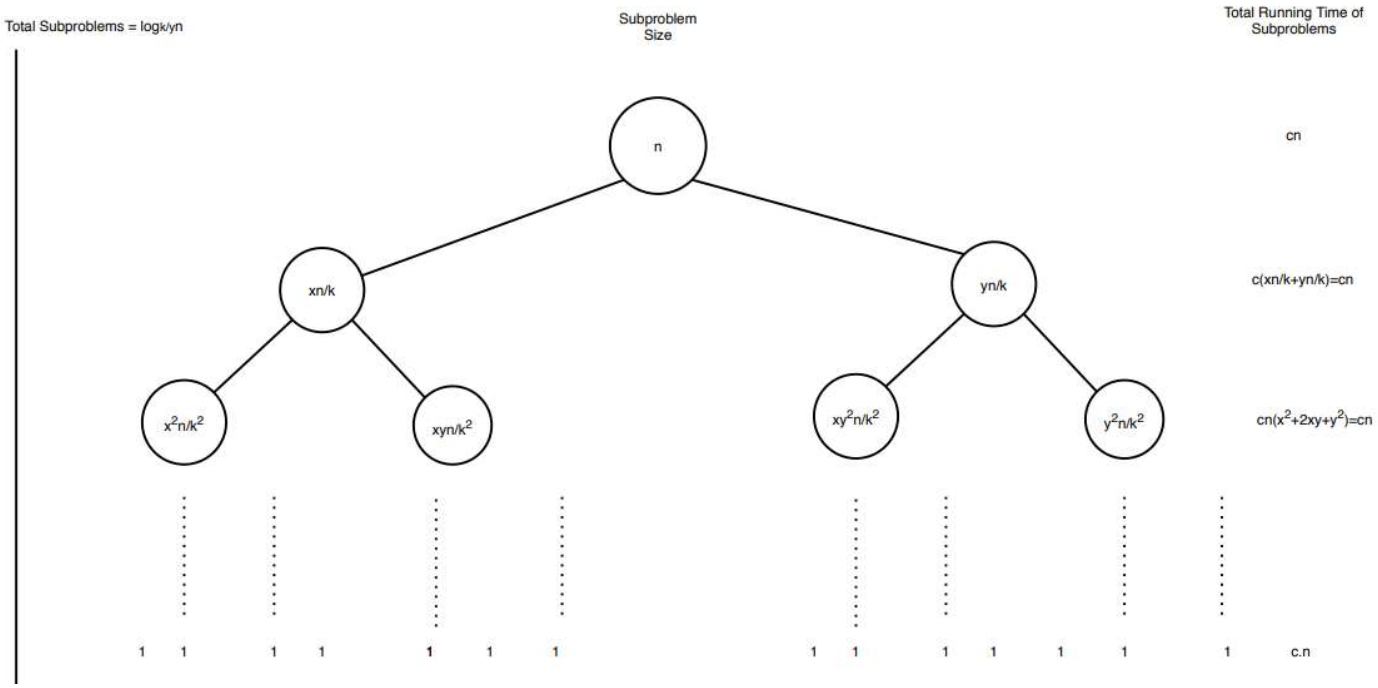
$$T(1) = 0$$

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

And c is ignored so big O notation is $\Theta(n^2)$

Asymptotic Upper Bound for Average Case

We do not know how the elements will partition in the average case, but we know that the best-case is n/2-n/2 and the worst case is 0-n. If we choose a value in these two intervals, this gives us information about the average case.
For example, the elements be divided into xn/k and yn/k. (x+y=k) and let assume that (x/y>1)
In this case , T(n) becomes:



$$T(n) = T\left(\frac{xn}{k}\right) + T\left(\frac{yn}{k}\right) + n$$

$$T(n) = T\left(\frac{x^2n}{k}\right) + 2T(\frac{2xyn}{k}) + T\left(\frac{y^2n}{k}\right) + 2n$$

…

Until the partitions 1, $n=(x/y)^s$ → (x/y>1 and s step size)

$$\log_{\frac{x}{y}} n = s$$

$$T(n) = nlog_{\frac{x}{y}}n \quad so \quad \Theta(nlogn)$$

B)
1.

1) Sort the sales.txt data by the total profits and write it into sorted_by_profits.txt
2) Sort the sorted_by_profits.txt data according to country names using QuickSort
Does this solution give us the desired output for all cases?

No, it does not give the desired output. Countries are sorted by name, but for the same country, total_profit are not sorted.

Because the countries with the same name cannot be compared with each other in terms of names because their names are the same in the QuickSort algorithm. That is, they do not enter the if block (the code below) in the Partition function and i always remains low-1.(i=low-1)

```
int Partition(Sale sales[], int low, int high) {
        Sale pivot = sales[high];
        int i = (low - 1);

        for (int j = low; j <= high-1; j++) {
                if (sales[j].getCountry() < pivot.getCountry()) {
                        i++;
                        swap(&sales[i], &sales[j]);
                }
        }
        swap(&sales[i+1], &sales[high]);
        return (i + 1);
}
```

Here, when total_profit is sorted in descending order, in countries with the same name, total_profit comes last in order, and since Pivot is always the last element, Pivot is the country with the smallest total_profit.

After the code leaves the Loop block, in other words, after variable j reaches to Pivot, **swap (& sales [i + 1], & sales [high]);** code runs and replaces the pivot with the first element, so from countries with the same name, the smallest total_profit comes first.

The same situation repeats in a new QuickSort block after determining the rank of the country with the smallest total_profit, but this time in countries with the same name, total_profit is the largest and comes after the smallest.

For example, let's assume that we list 6 countries in descending order according to total_profit. The names and total_profit of these countries (Other information is not used here because it is not important).

Nepal 1736439.69

Afghanistan 1646030.36

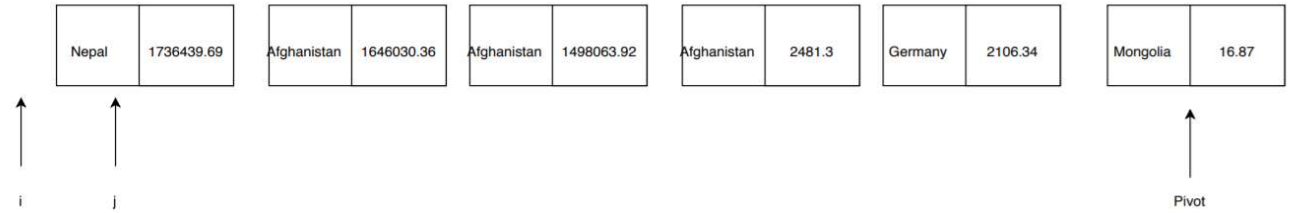
Afghanistan 1498063.92

Afghanistan 2481.3

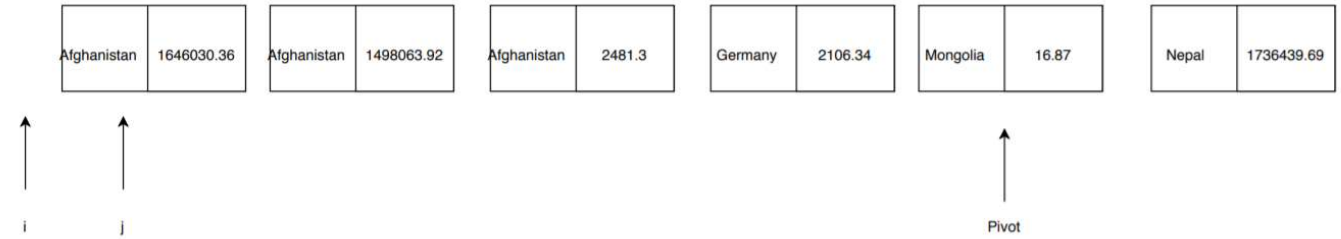
Germany 2106.34

Mongolia 16.87

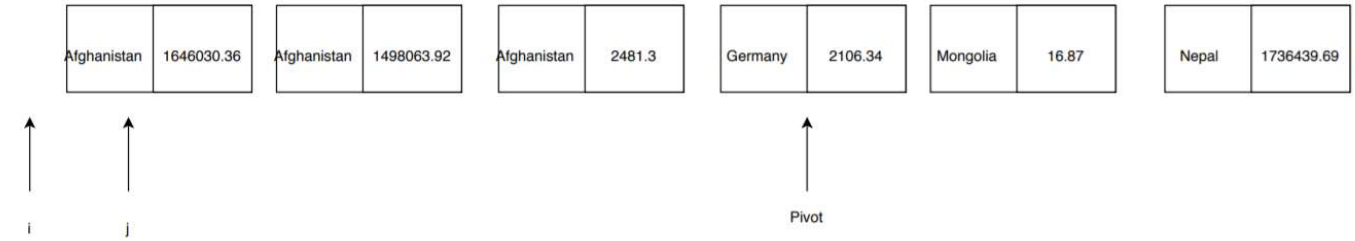
And apply QuickSort on this data for a better understanding of the situation.



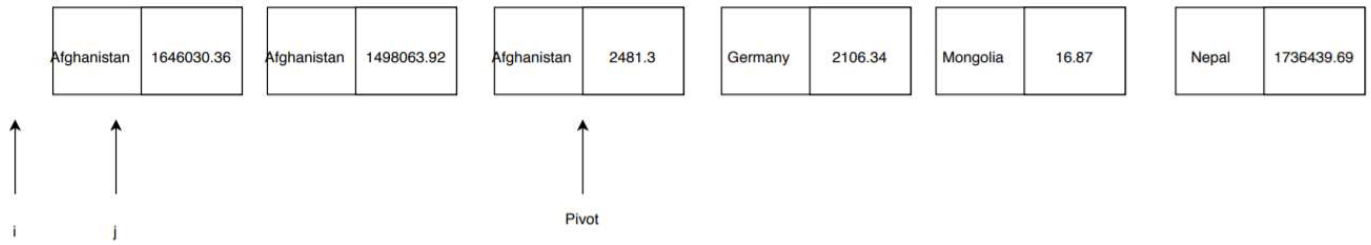
After first QuickSort Nepal's order will be determined and apply QuickSort other countries.



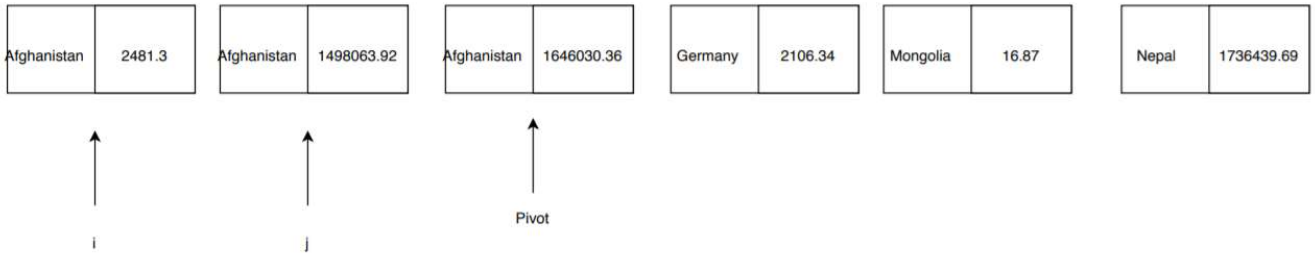
After second QuickSort Mongolia's order will be determined and apply QuickSort other countries.



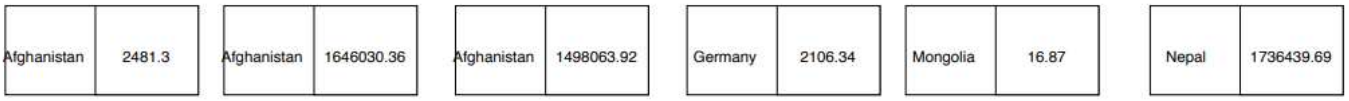
After second QuickSort Germany's order will be determined and apply QuickSort Afghanistan country. What follows is very important.



In this QuickSort, The if block will never be executed but in the last of Partition Function, **swap (& sales [i + 1], & sales [high]);** is executed and Pivot and J replace.



Now, Order of Afghanistan which total_profit is 2481.3 is determined and the other Afghanistan data will be apply Quicksort and Pivot and J will replace the above mentioned event will happen again.



and the final order will be like this.

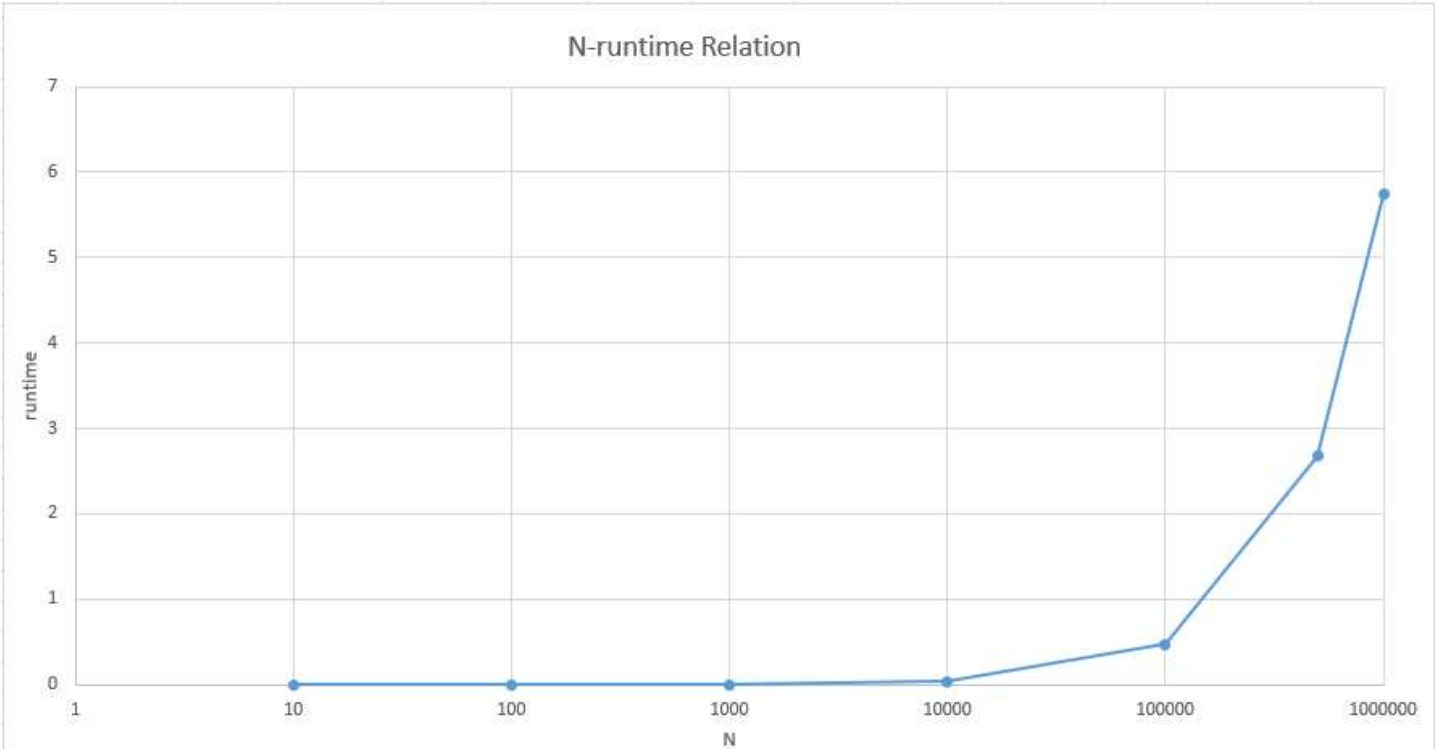
2.

Insertion Sort, Merge Sort,Bubble Sort algorithms give the desired output.

C)

N	10	100	1000	10K	100K	500K	1M
1	0.000034	0.0003	0.002	0.034	0.477	2.708	5.758
2	0.000026	0.0003	0.002	0.034	0.466	2.682	5.746
3	0.000029	0.00035	0.003	0.037	0.491	2.681	5.737
4	0.000023	0.00034	0.002	0.035	0.453	2.671	5.727
5	0.000018	0.00027	0.002	0.034	0.464	2.677	5.718
6	0.000019	0.00022	0.003	0.035	0.474	2.661	5.723
7	0.000020	0.00024	0.002	0.035	0.459	2.679	5.729
8	0.000022	0.0003	0.003	0.037	0.468	2.654	5.751
9	0.000021	0.0003	0.003	0.035	0.469	2.658	5.802
10	0.000018	0.00029	0.002	0.036	0.482	2.684	5.711

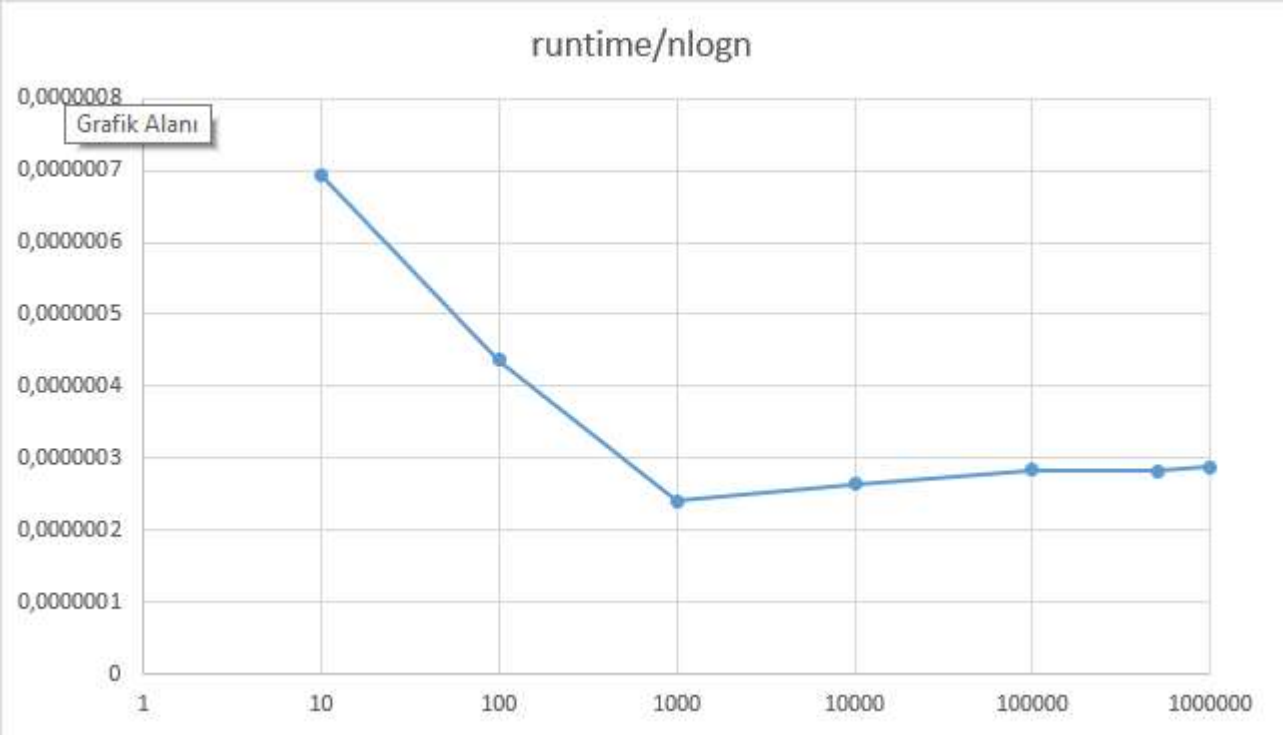
N	(average) runtime
10	0.000023
100	0.00029
1000	0.0024
10000	0.0352
100000	0.4703
500000	2.67
1000000	5.74



The input orders is a random orders. So, expected case is average case and expected time complexity is nlogn. And in experimental The "N-runtime relation" graph looks like

a nlogn graph, so The experimental results in this part confirm the average asymptotic bound "nlogn" in part A. When we do runtime / nlogn, we get similar results for N cases. (It turns out slightly higher for N = 10,100.)

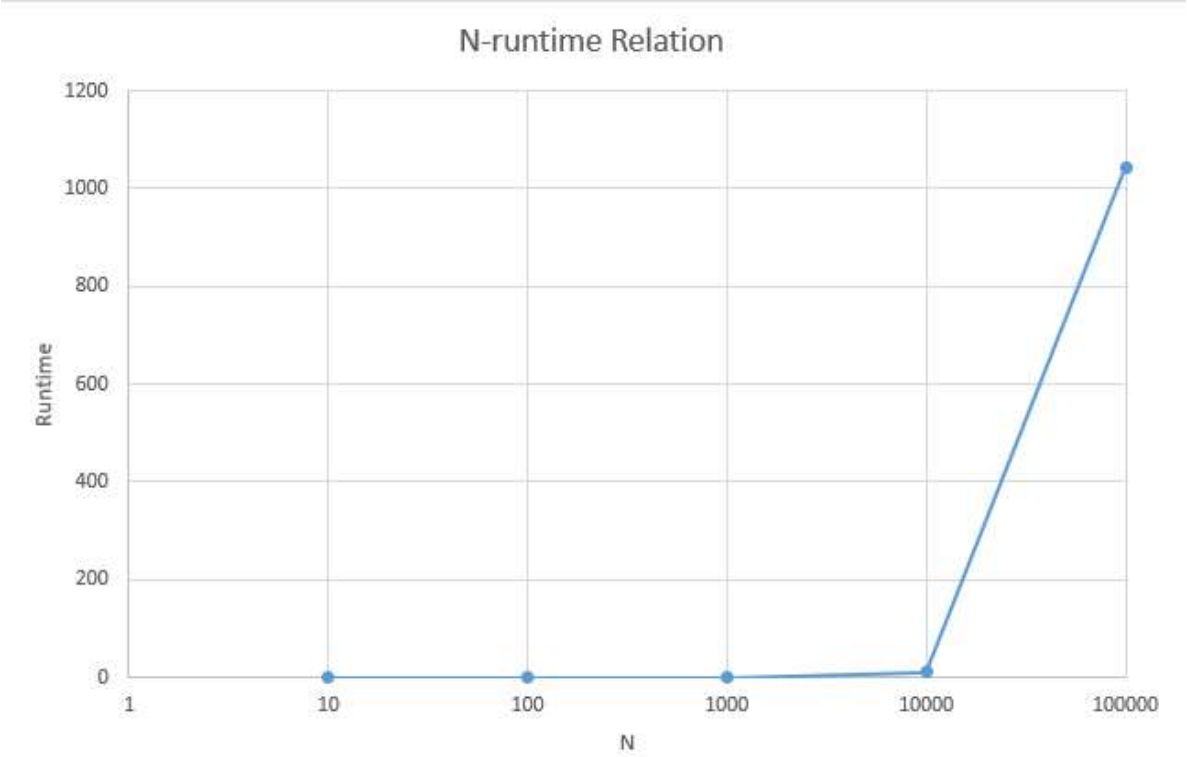
N	runtime/nlogn
10	6,92771E-07
100	4,36747E-07
1000	2,40843E-07
10000	2,64921E-07
100000	2,8316E-07
500000	2,82817E-07
1000000	2,87994E-07



Part D in the next page.

D)

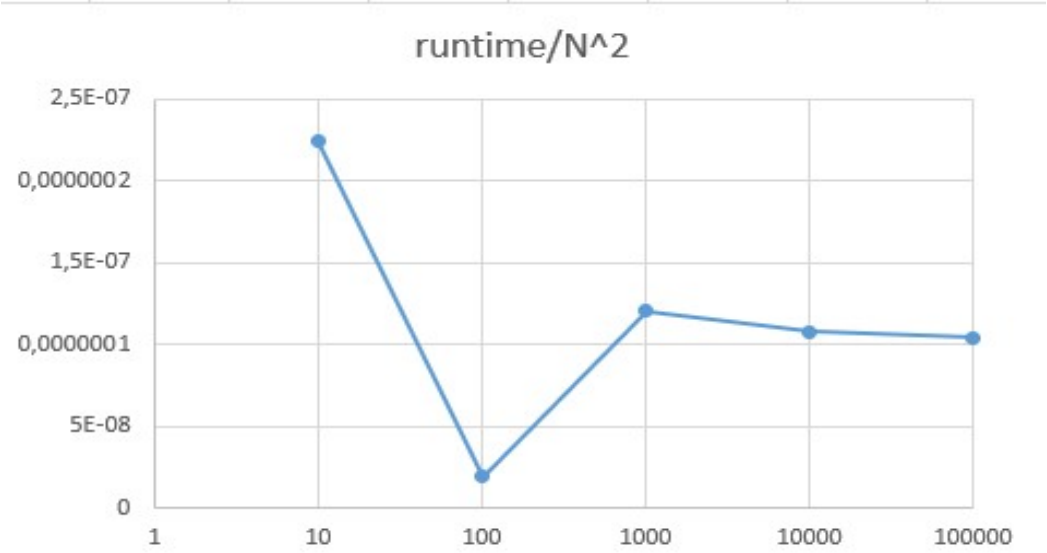
N	Runtime (s)
10	0,0000225
100	0,0001942
1000	0,1206
10000	10,8159
100000	1043,082
500000	Not Executed
1000000	Not Executed



1.

The input array is a sorted orders.So,expected case is the worst case and expected time complexity is n^2.And in my experiment the runtime in this part took too long. For 500K and 1M, the computer failed or the code did not finish execute even though I waited too long. We tried to reorder a sorted list here and this caused the worst case, so running time and n^2 relationship was created. When we look at the runtime / N^2 ratio for different N's, we see similar results.

N	runtime/N^2
10	0,000000225
100	1,9416E-08
1000	1,206E-07
10000	1,08159E-07
100000	1,04308E-07



2.

If the input was reversely sorted (in reverse order alphabetically by country name and the orders with the same country were ascending order of total_profit), we would get a similar result or If we chose the first element as the pivot

3.

Choosing the pivot right in the middle of the order will be the solution to this case.