ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 335E

ANALYSIS OF ALGORITHMS
HOMEWORK 1

CRN : 15175

DATE : 10.12.2020

NAME : ALI KEREM YILDIZ

STUDENT ID : : 150170013

FALL 2020-2021

a)

i. **Best case** :

To evaluate the best case of Quicksort algorithm, the important part is partioning phase. In best case partitioning we have 2 subproblems, each of size no more than n/2 and no less than n/2 -1. In this case, algorithm runs much faster. In other words, in this case while dividing every pivot step, themedian of the list chosen as pivot.

$$T(N) = \big(T(q) + T(n - q - 1)\big) + \Theta(N)$$

Assume that best-case running time is $O(n \log n)$ which provides,

$T(N) \geq c(n\, lgn + 2n)$ for some constant c.

$$T(N) \geq cqlg(q) + 2cq + c(n - q - 1)\lg(n - q - 1) + 2c(n - q - 1)) + \Theta(N)$$

For $q = \frac{n}{2}$

$$\frac{cn}{2}.\lg(\frac{n}{2}) + cn + c\left(\frac{n}{2} - 1\right)\lg(\frac{n}{2} - 1)cn - 2c + \Theta(N)$$

$$\geq \left(\frac{cn}{2}\right) lgn - \frac{cn}{2} + c\left(\frac{n}{2} - 1\right)(lgn - 2) + 2cn - 2c\, \Theta(N)$$

$$= cnlgn + \frac{cn}{2} - lgn + 2 - 2c + \Theta(N)$$

Taking c large enough dominates above term and it makes this greater than $cn\lg n$. This proof is the base logic for proving the bound but it is too theoratical. We can express this idea with another way.

2.Method:

Since we know by dividing two same size subproblems gives the best case for Quicksort. Thus, we can easily say that :

$$T(n) = 2T(n/2) + O(n)$$

$$T(n) = 2T(n/2) + O(n)$$

$$T(n/2) = 2T(n/4) + O(n/2)$$

$$T(n/4) = 2T(n/8) + O(n/4)$$

$$\ldots$$

$$T(n/(n/2)) = 2T\left(\frac{n}{n}\right) + O(1)$$

$$T(n) = 2T(n/2) + O(n) \quad , \quad T(n/2) = 2T(n/4) + O(n/2)$$

$$T(n) = 2[2[T(n/4) + O(n/2)]] + O(n)$$

$$T(n/4) = 2T(n/8) + O(n/4) \rightarrow T(n) = 2[2[2T(n/8) + O(n/4)] + O(n/2)] + O(n)$$

According to evaluated pattern, we can write in general,

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + cn\,(1^{k-1} + 1^{k-2} + \cdots 1)$$

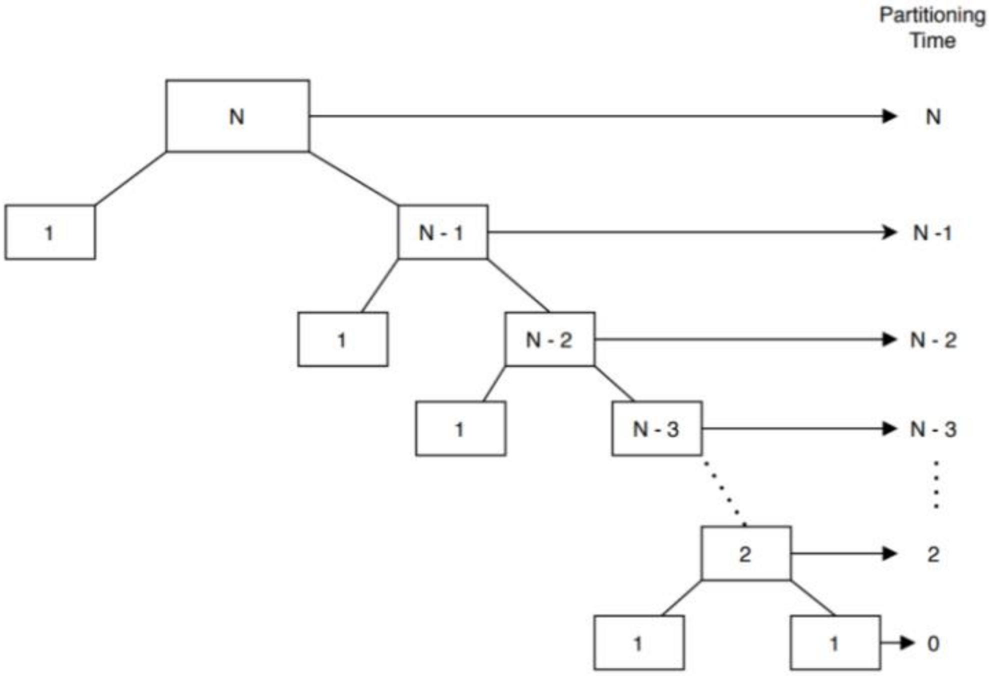
where $k = \log_2 n$ and $O(n) = cn$

Since $T\left(\frac{n}{n}\right) = T(1)$, which means it is already sorted because it has 1 element.

$$T(n) = cnk => cn\log_2 n => O(nlogn)$$

ii. **Worst-Case** :

The worst case occurs when partitioning route produces one sub-problem with n-1 elements and one with 0 elements. The worst case occurs when thechosen pivot is either largest or smallest. Below figure[1] shows how partition happens in worst case scenario



$$T(n) = \big(T(q) + T(n - q - 1)\big) + O(n)$$

Now, assume that $T(n) \leq cn^2$ for some constant c. Then,

$$T(n) \leq cq^2 + c(n - q - 1)^2 + O(n)$$

As we mentioned above we should choose $q = n - 1$

$$T(n) \leq cq^2 + c(n - (n - 1) - 1)^2 + O(n)$$

$$= T(n) \leq c(n - 1)^2 + O(n)$$

$$= T(n) \leq cn^2 - c(2n - 1) + O(n)$$

$$= T(n) \leq cn^2$$

Since we can pick constant c large enough so that $c(2n - 1)$ term dominates the $O(n)$ term. Thus, $T(n) = O(n^2)$

And in another way we can simply say that:

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

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

$$\dots$$

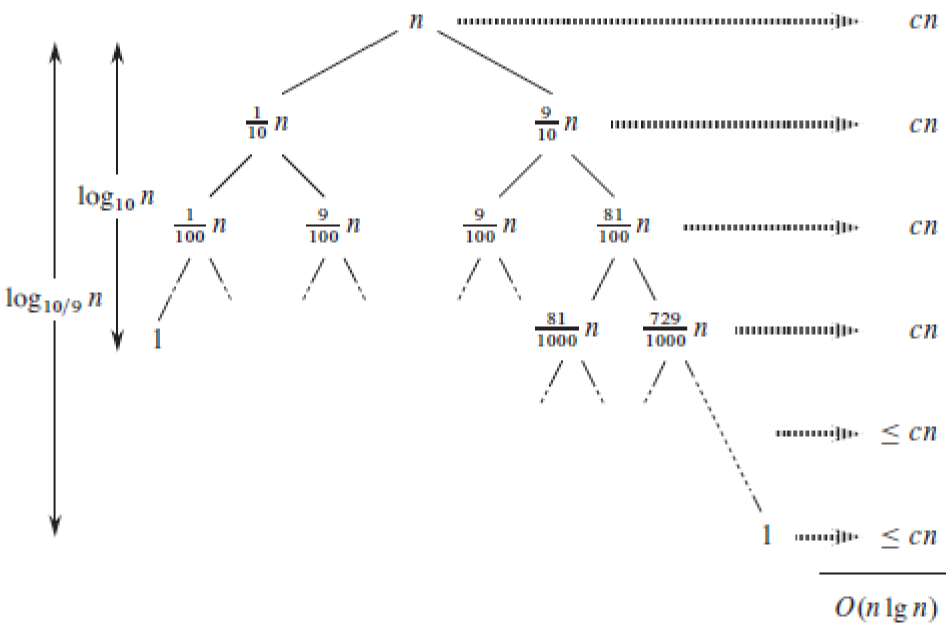
$$T(1) = T(0) + O(1)$$

By adding both sides :

$T(n) = n + n - 1 + n - 2 + \cdots 1 \ = n(n+1)/2$ Thus, $T(n) = O(n^2)$

iii. **Average Case** :

The average running time of Quicksort is much closer to the best case than worst case. We can call average case by almost best case. For example, assume $T(n) = T(9n/10) + T(n/10) + cn$ For better understanding of running time on average case we can visualize it by using recursion tree [2]



Notice that each level of the three has cost of cn , until reaches boundary condition at depth $\log_{10} n = O(lgn)$, and recursion terminates $\log_{10/9} n = O(lgn)$. Although it is quite unbalanced with a 9-to-1 split at every recurison, it yields $O(nlogn)$.

Now we demonstrated that visually, now lets show it with an algebraic way.

In average case , we may have slightly different partition procedure. Partition generates splits (0:n-1 , 1:n-2, 2:n-3, … n-2:1, n-1:0) each with probability $1/n$. $T(n)$ is the expected running time where:

$$T(n) = \frac{1}{n}.\sum_{k=0}^{n-1}[T(k) + T(n-1-k)] + O(n)$$

$$\left[\left(T(0) + T(n-1)\right) + \left(T(1) + T(n-2)\right) + \cdots \left(T(n-2) + T(1)\right) + \left(T(n-1) + T(0)\right)\right]$$
$$= 2\left(T(n-1) + T(n-2) + \cdots + T(1) + T(0)\right)$$

So, we show that

$$\frac{1}{n}.\sum_{k=0}^{n-1}[T(k) + T(n-1-k)] = \frac{2}{n}.\sum_{k=0}^{n-1}T(k)$$

$$T(n) = \frac{2}{n}.[\sum_{k=0}^{n-1}T(k)] + cn$$

By multiplying both sides with n, we get :

$$nT(n) = 2[\sum_{k=0}^{n-1}T(k)] + cn^2 \quad \rightarrow \ldots (1)$$

Since it is recursive, we can make an algebraic manipulation in order to solve this problem. By substituting n by n-1, we get :

$$(n-1)T(n-1) = 2[\sum_{k=0}^{n-2} T(k)] + c(n-1)^2 \quad \rightarrow \quad \ldots(2)$$

By substracting ...(2) from ...(1) :

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

$$\rightarrow nT(n) = (n+1)T(n-1) + 2cn - c$$

Can be omitted for simplification because it is constant

By dividing both sides by n(n+1)

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2c}{n+1}$$

Now, we can telescope.

$$\frac{T(n)}{n+1} = \cancel{\frac{T(n-1)}{n}} + \frac{2c}{n+1}$$

$$\cancel{\frac{T(n-1)}{n}} = \cancel{\frac{T(n-2)}{n-1}} + \frac{2c}{n}$$

$$\cancel{\frac{T(n-2)}{n-1}} = \cancel{\frac{T(n-3)}{n-2}} + \frac{2c}{n-1}$$

$$\ldots$$

$$\cancel{\frac{T(2)}{3}} = \frac{T(1)}{2} + \frac{2c}{3}$$

By adding all of these equations,

$$\frac{T(n)}{n+1} = \frac{T(1)}{2} + 2c \sum_{i=3}^{n+1} \frac{1}{i}$$

The sum is about $\approx \ln(n+1)$

$$\frac{T(n)}{n+1} = O(logN) \quad \rightarrow \quad T(n) = O(nlogn)$$

b) This solution does not give us the desired outputs. In desired output, when country name's are same, they sorting by total profit in descending order. But in this method, for example we first sort by total profit in descending order and then, in second step when sorting the "sorted_by_profits.txt", we are always encountering the sale which has same country name with another but greater total profit first then we encountered other one. But this may not be as same as in the 'sales.txt'. For example, when N=20 :

1)

sorted.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

Country	Item Type	Order ID	Units Sold	Total Profit
Algeria	Cosmetics	761723172	9669	1.68115e+006
Djibouti	Clothes	880811536	562	41273.3
Ethiopia	Cosmetics	807785928	662	115102
France	Cosmetics	324669444	5758	1.00114e+006
Ghana	Office Supplies	601245963	896	113120
Morocco	Clothes	667593514	4611	338632
Papua New Guinea	Clothes	647164094	9092	667716
Papua New Guinea	Meat	940995585	360	20592

sorted_by_profit_country.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

Country	Item Type	Order ID	Units Sold	Total Profit
Algeria	Cosmetics	761723172	9669	1.68115e+006
Djibouti	Clothes	880811536	562	41273.3
Ethiopia	Cosmetics	807785928	662	115102
France	Cosmetics	324669444	5758	1.00114e+006
Ghana	Office Supplies	601245963	896	113120
Morocco	Clothes	667593514	4611	338632
Papua New Guinea	Meat	940995585	360	20592
Papua New Guinea	Clothes	647164094	9092	667716

The left side we can see sorted as desired, and right side given methods output. We can see that in 'sorted.txt' we encountering first Papua New Guine sale with 667716 total profit. But we cannot see the same results on the right side. The reason for that is we encounter the greater total profit first in "sorted_by_profits.txt" the new comer sale with same country name but lesser total profit comes first in sorting for country part. As we can see in "sort_by_profit.txt" :

*sorted_by_profit.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

Uganda	Cosmetics	842238795	6031	1.04861e+006
France	Cosmetics	324669444	5758	1.00114e+006
Samoa	Household	937431466	5657	937535
Togo	Cosmetics	563681733	4806	835619
Taiwan	Cereal	498071897	9397	832480
Antigua and Barbuda	Office Supplies	286891067	6297	794996
Greece	Cereal	887124383	8674	768430
Albania	Baby Food	752525556	7890	756335
China	Office Supplies	198927056	5791	731114
Mali	Household	363086831	4317	715456
Switzerland	Office Supplies	830410039	5639	711924
Solomon Islands	Household	101328551	4225	700209
Papua New Guinea	Clothes	647164094	9092	667716
Romania	Cereal	633134210	7337	649985
Italy	Cereal	294530856	7080	627217
Pakistan	Meat	500371730	9969	570227
Nepal	Meat	179137074	9496	543171
Serbia	Clothes	925136649	7348	539637
Tonga	Baby Food	839094388	5531	530202
Democratic Republic of the Congo	Cosmetics	584356629	2967	515872
Vanuatu	Cereal	572335612	5681	503280
Lebanon	Meat	704205024	8770	501644
Tunisia	Cosmetics	479969346	2450	425982
Singapore	Snacks	176461303	7676	423255
Turkmenistan	Vegetables	116205585	6670	421077
South Korea	Meat	297876536	7141	408465
Mauritius	Clothes	349235904	5520	405389
Brunei	Cereal	153842341	4222	374027
Montenegro	Clothes	902511680	2117	155472
Malaysia	Beverages	955894076	9154	143352
Liberia	Baby Food	146634709	1324	126919
Ethiopia	Cosmetics	807785928	662	115102
Ghana	Office Supplies	601245963	896	113120
United States of America	Personal Care	190777862	4264	106856
Indonesia	Household	520480573	623	103250
Dominica	Beverages	438011872	6301	98673.7
New Zealand	Beverages	940980136	5788	90640.1
Sweden	Beverages	265081918	2485	38915.1
Albania	Personal Care	104191863	1543	38667.6
Dominican Republic	Baby Food	824714744	274	26265.6
Netherlands	Fruits	845056617	9887	23827.7
Tanzania	Fruits	156530129	9599	23133.6
Tanzania	Beverages	659878194	1476	23114.2
East Timor	Cereal	156295812	259	22944.8
The Gambia	Fruits	862861335	8699	20964.6
Papua New Guinea	Meat	940995585	360	20592

We can see that quicksort is not stable algorithm because of its this kind of behaviour

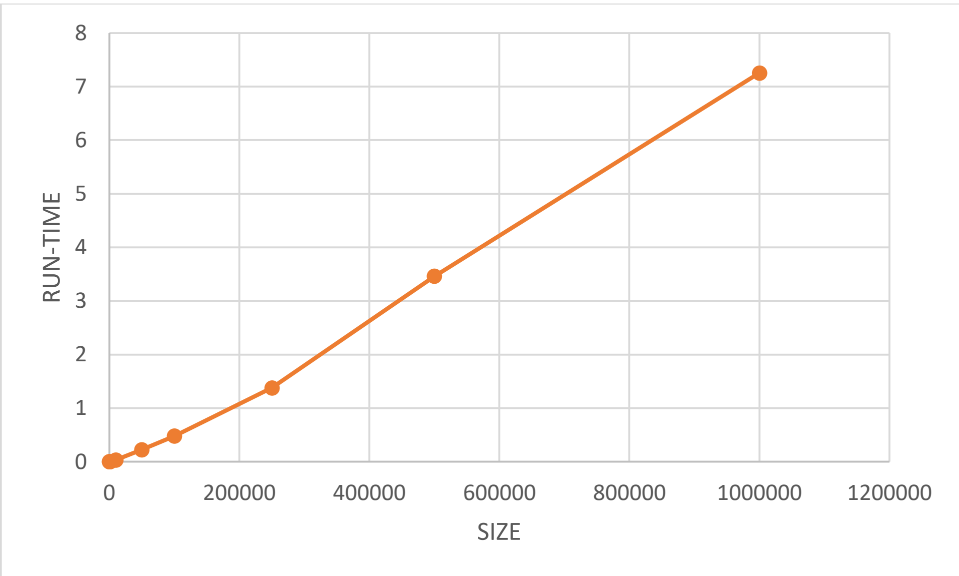
2)

If we use bubble sort, insertion sort, merge sort in general stable algorithms , we can provide desired output.

c) In this part, we run the algorithm for different N values {10, 100, 1000, 10000, 50000, 10000, 250000, 500000, 1000000} 10 times and evaluate the average time. Average execution times respect to size (N) shown below as table.

N	1	2	3	4	5	6	7	8	9	10	verage-tim
10	0	0	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0	0
1000	0	0	0	0	0	0	0	0	0	0	0
10000	0,03	0,03	0,04	0,03	0,04	0,03	0,03	0,03	0,03	0,03	0,032
50000	0,23	0,22	0,2	0,25	0,19	0,19	0,26	0,26	0,19	0,24	0,223
100000	0,44	0,52	0,46	0,47	0,49	0,46	0,46	0,48	0,48	0,5	0,476
250000	1,25	1,37	1,26	1,43	1,33	1,3	1,52	1,46	1,47	1,38	1,377
500000	3,35	3,27	3,33	3,15	3,31	3,26	3,57	3,99	3,78	3,58	3,459
1000000	7,14	7,38	7,19	7,31	7,21	7,04	6,21	7,45	7,14	8,47	7,254

For visualize this output to better understand, run-time N relation demonstrated on belown plot.



According to above plot, we can of obviously say that, algorithm does not run for worst case which means datas are not already sorted and first or last element is selected as pivot. Because as we proof in the first part of this question, worst case upper bound is $O(n^2)$. But above plot's upper bound is $O(nlogn)$. We knew the pivot is selected as last element, so we can say that datas are not already sorted.

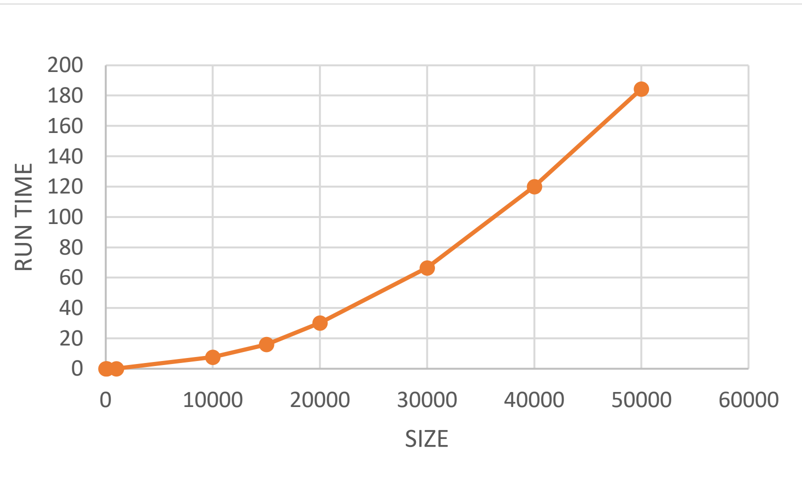
However, we cannot say it runs definetly for best-case or average case. Since we do not know whether it is divided perfectly by pivot which yields us best case, or balanced partition. Because both cases upper bound $O(nlogn)$. But we can definetly say that its upper bound is $O(nlogn)$

d)

In this part, we are going to run the algorithm for different N values {10, 100, 1000, 10000, 15000, 20000, 30000, 40000, 50000}. The table represantation of yielding results shown below.

N	1	2	3	4	5	6	7	8	9	10	average
10	0	0	0	0	0	0	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0	0
1000	0,09	0,08	0,1	0,08	0,08	0,09	0,08	0,09	0,08	0,08	0,085
10000	7,64	7,71	7,7	7,57	7,87	7,61	7,67	7,61	7,68	7,66	7,672
15000	≈16	≈16	≈16	≈16	≈16	≈16	≈16	≈16	≈16	≈16	16
20000	≈30.21	≈30.21	≈30.21	≈30.21	≈30.21	≈30.21	≈30.21	≈30.21	≈30.21	≈30.21	30,21
30000	≈66.42	≈66.42	≈66.42	≈66.42	≈66.42	≈66.42	≈66.42	≈66.42	≈66.42	≈66.42	66,42
40000	≈120	≈120	≈120	≈120	≈120	≈120	≈120	≈120	≈120	≈120	120
50000	≈184.31	≈184.31	≈184.31	≈184.31	≈184.31	≈184.31	≈184.31	≈184.31	≈184.31	≈184.31	184,31

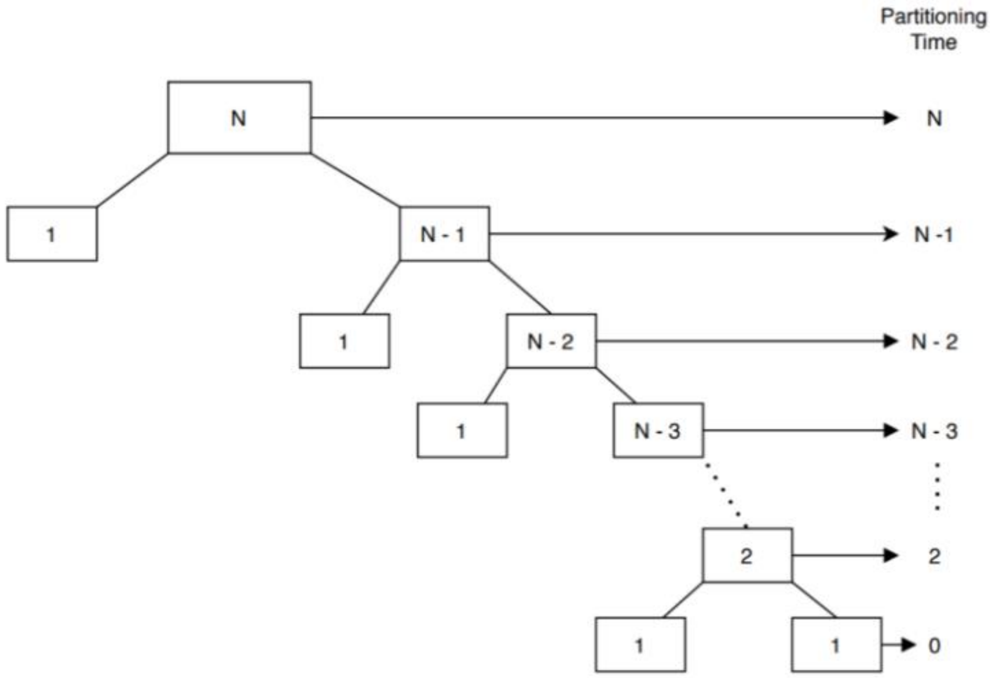
We need to show this results on plot to make more sense.



1.

In this case, we cannot use large inputs because computer cannot complete the sorting in a reasonable time. Hence we do not use large input sizes such as 100k, 500k and 1M. The reason for this is, when we run algorithm on ‘sorted.txt’, we are dealing with worst case. Because since datas are already sorted and pivot is the last element. In each recursion we divide the N-sized main problem to N-1 sized subproblem. Which make a differ with best case and average case. Because the step size we are going to make partititon is increase and it is related to N.(not log(N) as best case or average case). Since every partition step we deal with N elements. So as we explain in part a, upper bound of this case is $O(n^2)$.

If we compare this with the results we have obtained at (c), we can obviously say that run time is greater than the results in part c. In part c we are not trying to sort the already sorted datas, so we have less partition.



In worst case partition procedure shown above.

The differences can be explained by using the equations we used in part a. In this case $T(n) = T(n-1) + O(n)$, in previous case, time complexity may be $T(n) = [T(k) + T(n-1-k)]$ where k is not equal 0 as in worst case. In brief, since datas were already sorted.

2)

In quicksort, if the pivot is the first element or last element and already sorted or reverse sorted worst case occurs and when all elements are same, worst case occurs and these cases give the similar results.

3)

Since we run algorithm on already sorted data, to avoid of worst case, we should not select pivot as first or last element. We must pick a pivot element from the middle of the array.

References :

[1] https://www.baeldung.com/cs/quicksort-time-complexity-worst-case

- [2] Cormen, T. H., & Cormen, T. H. (2001). *Introduction to algorithms*. Cambridge, Mass: MIT Press.