BLG335E Algorithm Analysis Assignment1 Report

Mihriban Nur Koçak - 150180090

24.11.2021

Question 1

In general the running time of the Deterministic Quick Sort is (L is the left part and R is the right part):

$$\mathbf{T(n) = T(L) + T(R) + O(n)}$$

In ideal word, if pivot splits array into half, we obtain best case scenario:

$$\mathbf{T(n) = 2T(n/2) + O(n)}$$

We can solve this recurrence using Master Theorem:

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

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

Figure 1: Master Theorem from Lecture Slides

$$\text{d} = 1, \text{a} = 2, \text{b} = 2$$
$$2 = 2^1$$
$$a = b^d$$
$$\mathbf{T(n) = O(nlogn) \text{ for best case}}$$

To obtain the worst case scenario, we always select the max or min element as the pivot, so that the array is splitted as following:

$$\mathbf{T(n) = T(1) + T(n\text{-}1) + O(n)}$$

We can ignore T(1) since it is a very small value and then solve this recurrence using substitution:

$$\mathrm{T(n) = T(n\text{-}1) + O(n)}$$

$$\mathrm{T(n\text{-}1) = T(n\text{-}2) + O(n\text{-}1)}$$

this recurrence goes like that until:

$$\mathrm{T(2) = T(1) + O(2)}$$

by summing all of them, we can obtain:

$T(n) = T(1) + \sum_{i=2}^{n} O(i) = T(1) + O(n^2) = O(n^2)$ **for worst case**

Question 2

The running time of the comparison based sorting algorithms is equal to the total number of comparisons made. So to obtain the asymptotic upper bound for the Quick Sort with randomized pivot selection, we need to calculate the total number of comparisons made. To do that we need a random variable:

a is one of the elements of array, b is one of the elements of array
X(a,b) = 1, if a and b are ever compared
X(a,b) = 0, if a and b are never compared

So we can calculate expected total number of comparisons by using following equation:

$E[X] = \sum_{a=1}^{n} \sum_{b=a+1}^{n} (P(X(a,b) = 1).1 + P(X(a,b) = 0).0)$

$E[X] = \sum_{a=1}^{n} \sum_{b=a+1}^{n} P(X(a,b) = 1)$

$P(X(a,b) = 1)$ is the probability of a and b are ever compared. It is equal to the the probability that either a or b are picked first out of all of the b-a+1 numbers between them:

$P(X(a,b) = 1) = 2/(b-a+1)$ So the expected total number of comparisons:

$E[X] = \sum_{a=1}^{n} \sum_{b=a+1}^{n} (2/(b-a+1))$

Using Harmonic Series Formula we can rewrite the equation as:

$E[X] = \sum_{a=1}^{n} 2ln(n) = 2ln(n).n = O(nlogn)$

So we can say that the execution time of Quicksort with randomized pivot selection is $O(nlogn)$ for all cases.

Question 3

I have executed Quicksort with randomized pivot selection 5 times for every amount of unsorted data. The execution times of each execution and the calculated average execution time can be seen in Figure 2.

N	Execution 1(ms)	Execution 2(ms)	Execution 3(ms)	Execution 4(ms)	Execution 5(ms)	Average Execution Time (ms)
1000	0	0	0	0	0	0
10000	15.625	15.625	15.625	15.625	15.625	15.625
100000	234.375	250	312.5	250	250	259.375
500000	1671.88	1796.88	1812.5	1859.38	1828.12	1793.752
1000000	3546.88	3890.62	3906.25	3593.75	4015.62	3790.624

Figure 2: Application of quick sort with randomized pivot selection on unsorted data

To understand the relation between number of data sorted with randomized pivot selection Quicksort and the average execution times of them, I have obtained a plot using the results in Figure 2, x axis represents number of data and y axis represents execution time. As can be seen from Figure 3 the relation between number of data n and execution time similar to $nlogn$ so we can say that the practical result we obtained is same as the theoretical result we obtained in Question 2.

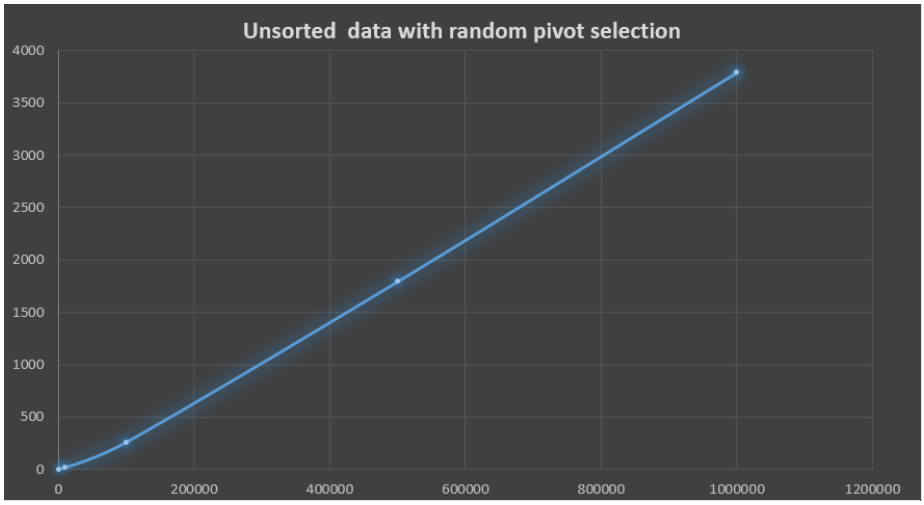

Figure 3: Plot of application of quick sort with randomized pivot selection on unsorted data

Then, I have executed Quicksort with deterministic pivot selection 5 times for every amount of unsorted data. The execution times of each execution and the calculated average execution time can be seen in Figure 4.

N	Execution 1(ms)	Execution 2(ms	Execution 3(ms)	Execution 4(ms)	Execution 5(ms)	Average Execution Time (ms)
1000	0	0	0	0	0	0
10000	15.625	15.625	15.625	15.625	15.625	15.625
100000	234.375	250	265.625	250	250	250
500000	1703.12	1859.38	1843.75	1843.75	1875	1825
1000000	3468.75	4000	4015.62	3565.25	3921.88	3794.3

Figure 4: Application of quick sort with deterministic pivot selection on unsorted data

To understand the relation between number of data sorted with deterministic pivot selection Quicksort and the average execution times of them, I have obtained a plot using the results in Figure 4, x axis represents number of data and y axis represents execution time. As can be seen from Figure 5, the relation between number of data n and execution time is similar to *nlogn* so we can claim that the algorithm works similar to the best case scenario since the last element is not always max or min element for unsorted data. Therefore we can say that the practical result we obtained is same as the theoretical result we obtained in Question 1.

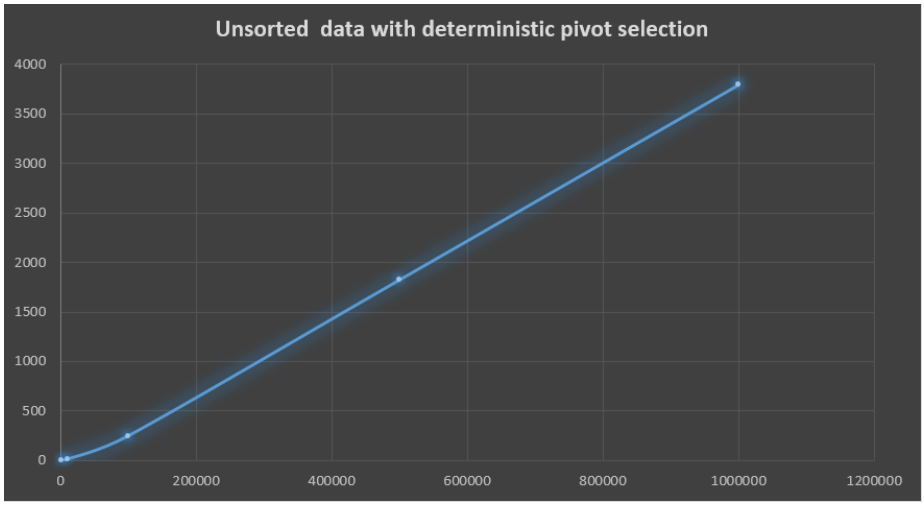

Figure 5: Plot of application of quick sort with deterministic pivot selection on unsorted data

In Figure 6, we can compare the behaviour of Quicksort with different pivot selection methods. The Seri1 is the plot of Quicksort with randomized pivot selection, and the Seri2 is the plot of Quicksort with deterministic pivot selection. As can be seen, both versions behave in similar ways for unsorted data.

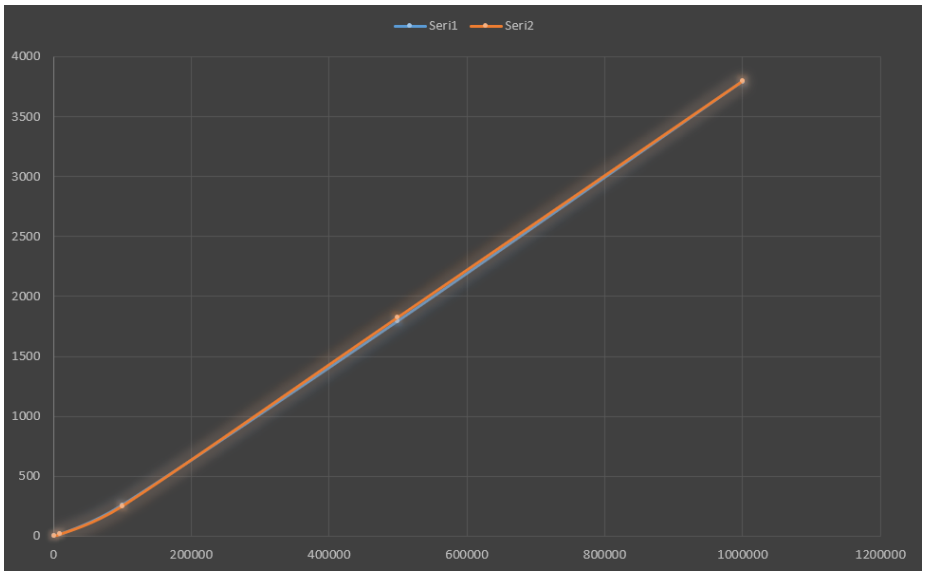

Figure 6: Comparison of Quicksort with both deterministic and randomized pivot selection on unsorted data

Question 4

I have executed Quicksort with randomized pivot selection 5 times for every amount of sorted data. The execution times of each execution and the calculated average execution time can be seen in Figure 7. If we compare the results with the ones we obtained in Question 3, we can see the similarity.

N	Execution 1(ms)	Execution 2(ms	Execution 3(ms)	Execution 4(ms)	Execution 5(ms)	Average Execution Time (ms)
1000	0	0	0	0	0	0
10000	15.625	31.25	15.625	15.625	15.625	18.75
100000	187.5	218.75	218.75	218.75	203.125	209.375
500000	1234.38	1375	1265.62	1250	1312.5	1287.5
1000000	2656.25	2703.12	2687.5	2750	2750	2709.374

Figure 7: Application of quick sort with randomized pivot selection on sorted data

Also by looking the plot Figure 8, we can say that the plot is similar to *nlogn* and it is similar to the one we obtained in Question 3. So, as we have found in Question 2, the expected execution time of the Quicksort with randomized pivot selection doesn't depend on the order of the data, whether it is sorted or not. It always works with O(nlogn).

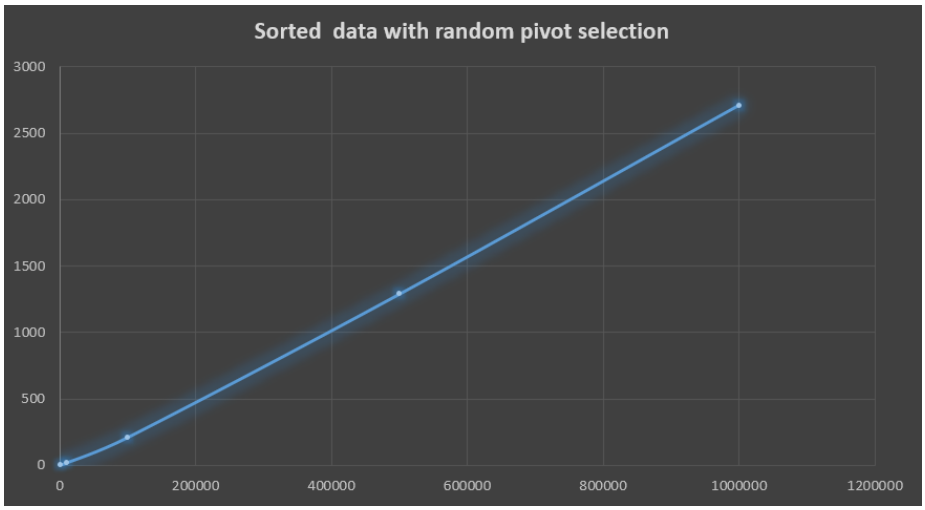

Figure 8: Plot of application of quick sort with randomized pivot selection on sorted data

Then, I have executed Quicksort with deterministic pivot selection 5 times for every amount of sorted data. We can claim that the algorithm works in the worst case scenario since the data is sorted so that the last element which is selected as pivot is always the max element. Therefore, using the result we obtained in Question 1, we can say that the execution time is $O(n^2)$.

Due to the large complexity, I am able to calculate exact values for only N=1000 and N= 10000. Since the executions of other amount of data takes very long time, I filled their values by making some calculations. For example for execution time of 100k, I have done following calculations:

for N= 10^5 and N= 10^4, $(10^5)^2/(10^4)^2 = 100$, I take this value as 95
$T(10^5)/T(10^4) = 95$ should hold
$T(10^5)/1796.874 = 95$
$T(10^5) = 170703.03ms$

The table can be seen in Figure 9. If we compare the results with the ones we obtained in Question 3, we can see that there are huge differences.

N	Execution 1(ms)	Execution 2(ms	Execution 3(ms)	Execution 4(ms)	Execution 5(ms)	Average Execution Time (ms)
1000	15.625	15.625	31.25	15.625	15.625	18.75
10000	1781.25	1812.5	1828.12	1765.62	1796.88	1796.874
100000	170703.03	170703.03	170703.03	170703.03	170703.03	170703.03
500000	3414060.6	3414060.6	3414060.6	3414060.6	3414060.6	3414060.6
1000000	13656242.4	13656242.4	13656242.4	13656242.4	13656242.4	13656242.4

Figure 9: Application of quick sort with deterministic pivot selection on sorted data

Also by looking the plot Figure 10, we can say that the plot is similar to n^2.

So, as we have found in Question 1, the execution times of worst case and best case scenario differs. Since this version works in worst case scenario with sorted data, it works with $O(n^2)$.

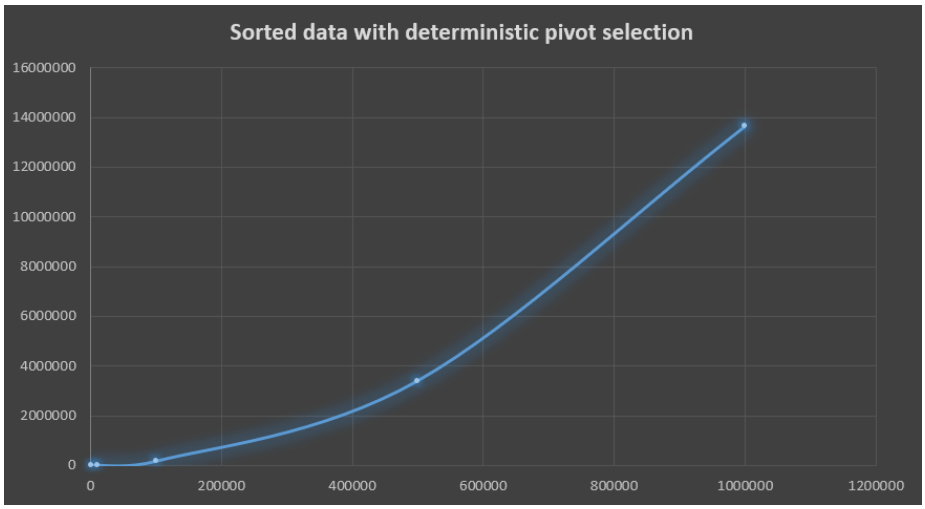

Figure 10: Plot of application of quick sort with deterministic pivot selection on sorted data

In Figure 11, we can compare the behaviour of Quicksort with different pivot selection methods. The Seri1 is the plot of Quicksort with randomized pivot selection, and the Seri2 is the plot of Quicksort with deterministic pivot selection. As can be seen, Quicksort with randomized pivot selection behaves much better than Quicksort with deterministic pivot selection for sorted data in terms of execution time.

In conclusion, for unsorted data, since the execution times of randomized and deterministic pivot selection don't differ considerably, I would prefer using Quicksort with deterministic pivot selection due to its implementation's simplicity since it doesn't need extra swapping.

On the other hand for sorted data, I would definitely prefer using Quicksort with randomized pivot selection since it has a much smaller time complexity than the deterministic one.

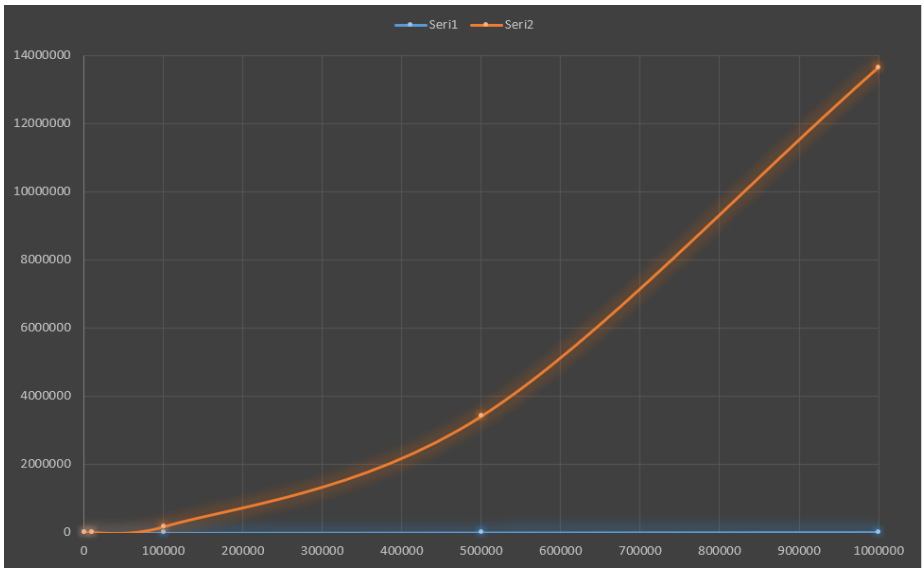

Figure 11: Comparison of Quicksort with both deterministic and randomized pivot selection on sorted data

Question 5

I prefer to analyze Dual Pivot Quicksort algorithm with deterministic pivot selection. In this case, the left pivot is chosen as the first element and the right pivot is chosen as the last element. In this way, at every iteration the array is splitted into three parts and O(n) operation is done like standart Quicksort. Let's say SL means the elements smaller than the left pivot, LR means elements between left and right pivot and BR means elements bigger than right pivot. So that we can write general execution time formula as:

$$\mathbf{T(n) = T(SL) + T(LR) + T(BR) + O(n)}$$

In ideal word, if the pivots split array in three equal parts, we obtain best case scenario:

$$\mathbf{T(n) = 3T(n/3) + O(n)}$$

We can solve this recurrence using Master Theorem:

$$\text{d} = 1, \text{a} = 3, \text{b} = 3$$
$$3 = 3^1$$
$$a = b^d$$
$$\mathbf{T(n) = O(nlogn) \text{ for best case}}$$

To obtain worst case scenario, we always select the min element as first pivot and select max element as second pivot so that the array is splitted as following:

$$\mathbf{T(n) = T(1) + T(n\text{-}2) + T(1) + O(n)}$$

We can ignore T(1)s since T(1) is a very small value and then solve this recurrence using substitution:

$$\mathrm{T(n) = T(n\text{-}2) + O(n)}$$

$$\mathrm{T(n\text{-}2) = T(n\text{-}4) + O(n\text{-}2)}$$

this recurrence goes like that until:

$$\mathrm{T(3) = T(1) + O(3)}$$

by summing all of them, we can obtain:

$T(n) = T(1) + (\sum_{i=3}^{n} O(i))/2 = T(1) + O(n^2) = O(n^2)$ for worst case

If we compare standart Quicksort which we analyzed in Question 1 with Dual Pivot Quicksort for worst cases:

Quicksort from Question1: $T(n) = T(1) + \sum_{i=2}^{n} O(i)$ Dual Pivot Quicksort: $T(n) = T(1) + (\sum_{i=3}^{n} O(i))/2$

Mathematically, we can claim that:

$T(1) + (\sum_{i=3}^{n} O(i))/2 < T(1) + \sum_{i=2}^{n} O(i)$

So that in general we may prefer to use Dual Pivot Quick Sort, but as can be seen there are not considerable differences between them in terms of execution times.