BLG335E
ANALYSIS OF ALGORITHMS I
11474

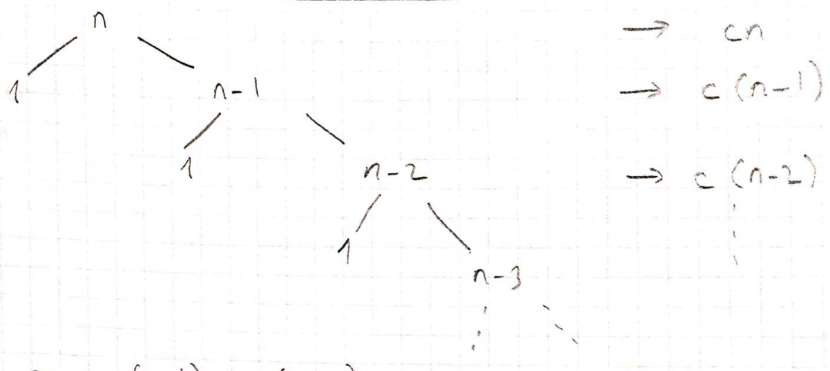
ASSIGNMENT I
REPORT I

IN PLACE QUICKSORT ANALYSIS

ARZUM AYDIN
150190701

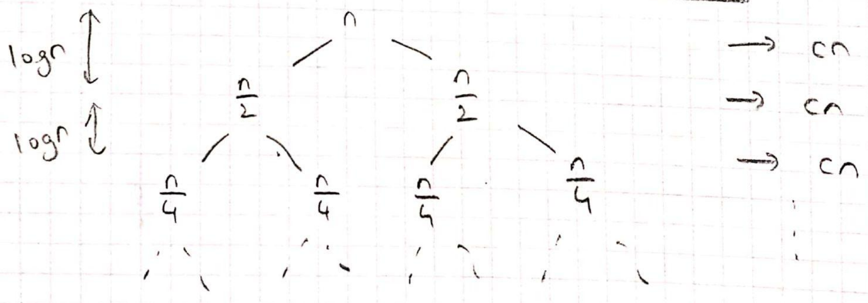
1.

ⓐ <u>Worst case for Deterministic Quicksort</u>



$$T(n) = cn + c(n-1) + c(n-2) + \dots + 1 + 0$$
$$= c.\left(\frac{n.(n+1)}{2}\right) = \frac{c}{2}(n^2+n)$$
$$= O(n^2)$$

<u>Best case for Deterministic Quicksort</u>



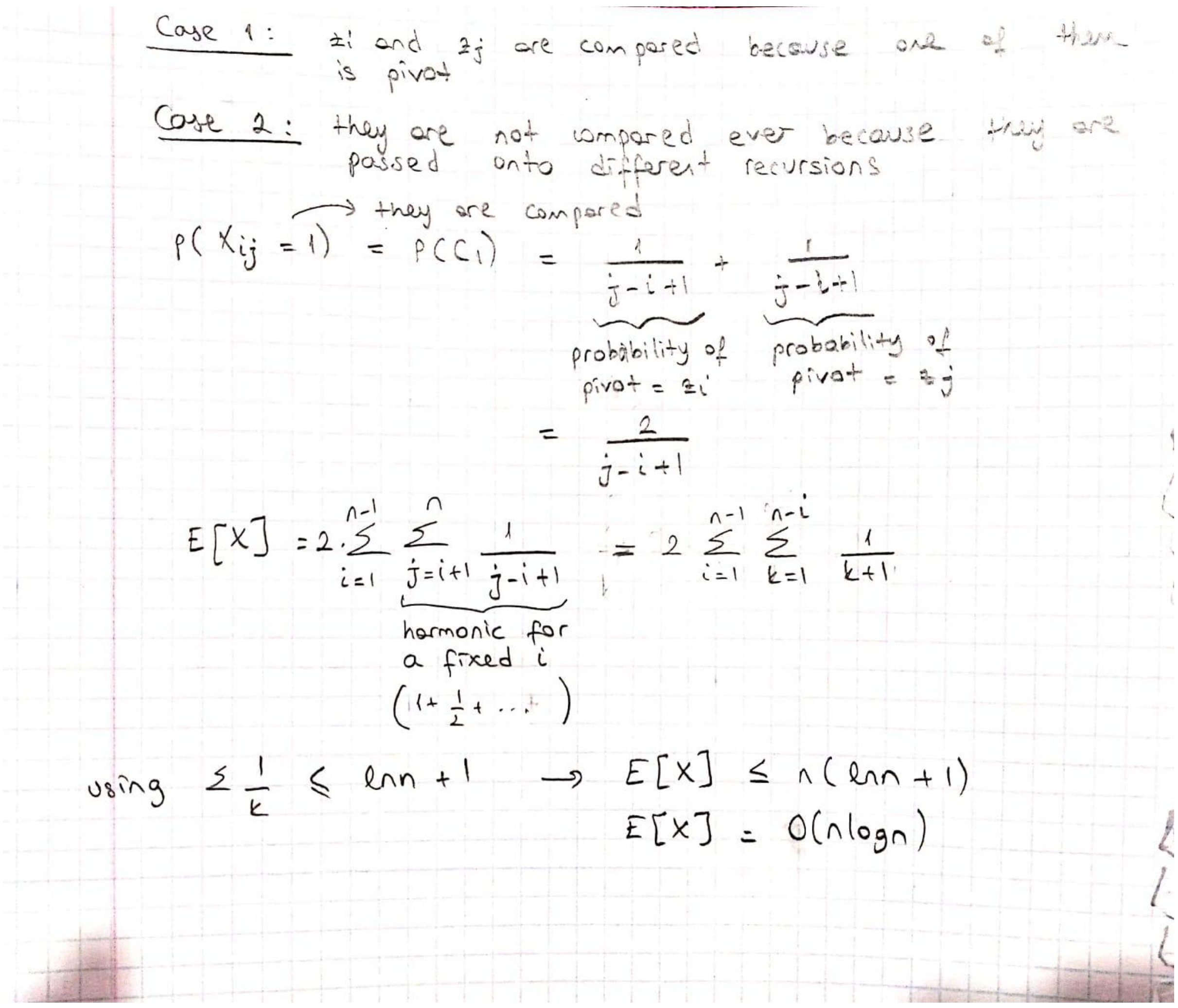
$$T(n) = T\left(\frac{n_1}{2}\right) + T\left(\frac{n_2}{2}\right) + cn$$
$$= cn.\log n$$
$$= O(n\log n)$$

2.

ⓑ <u>Probabilistic Analysis for Quicksort</u>

$$X_{ij} = \begin{cases} 0 & \text{if } z_i \text{ and } z_j \text{ is ever compared} \\ 1 & \end{cases}$$

$$E[X] = E\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{ij}\right]$$
$$= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \underbrace{P(X_{ij}=0).0}_{0} + P(X_{ij}=1).1$$
$$= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P(X_{ij}=1)$$



Case 1 : z_i and z_j are compared because one of them is pivot

Case 2 : they are not compared ever because they are passed onto different recursions

$$P(X_{ij}=1) \xrightarrow{\text{they are compared}} = P(C_1) = \underbrace{\frac{1}{j-i+1}}_{\text{probability of pivot} = z_i} + \underbrace{\frac{1}{j-i+1}}_{\text{probability of pivot} = z_j}$$

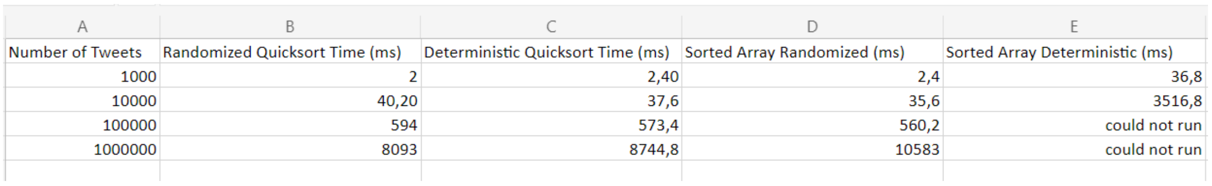
$$= \frac{2}{j-i+1}$$

$$E[X] = 2\cdot\sum_{i=1}^{n-1}\underbrace{\sum_{j=i+1}^{n}\frac{1}{j-i+1}}_{\text{harmonic for a fixed } i \; \left(1+\frac{1}{2}+\dots\right)} = 2\sum_{i=1}^{n-1}\sum_{k=1}^{n-i}\frac{1}{k+1}$$

using $\sum \frac{1}{k} \leq \ln n + 1 \rightarrow E[X] \leq n(\ln n + 1)$

$E[X] = O(n \log n)$

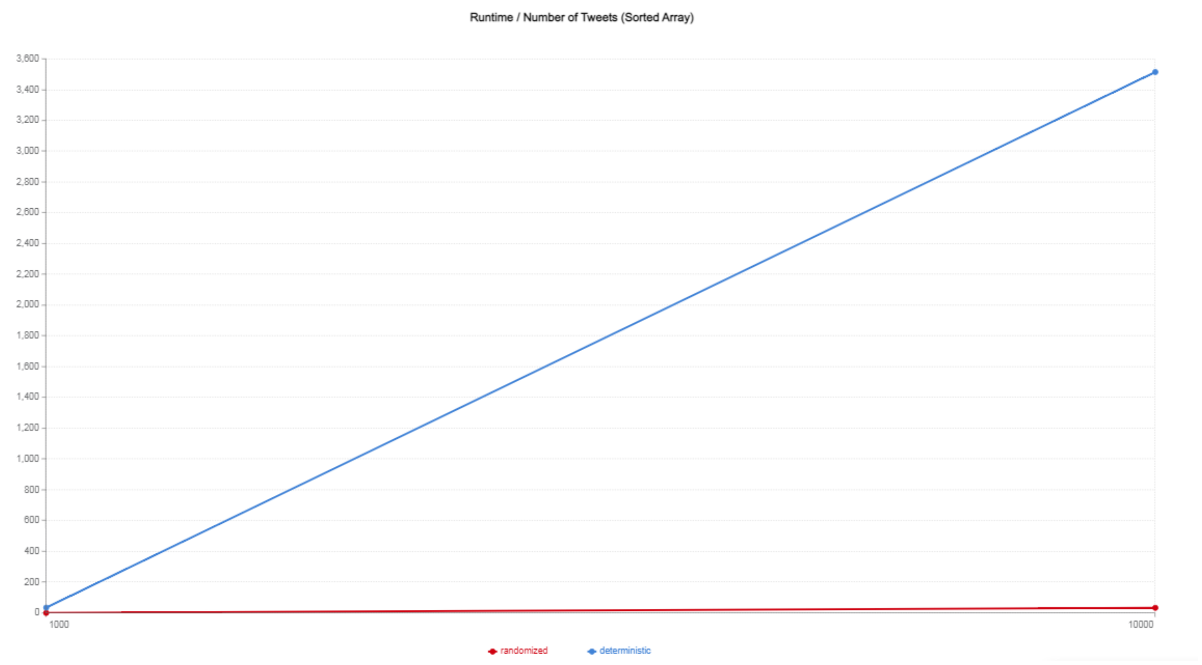
3.

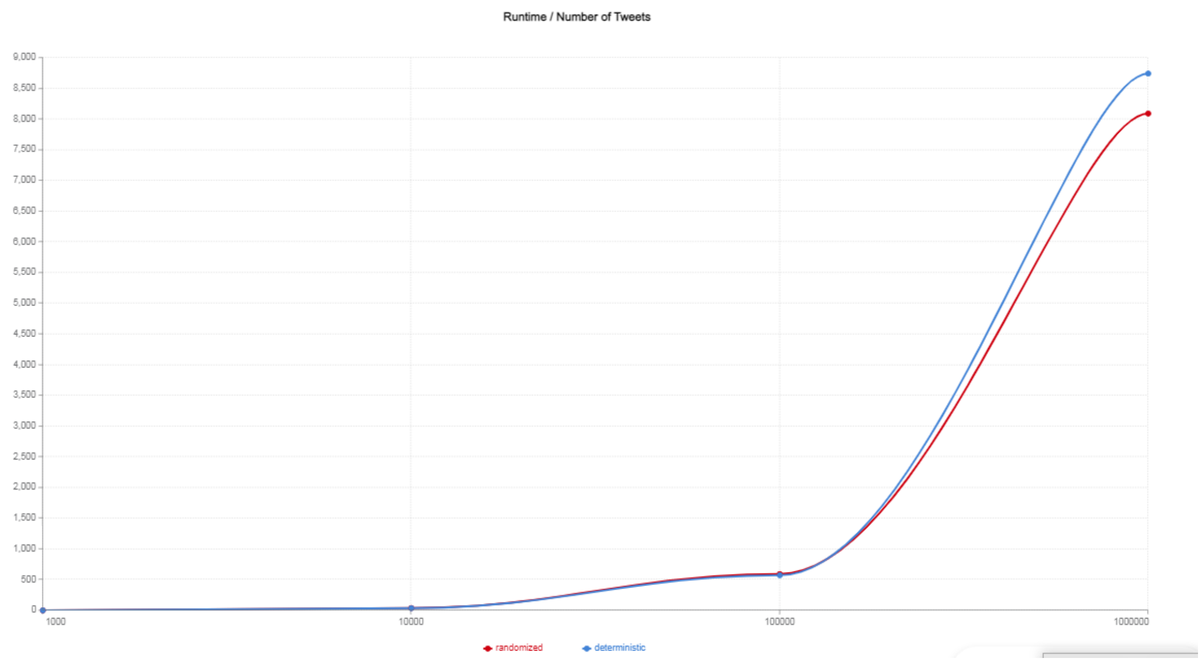


A	B	C	D	E
Number of Tweets	Randomized Quicksort Time (ms)	Deterministic Quicksort Time (ms)	Sorted Array Randomized (ms)	Sorted Array Deterministic (ms)
1000	2	2,40	2,4	36,8
10000	40,20	37,6	35,6	3516,8
100000	594	573,4	560,2	could not run
1000000	8093	8744,8	10583	could not run

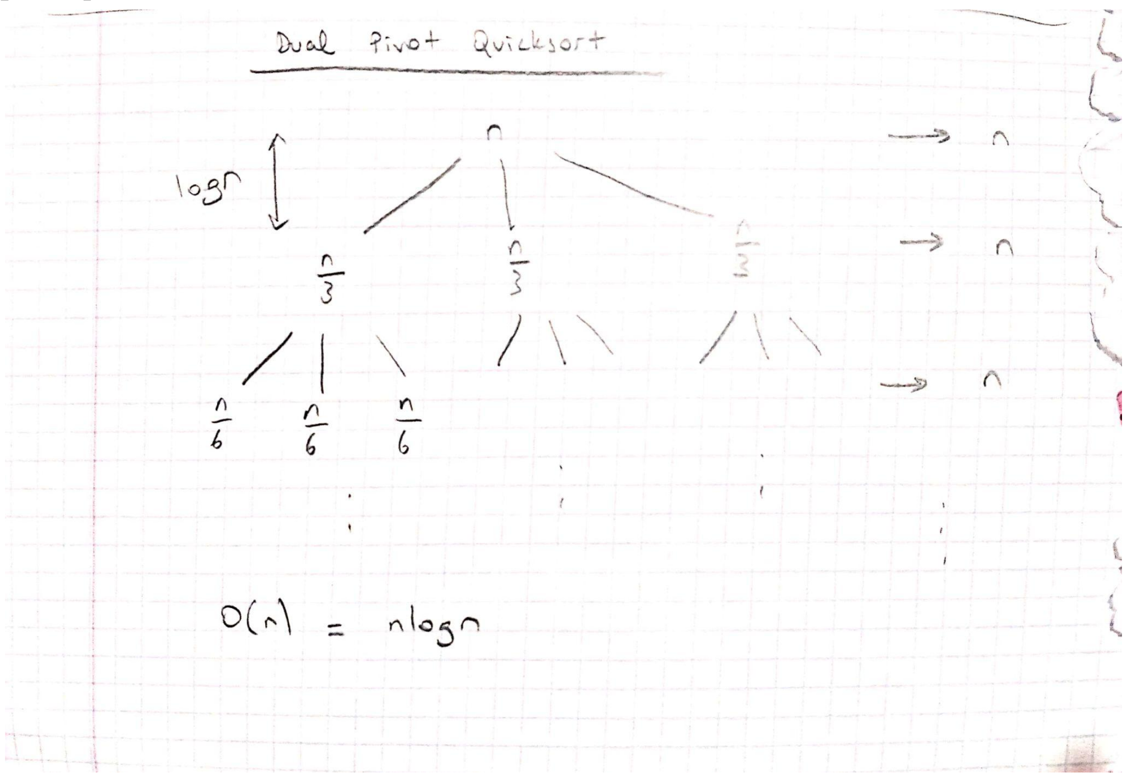
For unsorted array input, deterministic and randomized quicksort runtimes are close to each other. For randomized quicksort, when the input number of tweets is 1000 and gets multiplied by 10, runtime roughly increases 20 times. When it gets multiplied by 10 for the second time, runtime increases approximately 14.77 times. For the third time, it increases 13.62 times. nlogn more or less equals to 30 when n = 10 in binary logarithm. Therefore, it can be seen that the runtime has an upper bound of nlogn. Similar case is true for deterministic pivot selection as well.

For sorted array input, randomized quicksort acts similar to before. However, for the deterministic pivot selection, there is a drastic change. When input is 1000 tweets and it gets multiplied by 10, the runtime increases 95.56 times which is just a little less than 100. Therefore, it can be observed that runtime increases by n*n when n = 10. For the last two values, the computer that was used for the purpose could not run the program, expectedly because of time out. It is clearly seen that for the sorted array, randomized quicksort should be used and in cases where the input is definitely known to be not sorted, the deterministic approach can be used as well. Graphs of sorted input (with 2 inputs) and unsorted input runtimes can be examined as follows:





4. In dual pivot quicksort, if the pivot selection is deterministic as the last and first elements and array is already sorted, there will not be much change compared to single pivot quicksort. The same case will happen where all remaining elements will be bigger than the first pivot and all remaining elements will be smaller than the second pivot. Worst case runtime will be O(n*n). For unsorted arrays, whether random or not, dual pivot quicksort will operate similarly to single pivot quicksort with O(nlogn) . The only change in the algorithm is that the algorithm will check if the number to be sorted is in between the values of the first pivot and second pivot, smaller than both or bigger than both. It may be used where single pivot quicksort is used.


Dual Pivot Quicksort
logn
n
n/3
n/3
n/3
n/6
n/6
n/6
→ n
→ n
→ n
O(n) = nlogn