Fatima Rahimova 150180905

11474 – Analysis of Algorithms 1

Homework 1

Fatima Rahimova 150180905

(1) Best-Case:

According to our textbook best-case of quicksort algorithm is the case, when the subarrays of given array should always be balanced, and each subarray should has less elements than $n/2$.

$$T(n) = 2T(n/2) + \theta(n)$$

Since $T(n) = a\,T(n/b) + f(n)$

$a = 2$; $b = 2$; $d = 1$

According to Master Theorem; since $a = b^d$: $2 = 2^1$

$$T(n) = \theta(n^d \log n)$$

$$T(n) = \theta(n^1 \log n) = \theta(n \log n)$$

ANSWER: For Best-Case: $\boxed{T(n) = \theta(n \log n)}$

Fatima Rahimova 150180905

WORST-CASE:

According to our textbook, worst case of quicksort algorithm is the case, when the subarrays of given array in partition are unbalanced, and if one subarray has zero element and other one has n-1 elements.

For n-1 partition: $\theta(n)$

For 0 partition: $\theta(0)$

So; $T(n) = T(n-1) + \cancel{T(0)}^{\to 0} + \theta(n)$

$$T(n) = T(\cancel{n-1}) + \theta(n)$$

$$T(\cancel{n}) = T(\cancel{n-1}) + cn$$

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

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

$$T(\cancel{n-3}) = T(\cancel{n-4}) + c(n-3)$$

$$\vdots$$

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

$$T(n) = T(1) + c.\sum_{k=2}^{n} k$$

ANSWER: $\boxed{T(n) = \theta(n^2)}$ → FOR WORST-CASE OF QUICKSORT ALGORITHM

Fatima Rahimova 150180905

Fatima Rahimova 150180905

② As a first step; let's define indicator random variable:

For $k = 0, 1, \dots, n-1$:

$$X_k = \begin{cases} 1: \text{ if the partition generates a } \underset{\text{split}}{\boxed{k : n-k-1}} \\ 0: \text{ otherwise} \end{cases}$$

$$E[X_k] = P(X_k = 1) = \frac{1}{n}$$

$$T(n) = \begin{cases} T(0) + T(n-1) + \theta(n) & \text{if } 0 : n-1 \text{ split} \\ T(1) + T(n-2) + \theta(n) & \text{if } 1 : n-2 \text{ split} \\ \vdots & \\ T(n-1) + T(0) + \theta(n) & \text{if } n-1 : 0 \text{ split} \end{cases}$$

$$T(n) = \sum_{k=0}^{n-1} X_k (T(k) + T(n-k-1) + \theta(n))$$

Expectations of both sides:

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k (T(k) + T(n-k-1) + \theta(n))\right]$$

$$E[T(n)] = \sum_{k=0}^{n-1} E[X_k]\, E[T(k) + T(n-k-1) + \theta(n)]$$

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

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

Linearity of Expectation: $E[X_k] = \frac{1}{n}$

③

Fatima Rahimova 150180905

$E[T(n)] \leq an\log n$

↳ We should choose a large

Also, there is a fact that:

$$\sum_{k=2}^{n-1} k\lg k \leq \frac{n^2}{2}\lg n - \frac{n^2}{8}$$

By using Substitution Method:

$$E[T(n)] \leq \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n)$$

$$E[T(n)] \leq \frac{2a}{n}\left(\frac{n^2}{2}\lg n - \frac{n^2}{8}\right) + \Theta(n)$$

Desired - Residual:

$$E[T(n)] = an\lg n - \underbrace{\left(\frac{an}{4} - \Theta(n)\right)}_{\hookleftarrow}$$

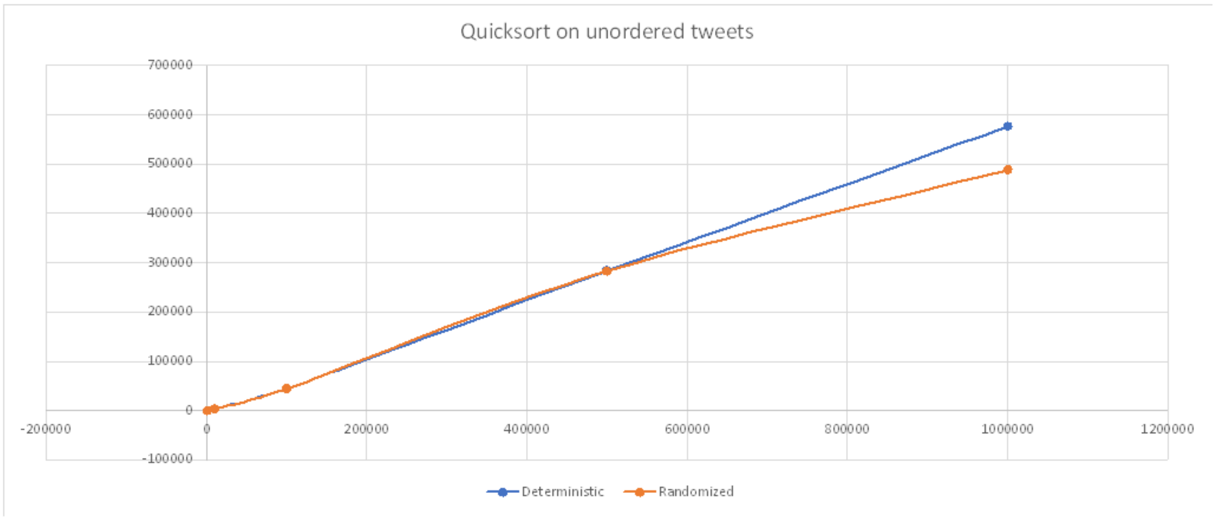
Since we choosed large numbers for a;

$\frac{an}{4}$ should be greater than $\Theta(n)$.

So, $E[T(n)] = an\lg n$

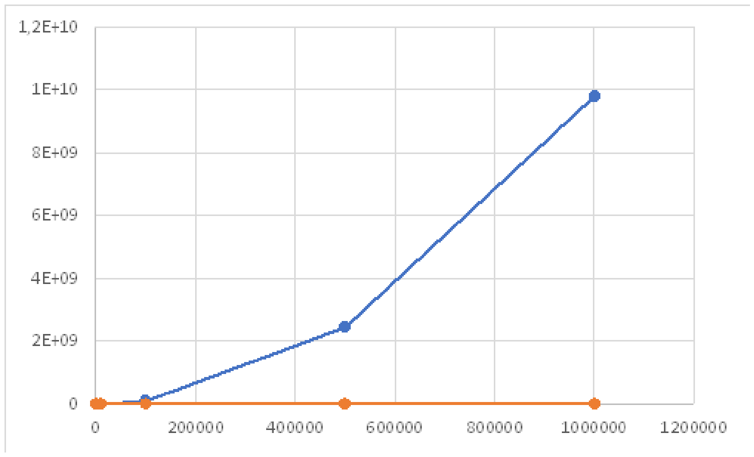
$E[T(n)] = \Theta(n\lg n)$

3)



I execute my both deterministic and randomized Quicksort algorithms for 5 different N values (1000, 10K, 100K, 500K, 1M). As you can see from the given figure in above, average time for deterministic and randomized quicksort algorithms in unordored tweets are too close to each other. Since data is unsorted, deterministic quicksort algorithm works similarly with randomized algorithm in terms of time complexity (all datas is in the randomized order). As you can see deterministic quicksort algorithm is getting more closer (since it works more faster and better than randomized algorithm) to $O(nlogn)$ which is best case asymptotic upper bound of quicksort algorithm.

4)



As you can see from the given figure in above, the randomized algorithm is much more faster in comparison with deterministic quicksort algorithm, since the given data is already sorted. But in the third question, when the given data is unsorted, randomized algorithm was slower than deterministic one. As you remember, in the third question the case was the best case, but in this question since data is already sorted it is worst case of quicksort algorithm, and the asymptotic upper bound for worst case deterministic quicksort algorithm is $O(n^2)$. Also, as I calculated in second question asymptotic upper bound of randomized quicksort algorithm is $O(nlgn)$. As we already know from Lecture 2, nlgn << n^2, we can say that randomized quicksort algorithm works faster and better than deterministic one in worst case scenarios (when given array is already sorted). (Vice versa is also true: Deterministic quicksort algorithm works faster and better than randomized one in best case scenarios).

Summary:

In best case, deterministic quicksort algorithm has better performance compared to randomized quicksort algorithm.

Whereas in worst case, randomized quicksort algorithm has better performance compared to deterministic quicksort algorithm.

Best scenario: Given input array is unsorted

Worst scenario: Given input array is already sorted

Fatima Rahimova 150180905

⑤ In order to find asymptotic upper bound, we should find worst case of dual pivot Quicksort algorithm.

Congruently with normal Quicksort algorithm, worst case is when one subarray has $n-2$ elements, and other 2 subarrays have 0 elements.

For $n-2$ partition: $\Theta(n)$
For 0 partition: $\Theta(0)$

So;

$$T(n) = T(n-2) + \Theta(n) + 2\cancel{\Theta(0)} \rightarrow 0$$
$$T(n) = T(n-2) + \Theta(n)$$
$$T(n) = \cancel{T(n-2)} + cn$$
$$\cancel{T(n-2)} = \cancel{T(n-4)} + c(n-2)$$
$$\cancel{T(n-4)} = \cancel{T(n-6)} + c(n-4)$$
$$\cancel{T(n-6)} = \cancel{T(n-8)} + c(n-6)$$
$$\vdots$$
$$\cancel{T(3)} = T(1) + c.3$$
$$T(n) = T(1) + c.\frac{n^2}{4}$$

ANSWER : $T(n) = \Theta(n^2)$ → Same with normal Quicksort Algorithm but $c\frac{n^2}{4} < c\sum_{k=2}^{n} k$ ← from normal Quicksort