Deterministic and Randomized Quicksort Algorithms Time Complexity Analysis

Tarun Kumar
UE143103
Computer Science Engineering
University Institute of Engineering and Technology
Panjab University
Chandigarh, 160030
Email: tarunk.1996@gmail.com

***Abstract*—This practical analyses the time complexity for sorting (arranging numbers either in ascending or descending order) a list of numbers using two different approaches i.e. Deterministic Quicksort and Randomized Quicksort Algorithms. The time complexities of both the algorithm depends upon the arrangement of numbers in the list.**

I. Introduction

Deterministic (”Common”) quick sort is a technique for sorting a list of numbers all the elements in the given list individually. This is one of the easiest algorithm which serves the purpose. In this paper, Brute force algorithm is used for finding the minimum and maximum valued elements in a given list of numbers. Based on Brute force algorithm, all the elements must be accessed and out of that minimum and maximum valued element are evaluated. Initially, the first element is assigned to minimum and maximum variable. Then each element in the list is compared with minimum variable if the number is smaller than minimum, assign that number to minimum, similarly for maximum if the number is greater than maximum then assign maximum with that number, do this until the list is exhausted completely. At the end, Maximum and minimum of out of that list is present in the variables return maximum and minimum to the main function. This algorithm has a $O(n)$ time complexity.
The other algorithm which is used for finding the minimum and maximum valued element in given set of numbers(list) is Divide and Conquer algorithm. In Divide and Conquer algorithm, The list is divided into several smaller list until a single element is left. Once the list is indivisible, we select the minimum and maximum valued element out of it and as the function is recursively called by passing array, minimum, maximum and size of the array as arguments to the function. Once minimum and maximum is returned to the parent function it evaluates the minimum and maximum obtained from the two function called above and evaluates the minimum and maximum out of it. This process is carried out until the maximum and minimum is not returned to the main function. This algorithm also have a $O(n)$ time complexity.

II. Problem

To analyze the time required to sort a list of numbers using two different approaches of quick sort and discover which one performs better in different cases.(i.e Deterministic quick sort and Randomized quick sort Algorithm).

III. Aim

To write a program which finds the average time required to sort (either in ascending or descending order) a list filled with random numbers using Deterministic quick sort and Randomized quick sort Algorithm. Also, plot a graph demonstrating the total average time required to sort the list of numbers where the size of list increases gradually and compare both the algorithms. Discuss about the factors involved in the nature of the curve observed in the graph.

IV. Approach

The problem at hand requires comparison of time taken by Randomized and Deterministic algorithms and finally predicting which algorithm yields results faster than the other. Yet again, to make observations and derive conclusions easily, graphical analysis is the most comprehensive way. To make the required analysis, time required to sort a given array by using both algorithms can be plotted on a graph. To obtain two curves, each produced by Randomized and Deterministic algorithms, the number of elements in the array can be gradually increased while sorting the array and catalog using the time taken by both the algorithms. The time taken can be plotted against the number of elements in the array to produce the required graph.

V. Experiment

To obtain the data needed for plotting the required graph and deriving conclusions from observations, a C++ program has been implemented. The program, initializes an array of maximum capacity of 500 elements. Although, initially it is populated with only 50 elements. The elements are chosen randomly by the function $(asize * 2)$. Here asize is the current size of the array. For example, initially, array is to be populated with 50 random elements. Hence asize here would be 50 and asize*2 would be 100. So the function would randomly select 50 elements from 0 to 100 and populate the array. The program used, includes two functions, dqpartition and rqpartition for partitioning the array using Deterministic and Randomized algorithms respectively.The function dqpartition chooses the middle element of the array as the pivot while in the function rqpartition, the pivot is randomly chosen from the array. The given unsorted array is sorted using both algorithms and the time required by each algorithm is recorded. However, computers being used today can compute such things in virtually no time, hence to obtain meaningful data, each experiment is performed 10000 times and the average time required to complete the sorting is calculated and recorded. Next, the number of elements is increased by 50 and the same procedure is repeated again. This procedure goes on till the number of elements populating the array becomes equal to 500 which is the maximum size of array and hence ten sets of values are obtained. The data obtained and the graph plotted from the obtained data are shown in Table1 and Graph1 respectively.

The results obtained in the following experiment is performed on a Ubuntu - Linux(x64 bit) based operating system. The system consist of 4GB of RAM and have a Intel(R) Core(TM) i3-2330M processor with a clock speed of 2.20GHz. To minimize the hindrance, heavy background process were terminated before running the code.

VI. Algorithms for Sorting List of numbers using Deterministic and Randomized Quick sort Technique

Algorithm 1 Deterministic Quicksort

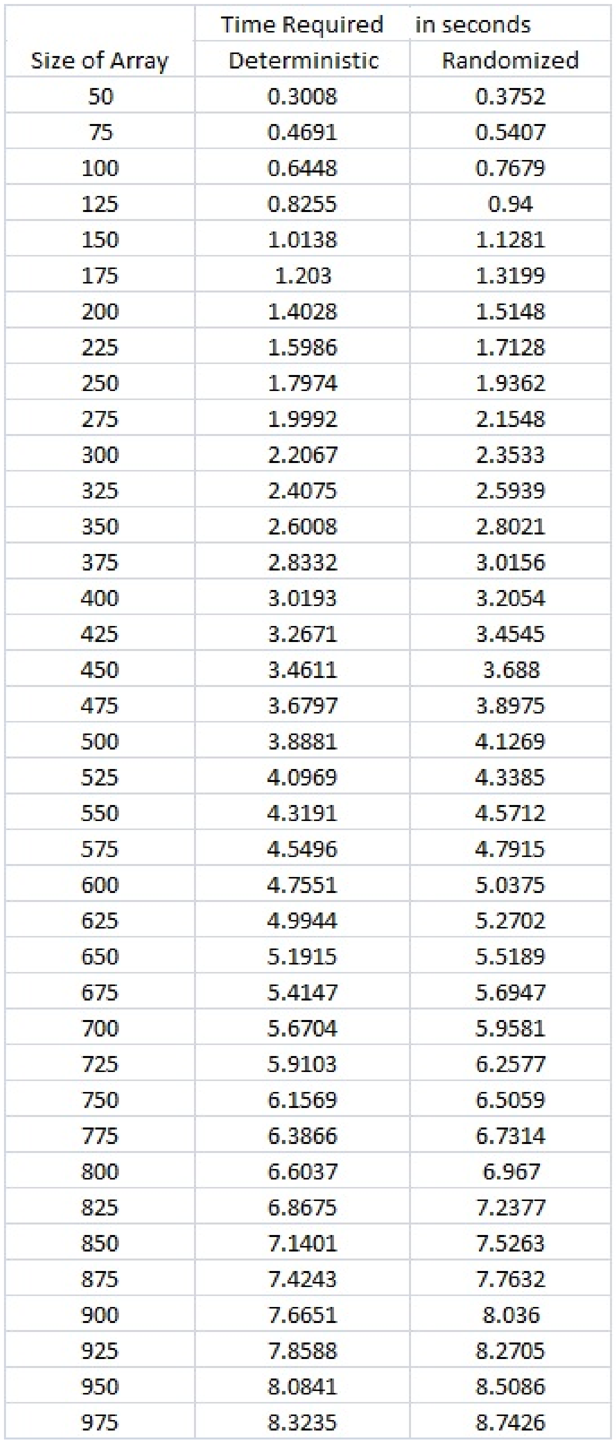
```
quicksort(a,low,high)
if $low \leq high$ then
    $r \leftarrow partition(a, low, high)$
    $quicksort(a, low, r-1)$
    $quicksort(a, r+1, high)$
end if
partition(a,low,high)
$value \leftarrow a[low]$
$left \leftarrow low$
$right \leftarrow high$
$a[high+1] \leftarrow infinity$
while $left < right$ do
    while $a[left] \leq value$ do
        $left++$
    end while
    while $a[right] > value$ do
        $right--$
    end while
    if $left < right$ then
        $swap(a[left], a[right])$
    end if
end while
if $left > right$ then
    $a[low] \leftarrow a[right]$
    $a[high] \leftarrow value$
end if
$return$
```

Algorithm 2 Randomized Quicksort

```
Randquicksort(a,low,high)
if $low \leq high$ then
    $r \leftarrow Randpartition(a, low, high)$
    $Randquicksort(a, low, r-1)$
    $Randquicksort(a, r+1, high)$
end if
Randpartition(a,low,high)
$ind \leftarrow (Randomlygeneratedindexfromlowtohigh)$
$swap(a[low], a[ind])$
$partition(a, low, high)$
```

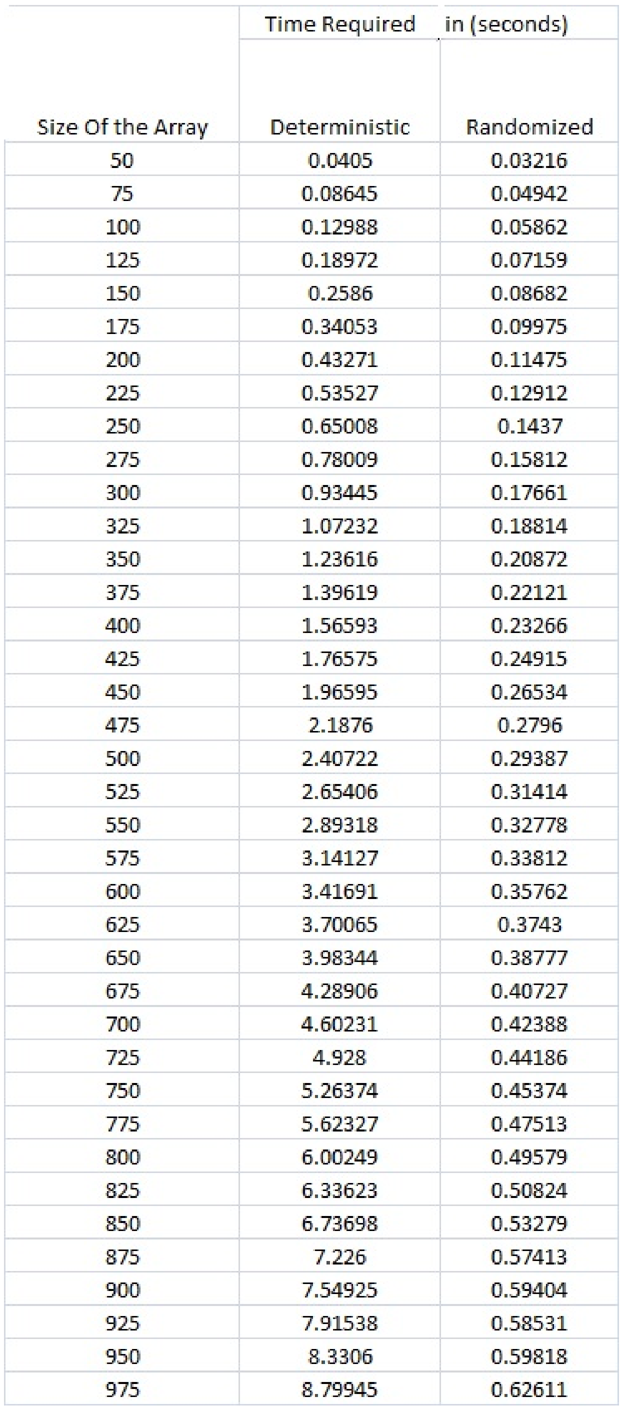
VII. Results and Graphs

Following are the graphs and tabled obtained after performing the experiment as specified above :



	Time Required	in seconds
Size of Array	Deterministic	Randomized
50	0.3008	0.3752
75	0.4691	0.5407
100	0.6448	0.7679
125	0.8255	0.94
150	1.0138	1.1281
175	1.203	1.3199
200	1.4028	1.5148
225	1.5986	1.7128
250	1.7974	1.9362
275	1.9992	2.1548
300	2.2067	2.3533
325	2.4075	2.5939
350	2.6008	2.8021
375	2.8332	3.0156
400	3.0193	3.2054
425	3.2671	3.4545
450	3.4611	3.688
475	3.6797	3.8975
500	3.8881	4.1269
525	4.0969	4.3385
550	4.3191	4.5712
575	4.5496	4.7915
600	4.7551	5.0375
625	4.9944	5.2702
650	5.1915	5.5189
675	5.4147	5.6947
700	5.6704	5.9581
725	5.9103	6.2577
750	6.1569	6.5059
775	6.3866	6.7314
800	6.6037	6.967
825	6.8675	7.2377
850	7.1401	7.5263
875	7.4243	7.7632
900	7.6651	8.036
925	7.8588	8.2705
950	8.0841	8.5086
975	8.3235	8.7426

Fig. 1. Time required to sort a given list random numbers using both Deterministic and Randomized Algorithm



	Time Required	in (seconds)
Size Of the Array	Deterministic	Randomized
50	0.0405	0.03216
75	0.08645	0.04942
100	0.12988	0.05862
125	0.18972	0.07159
150	0.2586	0.08682
175	0.34053	0.09975
200	0.43271	0.11475
225	0.53527	0.12912
250	0.65008	0.1437
275	0.78009	0.15812
300	0.93445	0.17661
325	1.07232	0.18814
350	1.23616	0.20872
375	1.39619	0.22121
400	1.56593	0.23266
425	1.76575	0.24915
450	1.96595	0.26534
475	2.1876	0.2796
500	2.40722	0.29387
525	2.65406	0.31414
550	2.89318	0.32778
575	3.14127	0.33812
600	3.41691	0.35762
625	3.70065	0.3743
650	3.98344	0.38777
675	4.28906	0.40727
700	4.60231	0.42388
725	4.928	0.44186
750	5.26374	0.45374
775	5.62327	0.47513
800	6.00249	0.49579
825	6.33623	0.50824
850	6.73698	0.53279
875	7.226	0.57413
900	7.54925	0.59404
925	7.91538	0.58531
950	8.3306	0.59818
975	8.79945	0.62611

Fig. 2. Time required to sort a given list of sorted numbers using both Deterministic and Randomized Algorithm

VIII. Discussion on the results

A. *In case of List filled with random numbers*

In figure 1, It shows the time required to sort a given list random numbers using both Deterministic and Randomized Algorithm.
Also, from fig 3. which is obtained from the results

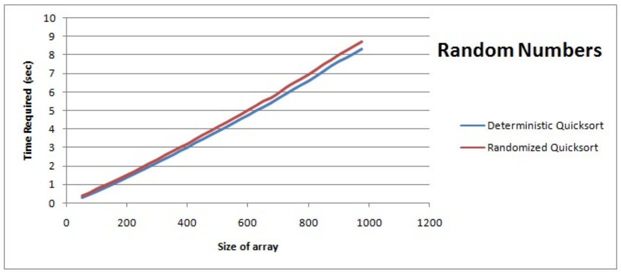


Fig. 3. Graph Obtained from the values listed on fig 1

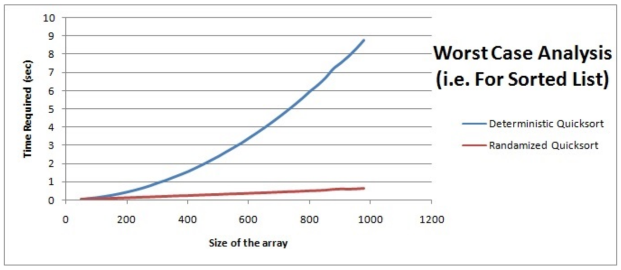


Fig. 4. Graph Obtained from the values listed on fig 2

of table 1, It shows that, in case of random numbers or jumbled numbers the time required to sort the given list of numbers is almost same for both the cases. Though Randomized quick sort is little slower compared to Deterministic quick sort as the curve is growing faster for randomized quick sort.
The anomaly observed in the curved must be because of the call of $rand()$ function. Which requires some time to return a value back to the pivot. This function is not called in case of deterministic quick sort as pivot is always the first element from the list. Thus, a growth is observed in the randomized quick sort algorithm.
The average time complexity of both randomized and deterministic quick sort curves in case of list filled with random numbers is $nlog(n)$. And the graph also proves that the results obtained from the program are similar to the expectation.

B. *In case of List filled with sorted numbers*

In figure 2, It shows the time required to sort a given list sorted(either in ascending or descending order) numbers using both Deterministic and Randomized Algorithm.
Also, from fig 4. which is obtained from the results of table 2, It shows that, in case of sorted numbers the time required to sort the given list of number using both the approaches i.e deterministic and randomized quick sort is quite variant. The time complexity of randomized quick sort is way more faster than that of deterministic quick sort.
The anomaly observed in the because the time complexity of the Randomized quick sort is $pow(n, 2)$ while for Deterministic it is still $nlog(n)$. Because of that, the difference in the nature of both the curve is clearly visible in the graph 4.

IX. Conclusion

- From the analysis conducted of the data and graphs obtained from the experiments, it can be easily concluded that Deterministic Quicksort algorithm on an average case produces results faster than Randomized Algorithm.
- But in the case of worst case analysis (where the array is in sorted order either in decreasing or increasing order), Randomized Quicksort is way more faster compared to Deterministic Quicksort.

X. Attachments

The Program code used for performing the above experiment is uploaded here : http://ideone.com/tqNddq