SORTING ALGORITHMS REPORT

Comparison between merge sort and quick sort

Name:Swarnali Saha

ID:1805104

CSE'18-B, L-2/T-1

Objective:

In this experiment, we will analyze the running time complexity of both quicksort and merge sort algorithm. These are both sorting algorithm and they use divide and conquer rule for solving the problems. Here, we will use array of random elements, array of ascending elements and descending elements for completing the experiment.

Complexity analysis:

Merge Sort:

It is a recursive algorithm. In this process, array is divided into two equal parts, sort the elements recursively and merge the sub-arrays. It doesn't depend on any elements. Rather, in all three cases(best, worst and average), it will dived the array into the length of two n/2 arrays, if the main array has n elements. Time complexity of this algorithm can be expressed as following recurrence relation.

T(n) $= 2T\left(\frac{n}{2}\right) + \theta(n)$

$= 4T\left(\frac{n}{4}\right) + 2\theta\left(\frac{n}{2}\right) + \theta(\mathrm{n})$

$= 4T\left(\frac{n}{4}\right) + 2\theta(\mathrm{n})$

$= 8T\left(\frac{n}{8}\right) + 3\theta(\mathrm{n})$

$= 2^3T\left(\frac{n}{2^3}\right) + 3\theta(\mathrm{n})$

…………………………………….

…………………………………….

$= 2^kT\left(\frac{n}{2^k}\right) + k\theta(\mathrm{n})$

$= 2^{\log_2 n}T\left(\frac{n}{2^{\log_2 n}}\right) + \log_2 n\,\theta(\mathrm{n})$ [for base case T(1)]

$= nT(1) + n\log n$ [θ(n) is substituted by cn and c is omitted]

$= n\log n$

$so, the\ time\ complexity\ of\ merge\ sort\ is\ O(n\log n).$

<u>Quick Sort:</u>

Quick sort doesn't divide the array into two equal parts always. It depends on the number of elements. So the time complexity depends on the balanced and imbalanced partitions.

Worst case scenario:

It occurs when the array is divided into two parts- one with 0th element and other with n-1 elements.

The cost for the partitioning is,

$$
\begin{aligned}
\text{T(n)} &= T(0) + T(n-1) + \theta(\mathrm{n}) \\
&= T(n-1) + \mathrm{cn} \\
&= T(n-2) + c(n-1) + \mathrm{cn} \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots. \\
&= T(1) + c + 2c + 3c + \ \ldots\ldots\ldots\ + (n-1)c + cn \\
&= c\frac{n(n-1)}{2} \\
&= O(n^2)
\end{aligned}
$$

So, the complexity of quicksort in worst case scenario is $O(n^2)$. And in this experiment ascending and descending array elements can be considered into this category.

Best case scenario:

It occurs when the array is divided into two parts- one with n/2 elements and other with (n/2-1) elements. The cost for best case is,

$$
\begin{aligned}
\text{T(n)} &= 2T\left(\frac{n}{2}\right) + \theta(n) \\
&= n\log n
\end{aligned}
$$

$so, the\ time\ complexity\ of quick\ sort\ in\ best\ case\ scenario\ \ is\ O(n\log n).$

Average case scenario:

T(n) = $T(\frac{n}{p1}) + T(1 - \frac{n}{p1}) + cn$ [partitions of size n/p1 and (1- n/p1)]

The recursion will reach its base case in logp1n and log(1- n/p1) levels and all levels have at most cn cost. So, considering log(p1/n) > log(1- n/p1)n, the runtime would be:

T (n) ≤ c n log (p1/n)

≈ C n log n

Therefore , Time complexity of Quick sort in random array = O(n log n) and here random array elements sorting is considered in this category.

Machine configuration:

Processor: Intel(R) Core(TM) i7-7500U@2.70GHz
RAM: 8.0 GB

Table:

	Time in micro seconds						
Input order	**n=** / **Sorting algorithm**	**10**	**100**	**1000**	**10000**	**100000**	**1000000**
Ascending	Merge	6.5	27.4	149.7	1891.6	5028.9	35833.33
	Quick	11.3	128.1	1935.4	15550.6	1038299.8	130030825.7
Descending	Merge	14.4	29.4	347.3	1020.6	2897.8	34130.33
	Quick	24.7	61.0	1000	19786.2	1241917.4	220599603.3
Random	Merge	16.21	29.1	53.6	523.8	4104.4	47126.6
	Quick	22.4	93.1	297.8	1780.6	9507.2	121338

Graph Plotting:

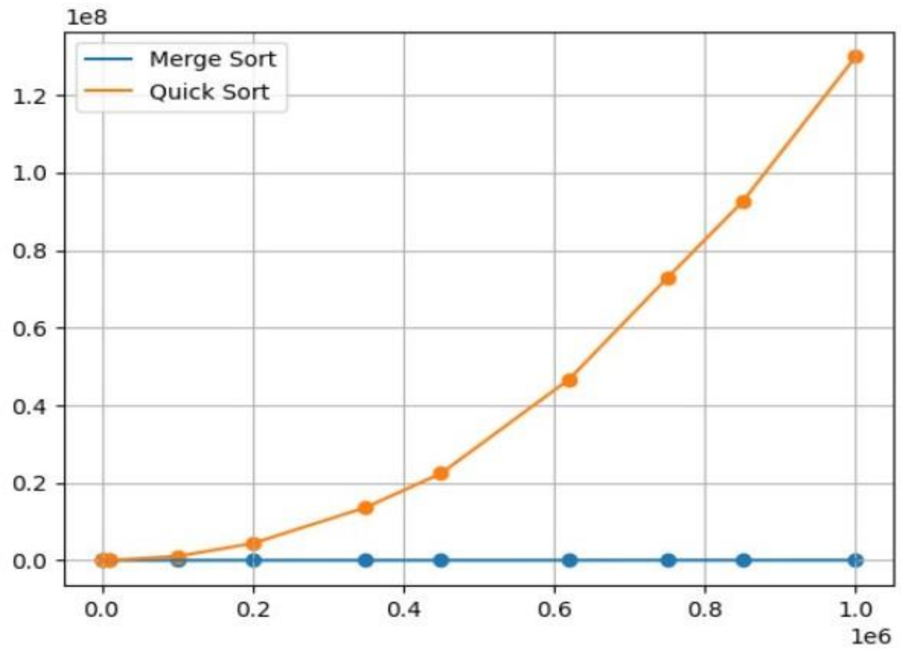

Fig-1:Comparison of two algorithms in ascending order input

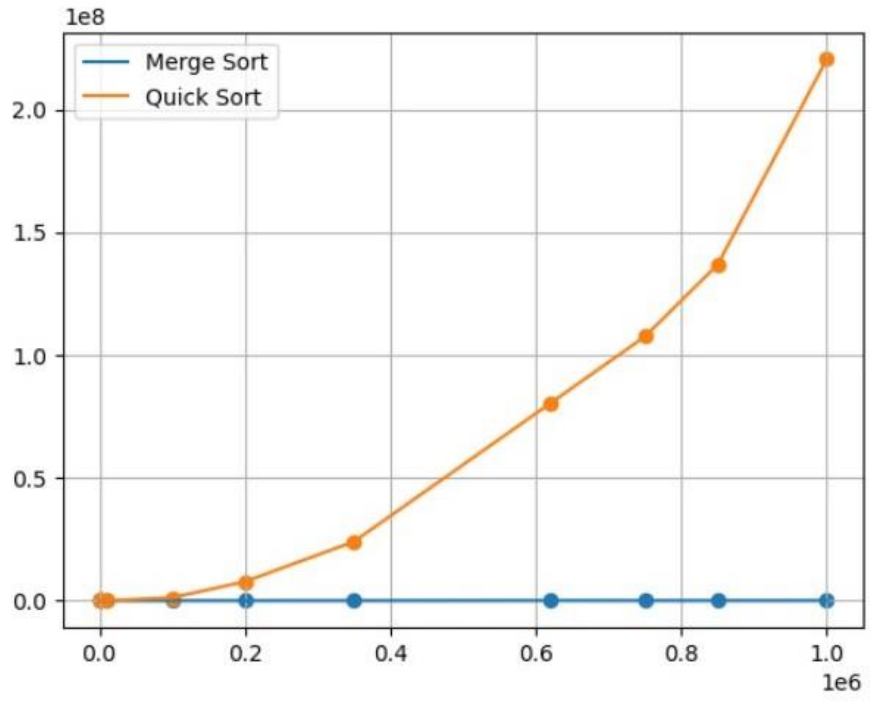

Fig-2: Comparison of two algorithms in descending order input

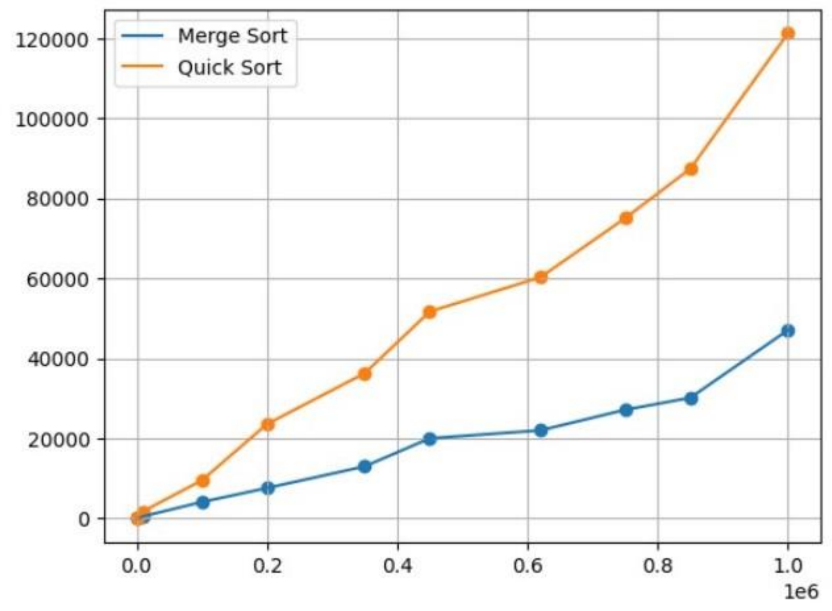

Fig-3: Comparison of two algorithms in random order input

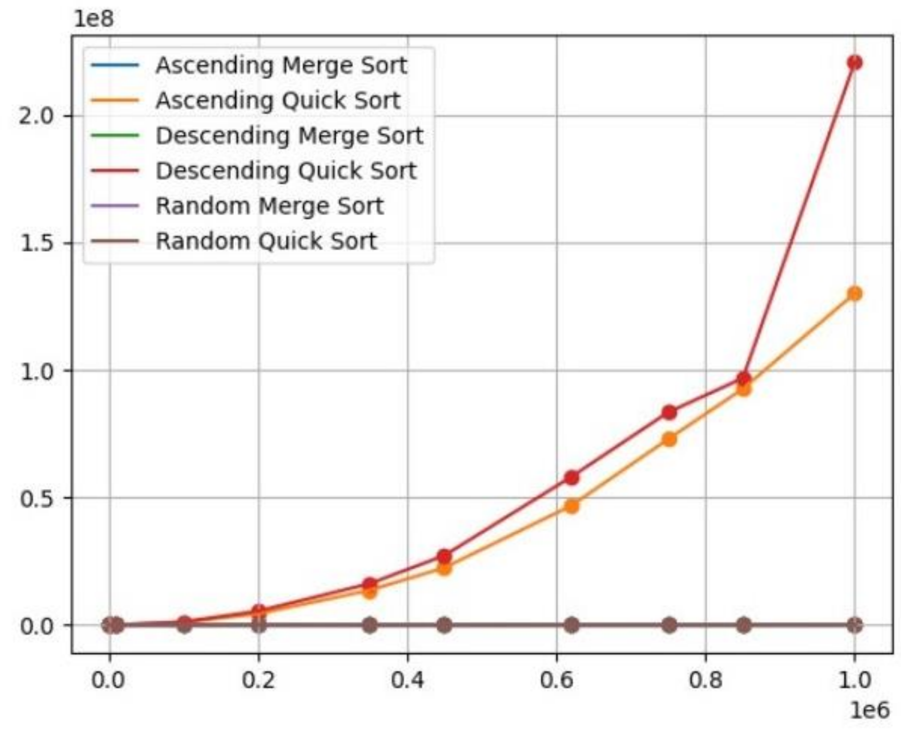

Fig-4: Comparison of two algorithms in ascending, descending and random order input

Discussion:

Here, for all three orders merge sort requires less time than quick sort. And, array of random elements takes less time in sorting than ascending and descending order. But there are also exceptions. For n=10^5 , merge sort of random array has taken more time than ascending and descending order arrays. It can be also seen from the graph that if we increase the length of the array, it will require more time than smaller array.