CSE 204

Data Structure & Algorithm I Sessional

A Report on Merge Sort v/s Quick Sort

Submitted By

Name: Md. Nazmul Islam Ananto

ID: 1805093

Date of Submission: 2021-06-11

Introduction

Merge Sort and Quick Sort both are some sorting algorithms with similar time complexity and the “Divide and Conquer” approach.

In Merge Sort, we divide the array in half every time until we are left with arrays of only one element and then we merge those arrays recursively and sort them accordingly. In Quick Sort, we take an element in the array as pivot (usually the last element) and then divide the rest into three portions as they are less, equal and greater than our pivot. We place the pivot in the perfect position and recursively so the same with our less, equal and greater portions.

Data Table

Input Order	Sorting Algorithm	Time in microseconds					
		n=10	100	1000	10000	100000	1000000
Ascending	Merge	20.25	52.95	312.95	2063.85	15766.45	166990.8
	Quick	34.43	208.05	1495.73	54969.75	3913774.0	388819802
Descending	Merge	20.74	82.04	246.07	2417.45	18173.7	155815.2
	Quick	5.78	111.39	1427.34	61919.75	3860951.2	399606938.4
Random	Merge	15.37	141.94	811.2	6266.05	37815.06	289328.94
	Quick	5.21	42.28	229.11	2388.7	23390.58	179852.36

Graphs & Complexity Analysis

Merge Sort

The division of the array is always done through the middle, that is the array is halved, so the resultant array has a length of n/2 if the original array has a length of n.

Now,

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

$= 4T(n/4) + 2\,\Theta(n/2) + \Theta(n)$

$= 4T(n/4) + 2\ \Theta(n)$

$= 8T(n/8) + 3\ \Theta(n)$

$\approx 2\log n\ T(n/n) + \log n\ \Theta(n)$

$\approx nT(1) + n\log n$ [$\Theta(n)$ is n]

$\approx n + n\log n$

$\approx n\log n$

Therefore, time complexity of merge sort = O(nlog)

Quick Sort

Here the pivot was taken as the last element, and in that case, If the array is in ascending or descending order,

$T(n) = T(0) + T(n-1) + Cn$

$= T(n-1) + Cn$

$= T(n-2) + C(n-1) + Cn$ ... ...

$= T(1) + C + 2C + 3C + ... + (n-1)C + nC$

$\approx Cn(n+1)/2$

Therefore, time complexity of quicksort in ascending or descending order = O(n^2)

If the array is initially in random order,

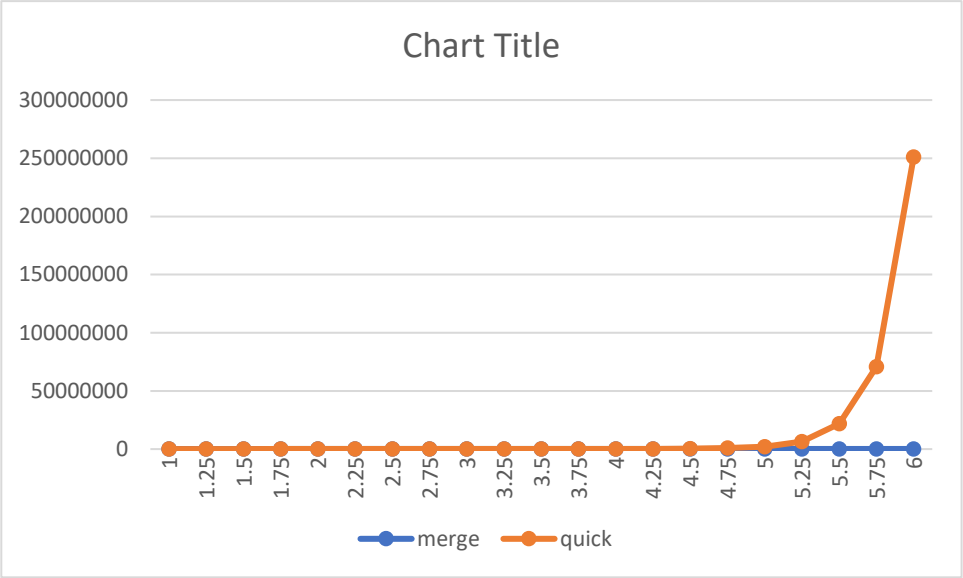
$T(n) = T(n/p) + T(n/(n-p)) + Cn$ [Let, the pivot index be p]

$\leq Cn\log_p n$

$\approx Cn\log n$

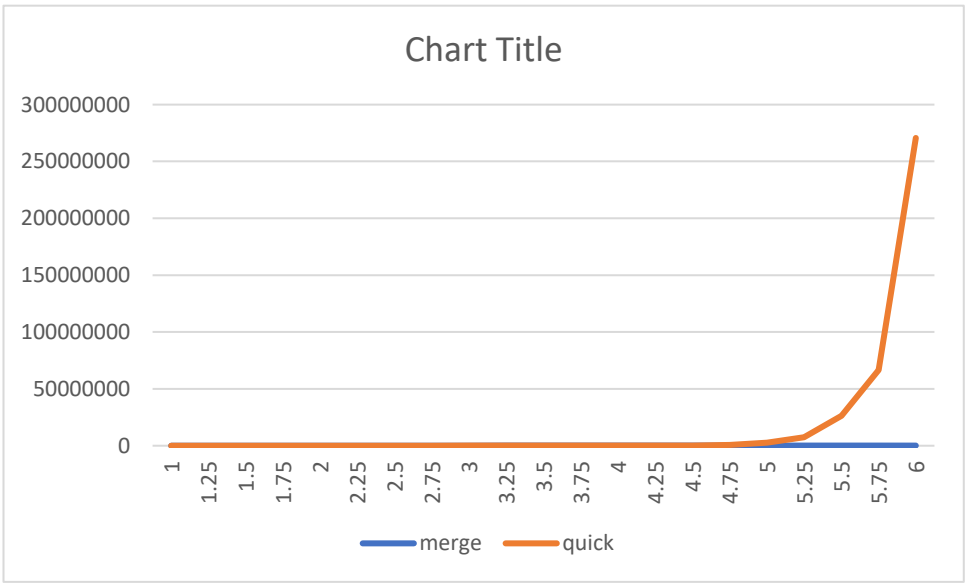
Therefore, time complexity of quicksort in random order = O(nlogn)

Ascending Order



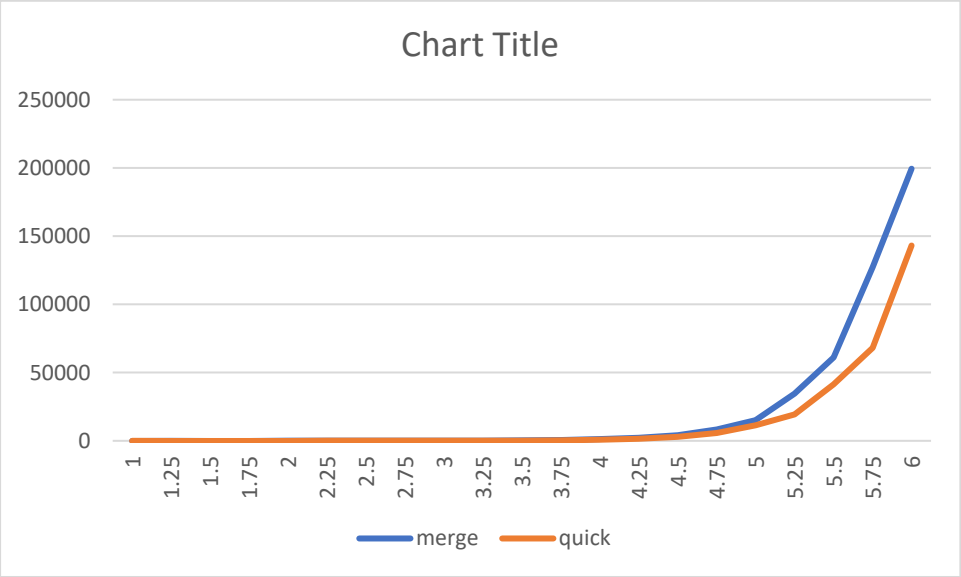
Here we can see that the quick sort takes significantly more time than the merge sort. This is because quick sort hits it worst case when the input array is in ascending order and in that case, the time complexity of the quick sort is n^2 where n is number of integers.

Descending Order



In the Descending order, merge sort still performs better than quick sort as the quick sort hits it worst case again in this scenario. If we took dynamic or median pivot then may be quick sort would have performed a bit better than now.

Random Order



We can see that in random order, our quick sort outperforms merge sort which is new. Actually, quick sort is better when the number of inputs is low or random. But as it gets higher or in some sort of ascending/descending order, quick sort starts to get significantly slower.

Machine Configuration

Processor: *Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz*

Ram: *8.00GB Transcend*

Storage: *256GB Transcend SSD Nvme*

OS: *64bit Windows 10 21H1*