Bangladesh University of Engineering and Technology

Course No: CSE 204

Course Title: Data Structure & Algorithm - I

Offline Assignment – 7

MERGE SORT VS QUICK SORT

Name: K.M. Fahim Shahriyar

Student Id: 1805113

Department : CSE

Section: B2

Submission Date : 11/06/2021

<u>DATA TABLE</u>

Average time (in ms) for sorting n integers in different input orders

Input Order	n = / Sorting Algorithm	10	100	1000	10000	100000	1000000
Ascending	Merge	0	0	0.75	1.9	12.86	97.8
	Quick	0	1	6.125	53.2	3921.86	479392.8
Descending	Merge	0	0	1	1.85	14.57	95.2
	Quick	0	0	6.6	43.3	3876.14	714507.5
Random	Merge	0	0	0.7	2.87	15.17	178.2
	Quick	0	0	0.9	3.25	14.17	82.9

Graph Plotting

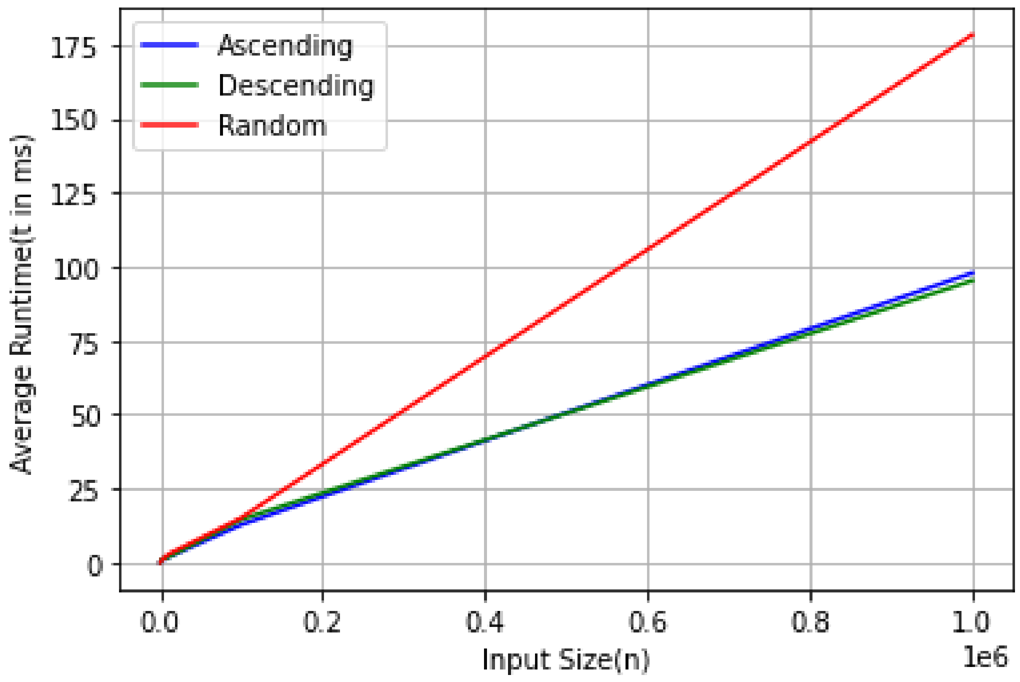

Fig : Merge sort analysis

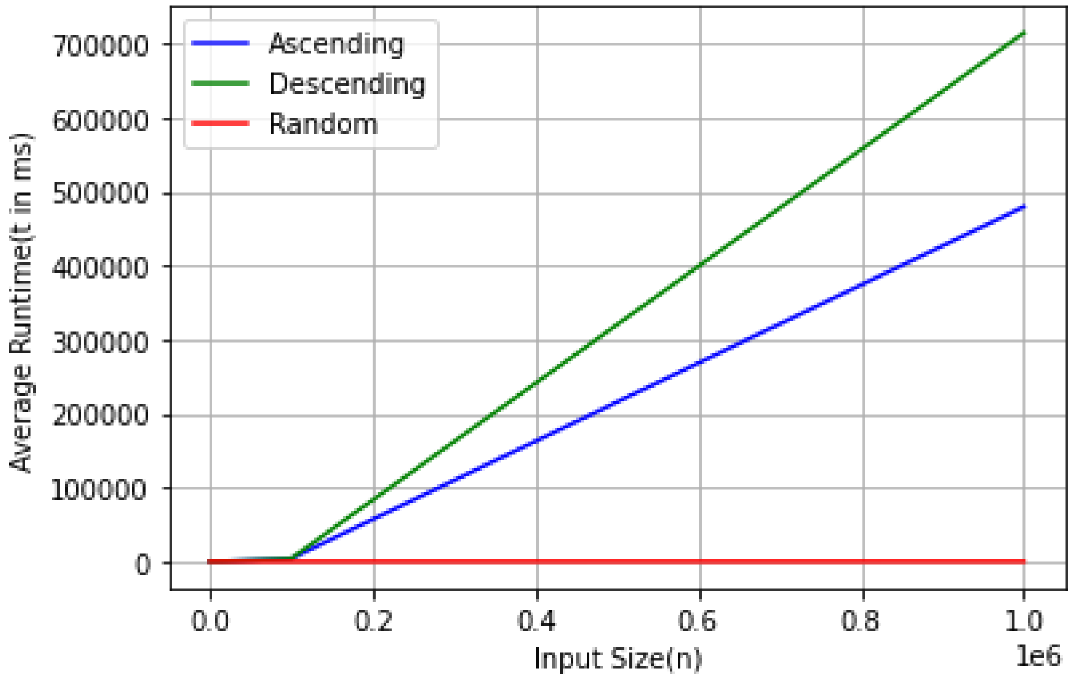

Fig:Quick Sort Analysis

<u>Machine Configuration:</u>

Processor : Intel core-i5 8th gen (base speed : 1.80 GHz)

Ram : 8 gb

SSD : 256 GB

OS: Windows 10 genuine

<u>Complexity Analysis</u>

<u>Merge Sort:</u>

<u>For Random Order:</u>

From the asymptotic analytic perception,we consider large input size such as n=1000,10000 ,100000,1000000 and the average running times for random ordered array for those input size are 0.7 ms, 2.87 ms, 15.17 ms and 178.3 ms respectively.

We can observe that, from n=1000 ,for increasing the input size 10 times each time the average run time is increasing 4.1 times, 5.28 times ,11.75 times respectively.

<u>For Ascending Order:</u>

From the asymptotic analytic perception,we consider large input size such as n=1000,10000 ,100000,1000000 and the average running times for those input size for ascending ordered array are 0.75 ms, 1.9 ms, 12.86 ms and 97.8 ms respectively.

We can observe that, from n=1000 ,for increasing the input size 10 times each time the average run time is increasing 2.5 times, 6.7 times and 7.6 times respectively.If we continue the observation for more two times increasing the input size 10 times,then the average time increases upto 10.5 times.

For Descending Order:

From the asymptotic analytic perception,we consider large input size such as n=1000,10000 ,100000,1000000 and the average running times for descending order for those input size are 1 ms, 1.85 ms, 14.57 ms and 95.2 ms respectively.

We can observe that, from n=1000 ,for increasing the input size 10 times each time the average run time is increasing 1.85 times, 7.9 times ,6.53 times respectively.If we continue the observation for more two times increasing the input size 10 times for each,then the average time increases upto around 11 times.

Complexity for all three order:

So,analyzing all the three order for merge sort,we can observe that each of the three cases giving almost same amount of time asymptotically.That's why from the above plotting of average runtime vs input size ,it is showing the almost same growth rate at each curve for all the three orders.

We can clearly observe that when input size is very large (n >= 1000000) ,increasing the input size for 10 times ,showing the average running time increasing more than 10 times but not increasing polynomially.So it is pretty obvious that average running time is not increasing linearly with input size.So,the run time is in between linear growth rate and polynomial growth rate.So the runtime can bounded by g(n) = nlogn asymptotically.

So,the complexity of merge sort for all of these three orders is asymptotically,

T(n) = O(nlogn)

Quick Sort:

For Random Order:

From the asymptotic analytic perception,we consider large input size such as n=1000,10000 ,100000,1000000 and the average running times for random ordered array for those input size are 0.9 ms, 3.25 ms, 14.17 ms and 82.9 ms respectively.

We can observe that, from n=1000 ,for increasing the input size 10 times each time the average run time is increasing 3.61 times, 4.36 times ,5.85 times respectively. If we continue the observation for more two times increasing the input size 10 times for each,then the average time increases upto around 11.11 times.

Complexity for Random order:

So,like merge sort here also,We can clearly observe that when input size is very large (n >= 1000000) ,increasing the input size for 10 times ,showing the average running time increasing more than 10 times but not increasing polynomially.So it is pretty obvious that average running time is not increasing linearly with input size.So,the run time is in between linear growth rate and polynomial growth rate.So the runtime can bounded by g(n) = nlogn asymptotically

So,the complexity of quick sort for random order is asymptotically,

T(n) = O(nlogn)

For Ascending Order:

From the asymptotic analytic perception,we consider large input size such as n=1000,10000 ,100000,1000000 and the average running times for those input size for ascending ordered array are 6.125 ms, 53.2 ms, 3921.86 ms and 479392.8 ms respectively.

We can observe that, from n=1000 ,for increasing the input size 10 times each time the average run time is increasing 8.68 times, 73.7 times and 122.23 times respectively.

Complexity for Ascending order:

So,from the increasing rate of running time with the input size,we can observe that when input size is very large ,the running time is increasing at the rate of almost square of the increment of input size.So,the running time is increasing polynomially.But the growth rate of running time is almost any constant multiplied by two degree polynomial with respect to input size.That’s why the running time in this case can be bounded by g(n)=k*n^2 (k=any constant).

So,for ascending order, the complexity of quick sort is asymptotically,

T(n) = O(n^2)

For Descending Order:

From the asymptotic analytic perception,we consider large input size such as n=1000,10000 ,100000,1000000 and the average running times for those input size for descending ordered array are 6.6 ms, 43.3 ms, 3876.14 ms and 714502.8 ms respectively.

We can observe that, from n=1000 ,for increasing the input size 10 times each time the average run time is increasing 6.56 times, 89.51 times and 184.33 times respectively.

Complexity for Descending order:

So,like the ascending order,from the increasing rate of running time with the input size,we can observe that when input size is very large ,the running time is increasing at the rate of almost square of the increment of input size.So,the running time is increasing polynomially.But the growth rate of running time is almost any constant multiplied by two degree polynomial with respect to input size.That’s why the running time in this case can be bounded by $g(n)=k*n^2$ (k=any constant).

So,also for descending order, the complexity of quick sort is asymptotically,

$T(n) = O(n^2)$

Conclusion

So,Analyzing both merge sort and quick sort for all of three orders we can see that asymptotically ,merge sort produces same complexity for any kind of input order and the complexity is T(n) = O(nlogn)

For quick sort we can observe,that it is very efficient for randomly ordered input and the complexity in ths case is T(n)=O(nlogn). But when the input is ascending or descending order ,the complexity becomes polynomial and producecs poor performance comparing with merge sort.For both descending and ascending order,the complexity of quick sort is T(n)=O(n^2)