

Que 17

Merge Sort comparison .

5	3	6	7	9	8	3	1	2
---	---	---	---	---	---	---	---	---

5	3	6	7	9	8	3	1	2
---	---	---	---	---	---	---	---	---

(1)	(6)	(10)
[5 3]	[6 7]	[9 8 3] [1 2]
(2) (5) (7) (8)	(11) (13) (16) (17)	
[5] [3] [6] [7]	[9 8] [3] [1] [2]	
(3)	(12) (14)	
[5] [3] [6] [7]	[9] [8] [3] [1] [2]	
(4) (9)	(15) (18)	
[3 5] [6 7]	[3 8 9] [1 2]	
(19)	(20)	
[3 5 6 7]	[1 2 3 8 9]	
(21)		
[1 2 3 3 5 6 7 8 9]		

Merge sort => divide into 3=>

5	3	6	7	9	8	3	1	2
---	---	---	---	---	---	---	---	---

(1)	(10)	(20)
[5 3 6]	[7 9 8]	[3 1 2]
(2) (3) (11) (12) (21) (22)		
[5 3] [6] [7 9] [8] [3 1] [2]		
(4) (5) (6) (13) (14) (15) (23) (24) (25)		
[5] [3] [6] [7] [9] [8] [3] [1] [2]		

(7)	(8)	(16)	(17)	(26)	(27)
[3 5]	[6]	[7 9]	[8]	[1 3]	[2]
(9)	(18)	(28)			
[3 5 6]	[7 8 9]	[1 2 3]			
(19)		(29)			
[3 5 6 7 8 9]		[1 2 3]			
(30)					
[1 2 3 3 5 6 7 8 9]					

Time Complexity Analysis=>

2-way Merge sort we get the equation: $T(n) = 2T(n/2) + O(n)$

Similarly, in case of 3-way Merge sort we get the equation: $T(n) = 3T(n/3) + O(n)$

Master Theorem=>

For 2 way merge sort, $a=2$, $b=2$, $c=\log(\text{to the base } b)a$, $c=1$, $k=0$

Run time complexity=> $O(n \log n)$

For 3 way merge sort, $a=3$, $b=3$, $c=\log(\text{to the base } b)a$

Run time complexity is => **$O(n \log_3 n)$**

By solving it using Master Theorem, we get its complexity as **$O(n \log_3 n)$** ..

Although time complexity looks less compared to 2 way merge sort, the time taken actually may become higher because number of comparisons in merge function go higher

Conclusion:

- **Are the two approaches tied?**
- **=>No**

- **If one of the approaches is better, which one is better? why?**

=>I think , 2 way merge sort is better. Although time complexity looks less compared to 2 way merge sort, the time taken actually may become higher for 3 way merge sort because number of comparisons in merge function go higher.