☆ Sorting Techniques-

Bubble Sort :-

Most Basic Sorting algo! compare each element in the array and check if arr[j] > arr[j+1] if so swap them! eventually after each iteration the biggest element will reach to the end now iterate again but this time n-1 time than n-2 time, then 3 and so on....

Here's the DryRun:- (will do later) but we know we have to ~~use~~ iterate it for (n times) and for (n-i) times if an array is already sorted we can check if any swapping is made or not in first iteration. we can use a bool check in loop (first loop) to see if any swapping is required in next loop if any swap happens we mark it as false and will check again!

Dry Run:- For Input ~~7,9,1, 2, 3, 5,~~

5	4	3	2	1

swap swap.

After 1st iteration:

5>4. so swap 5 with 4

4	5	3	2	1

S: (5>3) so, | 4, | 3, | 5, | 2 | 1 |

(5>2) | 4 | 3 | 2 | 5 | 1 |

| 5 > 1 | | 4 | 3 | 2 | 1 | 5 | → Hence last element is placed will consider → this part only.

→ | 3 | 2 | 1 | 4 | 5 |

| 2 | 1 | 3 | 4 | 5 |

→ | 1 | 2 | 3 | 4 | 5 | Hence Sorted !

Optimised Bubble sort will solve.

1 2 3 4 5 like:- No swapping made Array is already sorted.

5, 1, 2, 3, 4 :- It will solve it in one iteration only.

| 1 | 2 | 3 | 4 | 5 |

and in next iteration no swapping is made. Hence it'll lower it's time Complexity.

Selection .Sort:-

→ Find the largest element!
→ swap it with end inden element.
→ Decrement end inden by 1.
→ keep on doing it Boom!
It works

you can optimize it as well.

Insertion Sort.

→ Think that first element is sorted already.
→ Now take current element = arr[i].
→ and an index j that will point to element just before the current element.

```
Now  while (arr [j] > current && j>=0)
        { arr [j+1] = arr [j];
                j--;
        }
          arr [j+1] = current;
       }.
```

let's Dry run on Example : 5 4 3 6 9

| 5 | 4 | 3 | 6 | 9 |

↓j ↓i   Current = 4.

is j > current? yes

| 5 | 5 | 3 | 6 | 9 | (arr[j+1] = arr[j]). Sunday 01

j<0 loop ends →

| 4 | 5 | 3 | 6 | 9 | (arr[j+1] = current).

↓j ↓i

j > current? (yes)

| 4 | 5 | 5 | 6 | 9 |

↓j +1 ↓i   (arr[j] > current)? Yes (5 4>3)

loop ends :-

| 4 | |

Dry Run on Example:-

| 5 | 4 | 3 | 6 | 9 |

j i

arr[j] > Current ? Yes :→ arr[j+1] = arr[j]

| 5 | 5 | 3 | 6 | 9 |

j i

j < 0 ; arr[j+1] = Current

| 4 | 5 | 3 | 6 | 9 |

j i

current=3

5 > 3 yes :→ arr[j+1] = arr[j]

| 4 | 5 | 5 | 6 | 9 |

j i

4 > 3 yes :- arr[j+1] = arr[j]

| 4 | 4 | 5 | 6 9 |

j < 0 :- loop out

arr[j+1] = Current

| 3 | 4 | 5 | 6 | 9 |

j i

5 > 6 no :- | 3 | 4 | 5 | 6 | 9 |

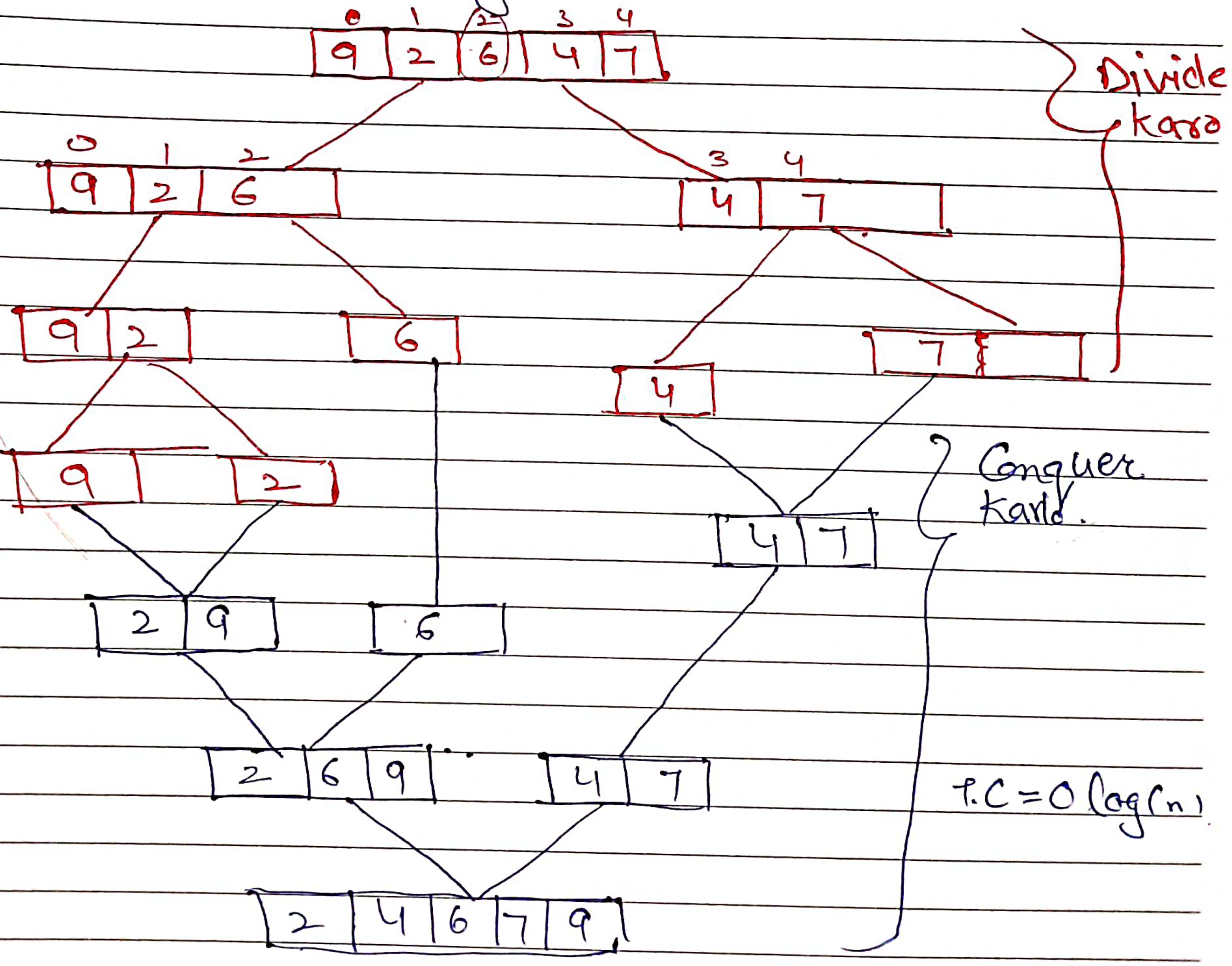
j i

Mubarakh ho Array Sort Hogaya !
Ab zindagi ka bhi kuch algo dundho;

Merge Sort :-

Divide and Conquer (Wohi Angrezo wala).

→ Divide the Arrays into single Element block (Ab single hai to sort hi hoga ~ Lord Abhi). Now merge it back (like you merge two sorted Arrays). and boom Aapka Apna Array sort. Ab zindagi ka bhi sochlo.



$T.C = O\log(n)$