

Q12- Merge 2 sorted arrays in O(1) space.

 $i/p \rightarrow aux1[J = \{1,3,5,7\}$   $avx2[J = \{0,2,6,8,9\}$   $0/p \rightarrow aux1[J = \{0,1,2,3\}$   $avx2[J = \{5,6,7,8,9\}$ 

now here we need to merge 2 sorted arrays in the constant space. In merge sort, we merged arrays by taking 2 extra arrays. So let's jump on to the approach & dry run.

Let's say i = n-1 where n is the size of the 1st array and j=0 where j is pointer to 0th index of 2nd array & the size of 2nd array is m.

Just traverse until any one of the away finishes. If while traversing we found arrilli] > arrilli], then we need to swap as arril will have lower elements whereas arrill have larger / bigger elements. Simply decrement is a increment j after the swapping is done.

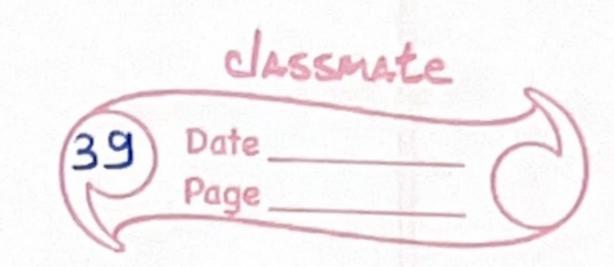
now if awilli] < awiz[j], we simply need to exit as for further i we won't get any i such that arrilli] > awiz[j] as we are given sorted array, so simply break in the else fart.

Dry run.

1) avu  $1 \rightarrow \{1, 3, 5, 7\}$  avu  $2 \rightarrow \{0, 2, 6, 8, 9\}$ 

avoil[i] > avoil[j] -1 simply swap & we get updated avoiags. Also decrement i & increment j.

avoi 2 1 {7,2,6,8,93 aur 1 - 31, 3, 5, 03 avril [i] > avr2 [j] + True & hence swap & then decrement i & increment j. avel + {1,3,2,0} ave2 + {7,5,6,8,9} avrillij > avri2[j] - False & hence break. Now final step is that sort both the arrays. and 1 / 0,1,2,33 avy 2 -1 { 5, 6, 7, 8, 9 3 Hence we have got the final answer. void merge (long long avril [], long long avril] int no int m) { int i=n-1; //Start from last index of and int j = 0; //Start from 0th index of arm 2 //Traverse until any one array or both finishes while (1°>=0 44 j<m) { if (avr [[i]) 2 avr 2 [j]) { swap (avulli), avu2 (j)) j i-- i// Decrement i 1++ j//Increment 1° else break i // We are giver sorted avorage



```
J/Final step

Sort (aurl, aurl +n);

sort (aur2, aur2+n);
```

Time complexity = 0 ((n+m) log (n+m))

Space complexity = 0(1)