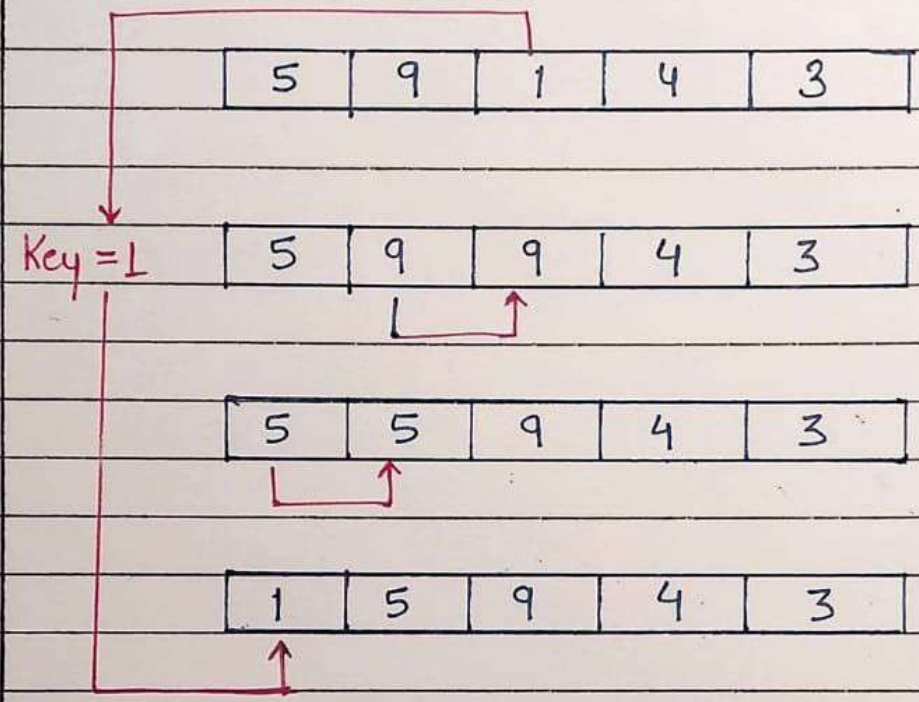
Topic _____ Part-2 _____

Exp. PG. #_____

Date____/____/____

STEP = 2

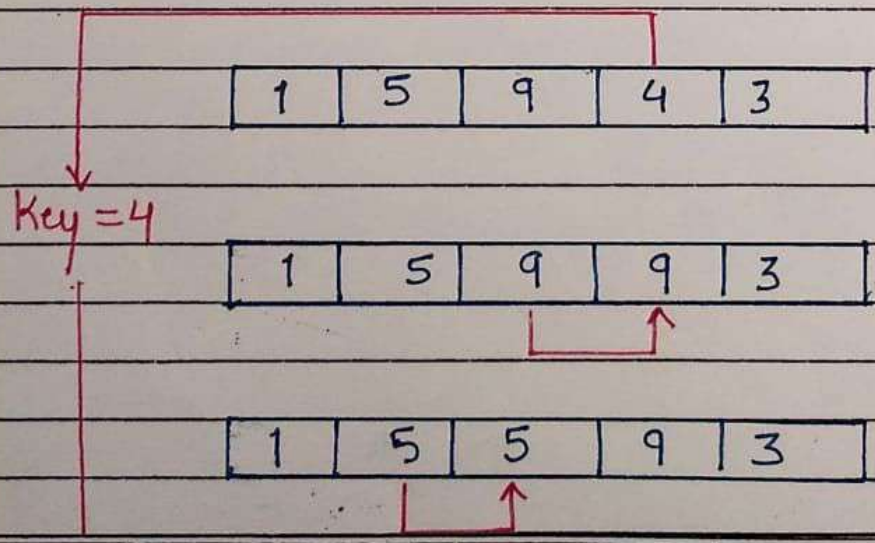


Place 1 at the beginning

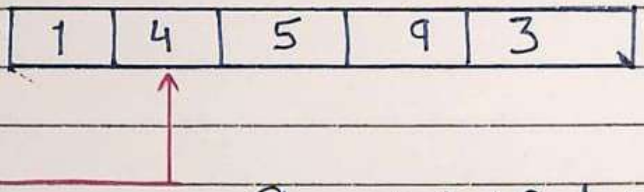
3. Similarly, place every unsorted element at its correct position.

Curious_. programmer

STEP = 3

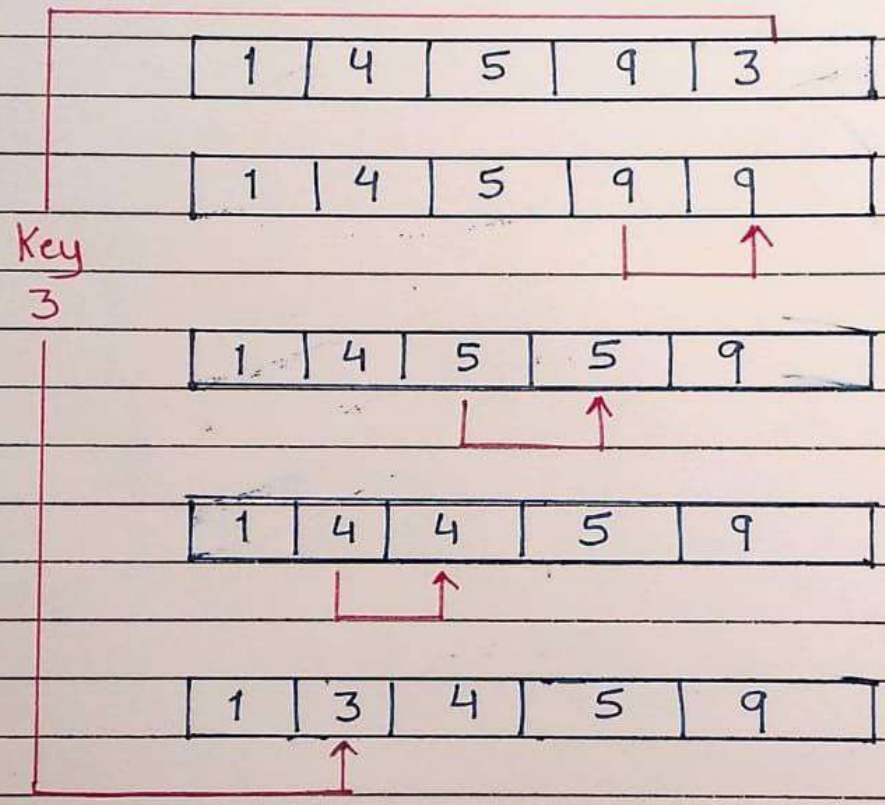


Teacher's Signature_____



Place 4 behind 1

STEP = 4 Curious_.programmer



Place 3 behind 1 and the array is sorted.

4. Merge Sort

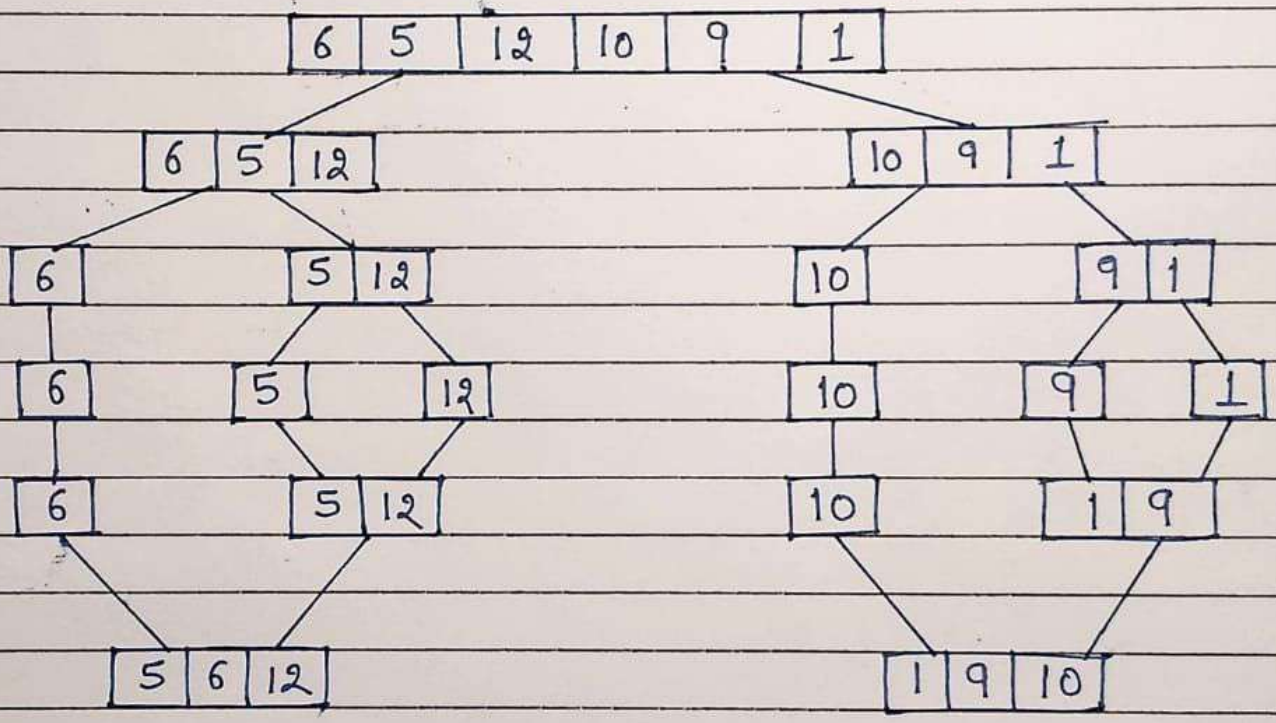
Merge sort is one of the most popular sorting algorithm that is based on the principle of divide and conquer Algorithm.

Exp. PG. #_____

Topic _____ Date ___/___/___

A problem is divided into multiple sub-problems. Each sub-problem is solved individually. Finally sub-problems are combined to form the final solution.

Curious_.programmer



1 5 6 9 10 12

Merge sort example.

4. Quick Sort

Quicksort is a sorting algorithm based on the divide and conquer approach. There,

Teacher's Signature_____

Topic _____ Exp. PG. #_____

Date ___/___/___

1. An array is divided into subarrays by selecting a pivot element (element selected from the array).

   While dividing the array the pivot element should be positioned in such a way that element less than pivot are kept on the left side and elements greater than pivot are on the right side of the pivot.

2. The left and right subarray are also divided using the same approach. This process continues until each subarray contains a single element.

3. At this point, element are already sorted. finally, elements are combined to form a sorted array.

* Working of quick Sort Algorithm

Curious_.programmer

* Select the pivot element

There are different variations of quicksort where the pivot element is selected from different positions. Here we will be selecting the rightmost element of the array as the pivot element.

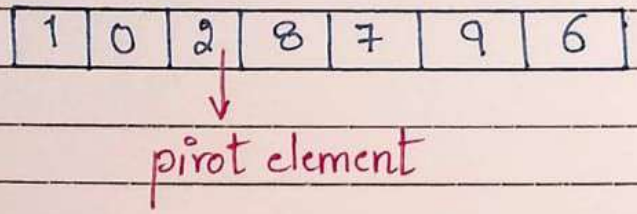
8	7	6	1	0	9	2	→ pivot element

* Rearrange the array

Now the elements of the array are rearranged so

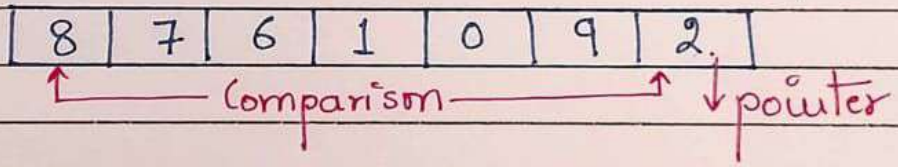
Teacher's Signature_____

that element that are smaller than the pivot are put on the left and the element greater than pivot are put on the right.

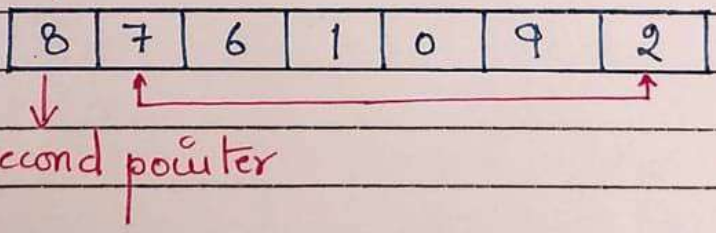


1. ~~put~~ A pointer is fixed at the pivot element. The pivot element is compared with the elements beginning from the first index.

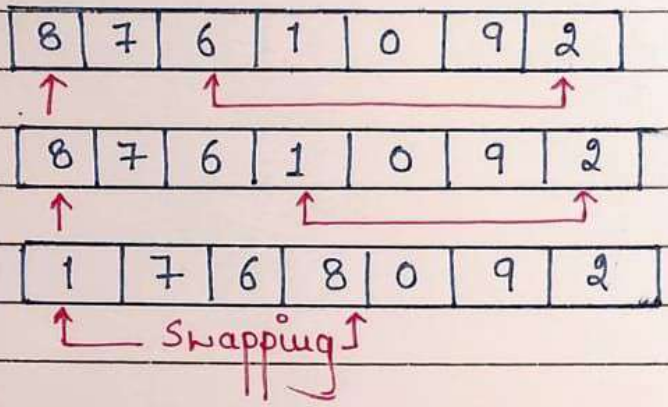
Curious_. programmer



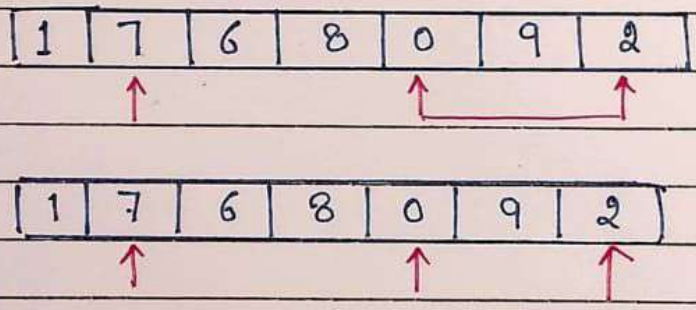
2. If the element is greater than the pivot element, a second pointer is set for that element.



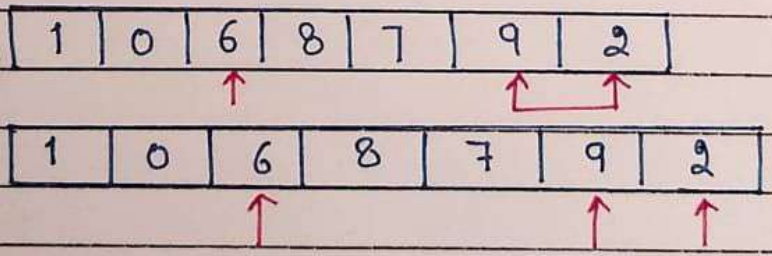
3. Now, pivot is compared with other elements. If an element smaller than the pivot element is reached, the smaller element is swapped with the greater element found earlier.



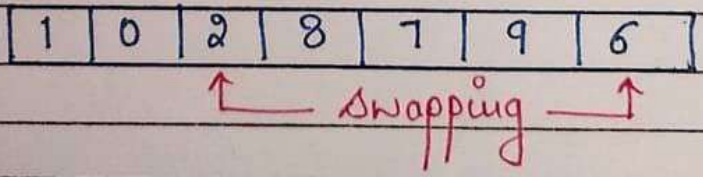
4. Again, the process is repeated to set the next greater element as the second pointer. And, swap it with another smaller element.



5. The process goes on until the second last element is reached.



6. Finally, the pivot element is swapped with the second pointer.



Topic _____ Exp. PG. #_____

Date ___/___/___

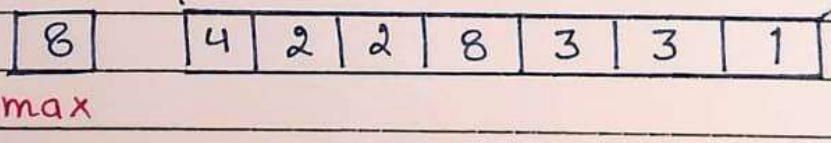
* Non-Comparison Based. (Sorting)

1. Counting Sort.

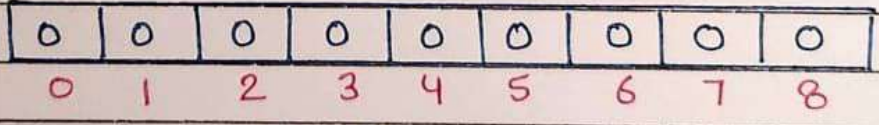
Counting sort is a sorting algorithm that sorts the elements of an array by counting the number of occurences of each unique element in the array. The count is stored in an auxilary array and the sorting is done by mapping the count as an index of the auxilary array.

* Working of Counting Sort. Curious_. programmer

1. find the maximum element (let it be max) from the given array.



2. Initialize an array of length max + 1 with all elements 0. This array is used for storing the count of the elements in the array.



Count array.

3. Store the count of each element at their respective index in count array.

Teacher's Signature_____

Topic _____ Exp. PG. #_____

Date ___/___/___

for example: If the count of element 3 is 2 then, 2 is stored in the 3rd position of count array. If element "5" is not present in the array, then 0 is stored in 5th position.

0	1	2	3	4	5	6	7	8
0	1	2	2	1	0	0	0	1

Count of each element stored

4. Store cumulative sum of the elements of the count array. It help in placing the elements into the correct index of the sorted array.

0	1	2	3	4	5	6	7	8
0	1	3	5	6	6	6	6	7

Curious_.programmer

Cumulative Count

5. find the index of each element of the original array in the count array. This gives the cumulative count. Place the element at the index calculated as shown in

array

4	2	2	8	3	3	1

count

0	1	2	3	4	5	6	7	8
0	1	3	5	6	6	6	6	7

↓ 6-1=5

output

0	1	2	3	4	5	6
01	2	2	3	3	4	8

Teacher's Signature_____

Topic _____ Exp. PG. #_____

Date ___/___/___

6. After placing each element at its correct position, decreased its count by one.

Curious_programmer

2. Radix Sort.

Radix sort is a sorting algorithm that sort the elements by first grouping the individual digit of the samed place value. Then, sort the elements according to their increasing / decreasing order.

Suppose we have an array of 8 elements. First, we will sort elements based on the value of the unit place. Then, we will sort elements based on the value of the tenth place. This process goes on until the last significant place.

let the inital array be [121, 432, 564, 23, 1, 45, 788]. It is sorted according to radix sort as shown in figure.

1	2	1
0	0	1
4	3	2
0	2	3
5	6	4
0	4	5
7	8	8

0	0	1
1	2	1
0	2	3
4	3	2
0	4	5
5	6	4
7	8	8

0	0	1
0	2	3
0	4	5
1	2	1
4	3	2
5	6	4
7	8	8

Teacher's Signature_____

Topic _____ Exp. PG. #_____

Date ___/___/___

* Working of radix Sort. Curious_.programmer

1. In the array we have the largest number 788. It has 3 digits. Therefore, the loop should go upto hundred place (3 times).

2. Now, go through each significant place one by One. Use any stable sorting technique to sort the digit at each significant place. We used counting sort for this. Sort the element based on the unit place digit $x=0$.

Array

1	2	4	3	1	5	8

	0	1	2	3	4	5	6	7	8
count	0	2	3	4	5	6	6	6	7

↓ 3-1=2

	1	1	2	3	4	5	8
output							

121	001	432	23	564	045	788

3. Now, sort the elements based on digit at tens place.

001	121	023	432	045	564	788

4. finally, sort the elements based on the digit at hundreds place.

001	023	045	121	432	564	788

Teacher's Signature_____

PDF uploaded on Telegram (Link in bio)

Follow For Posts like this!