sorting

25 December 2022 21:13

Selection sort:-

#selecting sort:-

0	1	2	3
7	5	4	2

array 2 part

Sorted → left

Unsorted. → Right

Pahle minimum element Utha ke lao.

Round 1:-

0	1	2	3
7	5	4	2

↑ i=0 ↑ min.

Round 1:-

$\underline{2}$ $\underline{5\ 4\ 7}$

Sorted ↗ i=1 Unsorted.

Round2:-

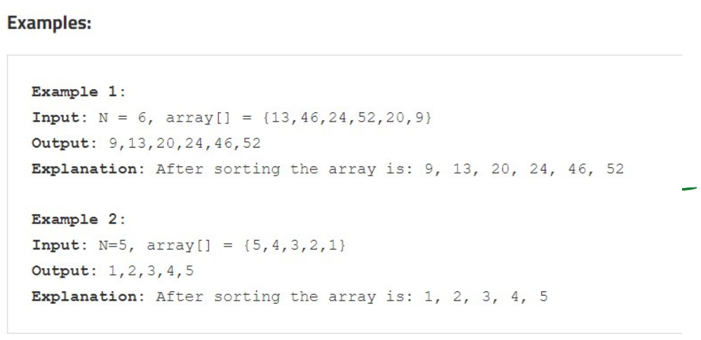
$\underline{2\ \ 4}$ $\underline{5\ \ 7}$

Sorted Unsorted.

2 4 5 7

$\underline{2\ \ 4\ \ 5}$ $\underline{7}$ ↙ min.

sorted. i= 3



Examples:

```
Example 1:
Input: N = 6, array[] = {13,46,24,52,20,9}
Output: 9,13,20,24,46,52
Explanation: After sorting the array is: 9, 13, 20, 24, 46, 52

Example 2:
Input: N=5, array[] = {5,4,3,2,1}
Output: 1,2,3,4,5
Explanation: After sorting the array is: 1, 2, 3, 4, 5
```

Screen clipping taken: 25-12-2022 21:20

0	1	2	3	4	5
13	46	24	52	20	9

↑ ↑ min

13>=9 swap

Round 1:- 9 | $\boxed{46}$ 24 52 20 $\boxed{13}$

Sorted Unsorted ↑ min.

46>=13 Swap

Round 2:- 9 13 24 52 20 46

24>=20 swap

Round 3:- 9 13 20 52 24 46

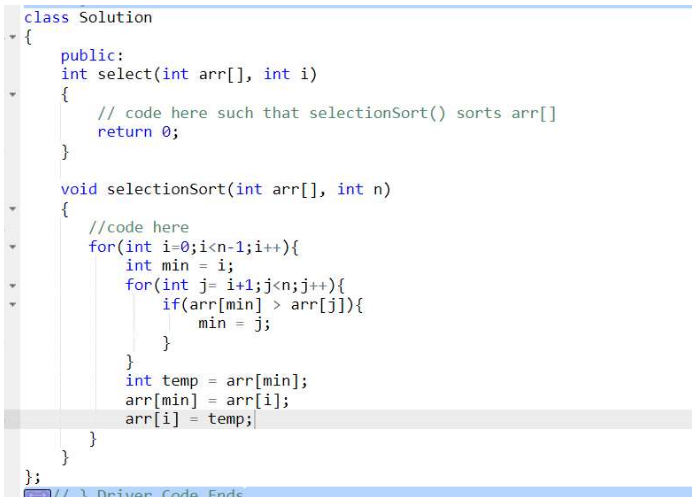
52>=24

Round 4:- 9 13 20 24 52 46

52>=46

Round 5:- 9 13 20 24 46 52

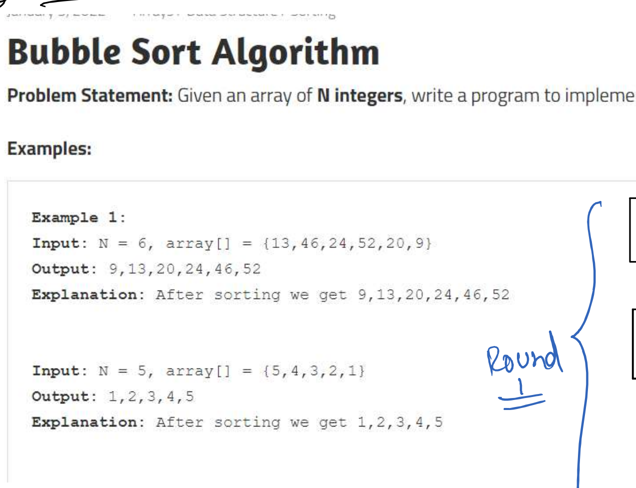
sorted Array



```
class Solution
{
    public:
    int select(int arr[], int i)
    {
        // code here such that selectionSort() sorts arr[]
        return 0;
    }

    void selectionSort(int arr[], int n)
    {
       //code here
       for(int i=0;i<n-1;i++){
           int min = i;
           for(int j= i+1;j<n;j++){
               if(arr[min] > arr[j]){
                   min = j;
               }
           }
           int temp = arr[min];
           arr[min] = arr[i];
           arr[i] = temp;
       }
    }
};
// } Driver Code Ends
```

Screen clipping taken: 25-12-2022 21:43

② Bubble Sort:-



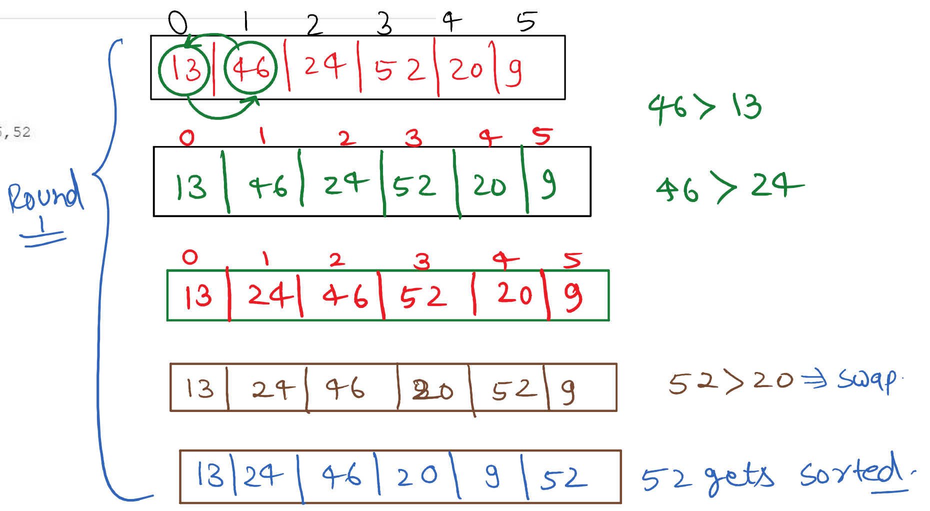
Bubble Sort Algorithm

Problem Statement: Given an array of **N integers**, write a program to implement the Bubble Sorting algorithm.

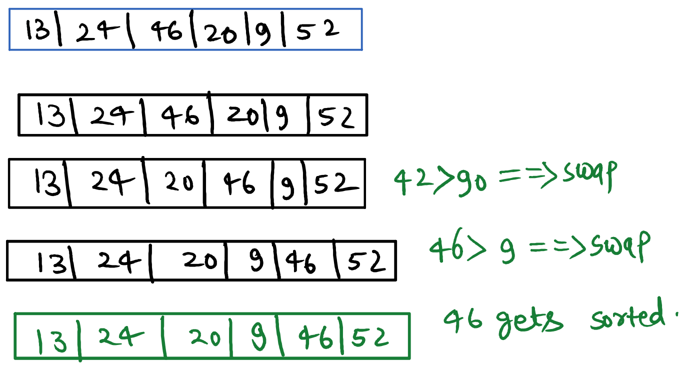
Examples:

```
Example 1:
Input: N = 6, array[] = {13,46,24,52,20,9}
Output: 9,13,20,24,46,52
Explanation: After sorting we get 9,13,20,24,46,52


Input: N = 5, array[] = {5,4,3,2,1}
Output: 1,2,3,4,5
Explanation: After sorting we get 1,2,3,4,5
```

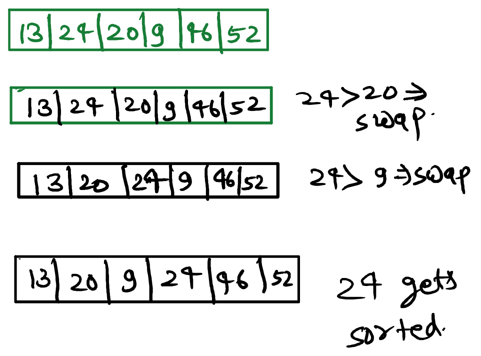
Screen clipping taken: 25-12-2022 21:44



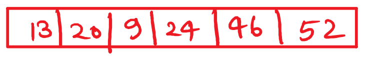
Round 2:-



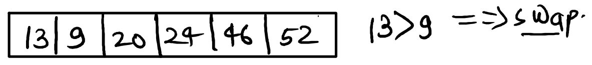
Round 3:-



Iteration 4:-



Iteration 5:-



13	20	9	24	46	52

20>9 ⇒ swap

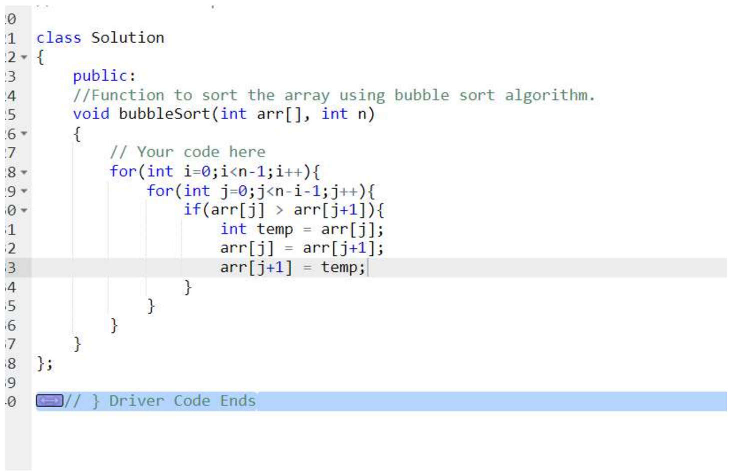
13	9	20	24	46	52

20 gets sorted

9	13	20	24	46	52

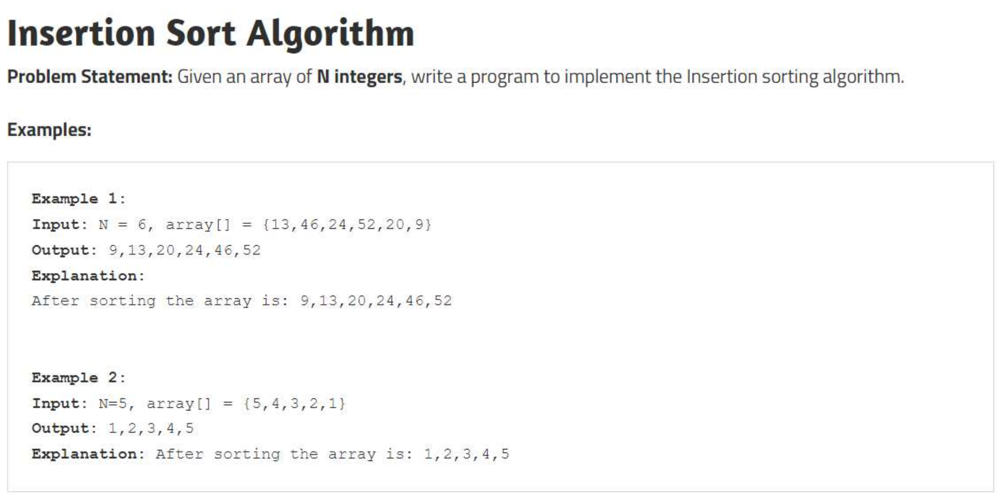
final sorted array.

Screen clipping taken: 25-12-2022 22:07



```
class Solution
{
    public:
    //Function to sort the array using bubble sort algorithm.
    void bubbleSort(int arr[], int n)
    {
        // Your code here
        for(int i=0;i<n-1;i++){
            for(int j=0;j<n-i-1;j++){
                if(arr[j] > arr[j+1]){
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
            }
        }
    }
};

// } Driver Code Ends
```

que 2:- Insertion Sort:-



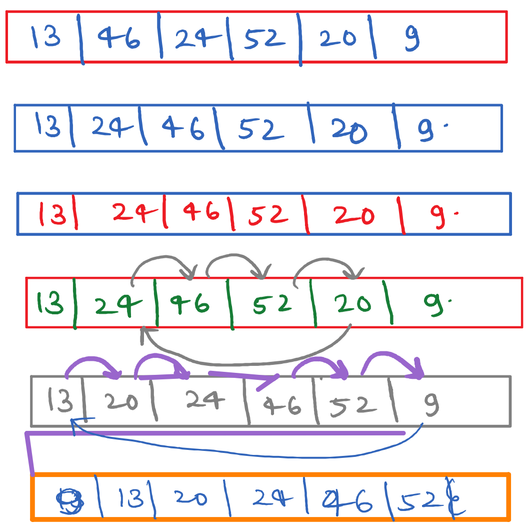
Insertion Sort Algorithm

Problem Statement: Given an array of **N integers**, write a program to implement the Insertion sorting algorithm.

Examples:

```
Example 1:
Input: N = 6, array[] = {13,46,24,52,20,9}
Output: 9,13,20,24,46,52
Explanation:
After sorting the array is: 9,13,20,24,46,52


Example 2:
Input: N=5, array[] = {5,4,3,2,1}
Output: 1,2,3,4,5
Explanation: After sorting the array is: 1,2,3,4,5
```

Screen clipping taken: 25-12-2022 22:08



Remain as it is as 13<46.

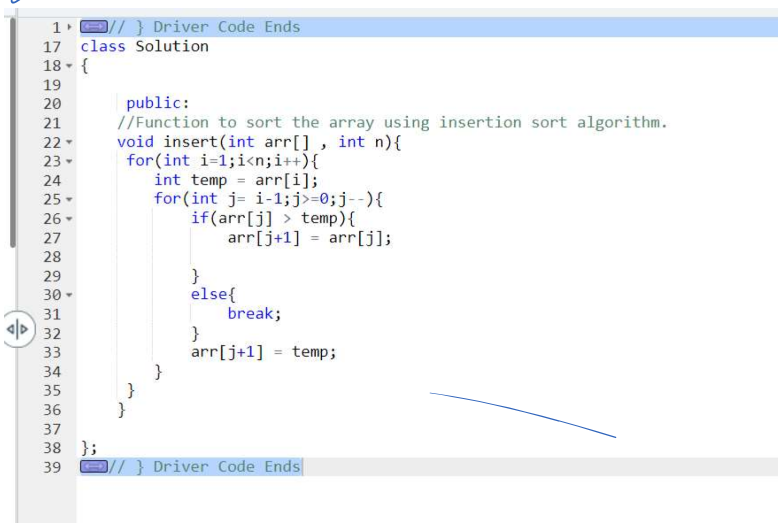
24<46 ⇒ swap.

Remain as it is as 46<52

placing 20 at it's appropriate position.

placing 9 at it's appropriate position.

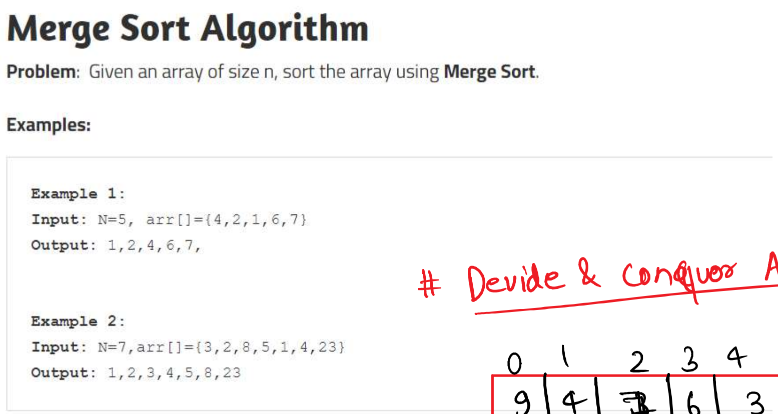
final sorted array



```
// } Driver Code Ends
class Solution
{

    public:
    //Function to sort the array using insertion sort algorithm.
    void insert(int arr[] , int n){
     for(int i=1;i<n;i++){
        int temp = arr[i];
        for(int j= i-1;j>=0;j--){
            if(arr[j] > temp){
                arr[j+1] = arr[j];

            }
            else{
                break;
            }
            arr[j+1] = temp;
        }
     }
    }

};
// } Driver Code Ends
```

Screen clipping taken: 25-12-2022 22:22

4:- Merge Sort:



Merge Sort Algorithm

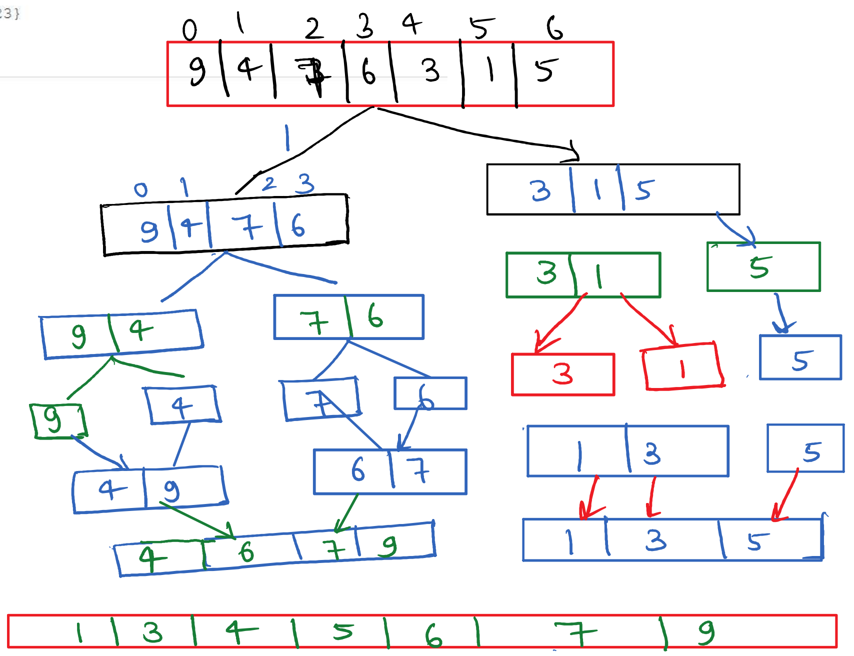
Problem: Given an array of size n, sort the array using **Merge Sort**.

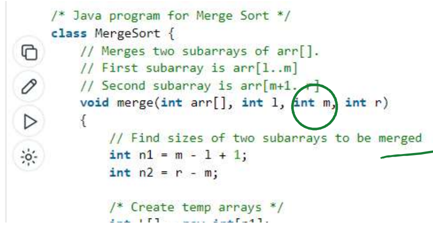
Examples:

```
Example 1:
Input: N=5, arr[]={4,2,1,6,7}
Output: 1,2,4,6,7,


Example 2:
Input: N=7,arr[]={3,2,8,5,1,4,23}
Output: 1,2,3,4,5,8,23
```

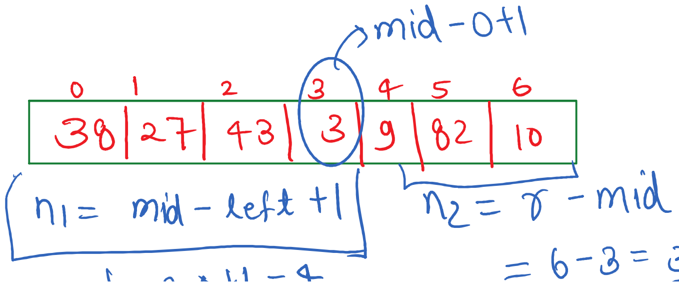
Screen clipping taken: 25-12-2022 22:24

Devide & conquer Algorithm:





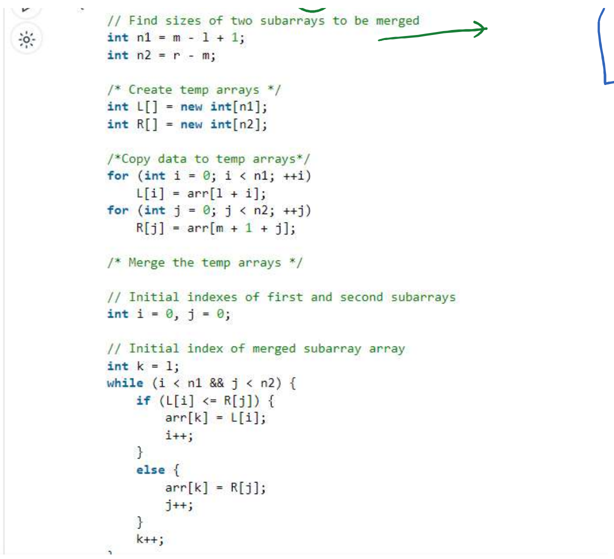
```
/* Java program for Merge Sort */
class MergeSort {
    // Merges two subarrays of arr[].
    // First subarray is arr[l..m]
    // Second subarray is arr[m+1..r]
    void merge(int arr[], int l, int m, int r)
    {
        // Find sizes of two subarrays to be merged
        int n1 = m - l + 1;
        int n2 = r - m;

        /* Create temp arrays */
```



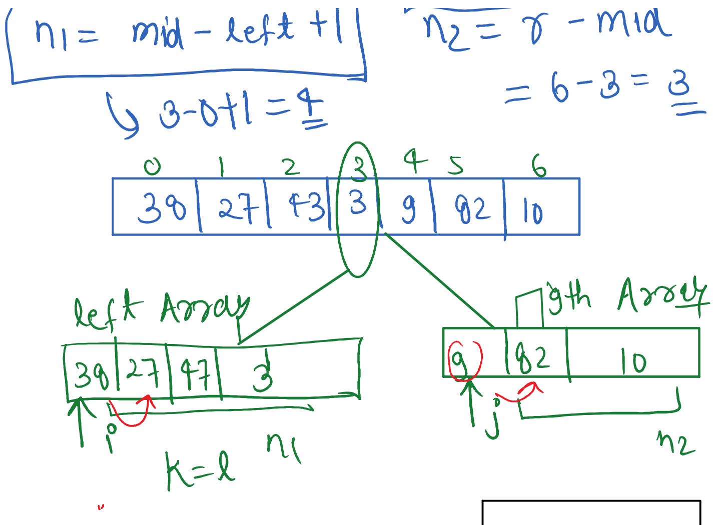
$n_1 = mid - left + 1$

$n_2 = r - mid$

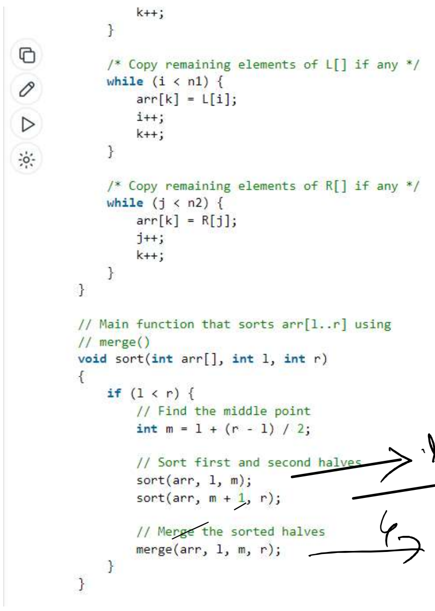
$= 6 - 3 = 3$



```java
    // Find sizes of two subarrays to be merged
    int n1 = m - l + 1;
    int n2 = r - m;

    /* Create temp arrays */
    int L[] = new int[n1];
    int R[] = new int[n2];

    /*Copy data to temp arrays*/
    for (int i = 0; i < n1; ++i)
        L[i] = arr[l + i];
    for (int j = 0; j < n2; ++j)
        R[j] = arr[m + 1 + j];

    /* Merge the temp arrays */

    // Initial indexes of first and second subarrays
    int i = 0, j = 0;

    // Initial index of merged subarray array
    int k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
```



Screen clipping taken: 25-12-2022 23:00



```java
        k++;
    }

    /* Copy remaining elements of L[] if any */
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy remaining elements of R[] if any */
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

// Main function that sorts arr[l..r] using
// merge()
void sort(int arr[], int l, int r)
{
    if (l < r) {
        // Find the middle point
        int m = l + (r - l) / 2;

        // Sort first and second halves
        sort(arr, l, m);
        sort(arr, m + 1, r);

        // Merge the sorted halves
        merge(arr, l, m, r);
    }
}
```

TC :- O(n logn)   SC :- O(n)

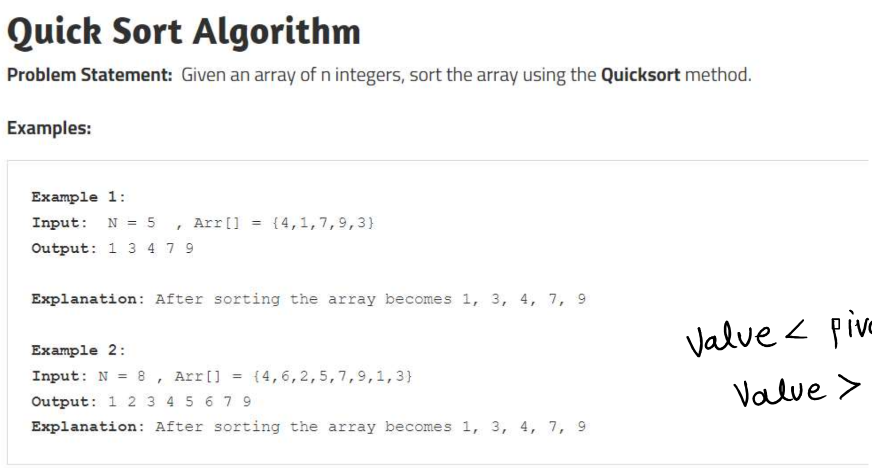
→ left , mid → first half sort

second half sort .

↳ merge both the Array left half & Right half

Screen clipping taken: 25-12-2022 23:00

ques :- Quick sort :



Quick Sort Algorithm

Problem Statement: Given an array of n integers, sort the array using the **Quicksort** method.

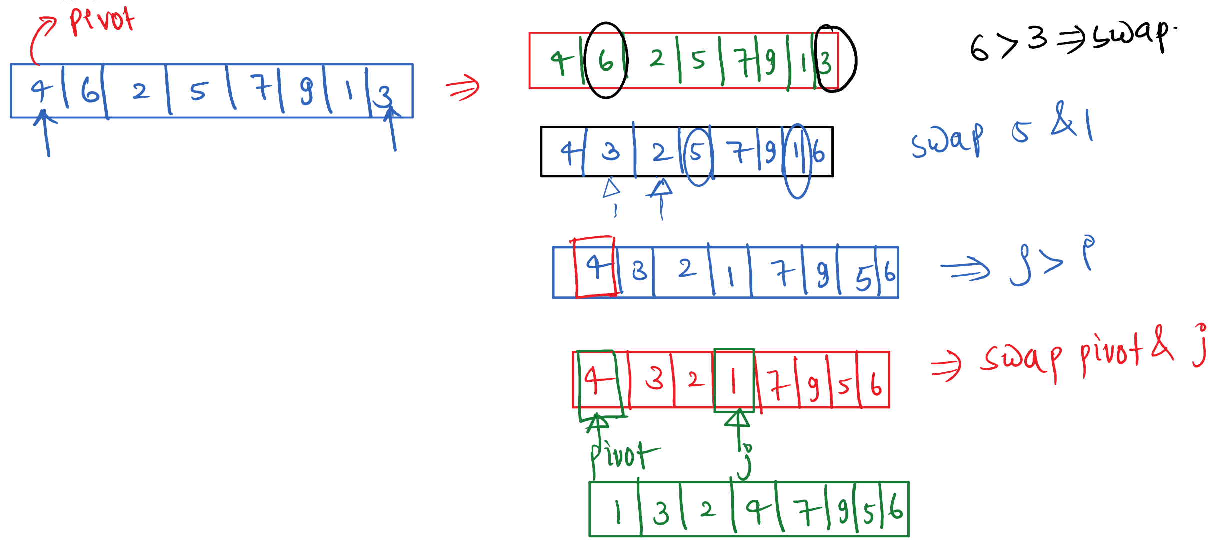
Examples:

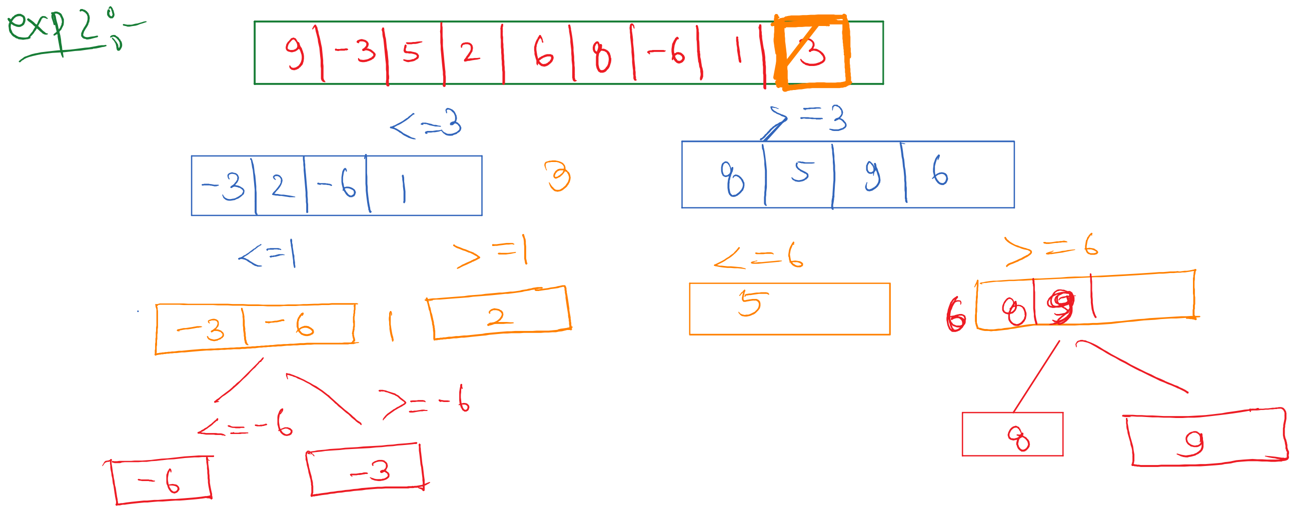
```
Example 1:
Input:  N = 5  , Arr[] = {4,1,7,9,3}
Output: 1 3 4 7 9

Explanation: After sorting the array becomes 1, 3, 4, 7, 9

Example 2:
Input: N = 8 , Arr[] = {4,6,2,5,7,9,1,3}
Output: 1 2 3 4 5 6 7 9
Explanation: After sorting the array becomes 1, 3, 4, 7, 9
```

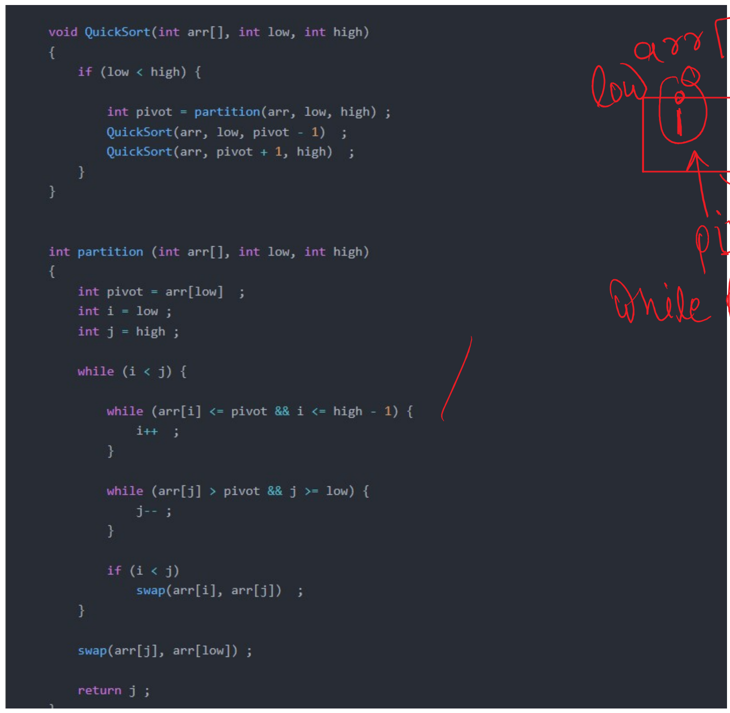
Value < pivot → left

Value > pivot → Right

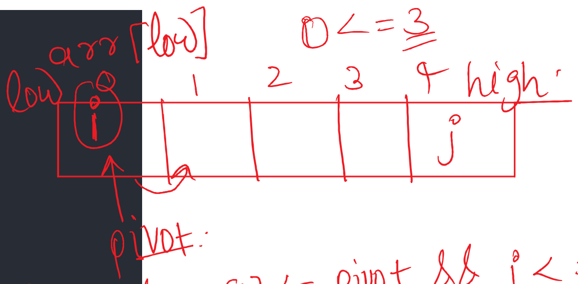
Screen clipping taken: 25-12-2022 23:09







```
void QuickSort(int arr[], int low, int high)
{
    if (low < high) {

        int pivot = partition(arr, low, high) ;
        QuickSort(arr, low, pivot - 1)  ;
        QuickSort(arr, pivot + 1, high)  ;
    }
}


int partition (int arr[], int low, int high)
{
    int pivot = arr[low]  ;
    int i = low ;
    int j = high ;

    while (i < j) {

        while (arr[i] <= pivot && i <= high - 1) {
            i++  ;
        }

        while (arr[j] > pivot && j >= low) {
            j-- ;
        }

        if (i < j)
            swap(arr[i], arr[j])  ;
    }

    swap(arr[j], arr[low]) ;

    return j ;
```



while (arr[i] <= pivot && i <= high-1)
i++ left side chhota

arr[j] > pivot && j >= low
j-- right side bada

if (i<j)
{ swap (arr[i], arr[j])
(arr[j]), arr[i])

Screen clipping taken: 25-12-2022 23:43

```
        swap(arr[j], arr[low]) ;

        return j ;
```

Screen clipping taken: 25-12-2022 23:43

} swap(arr[j],

swap(arr[j], arr[i])

return j;