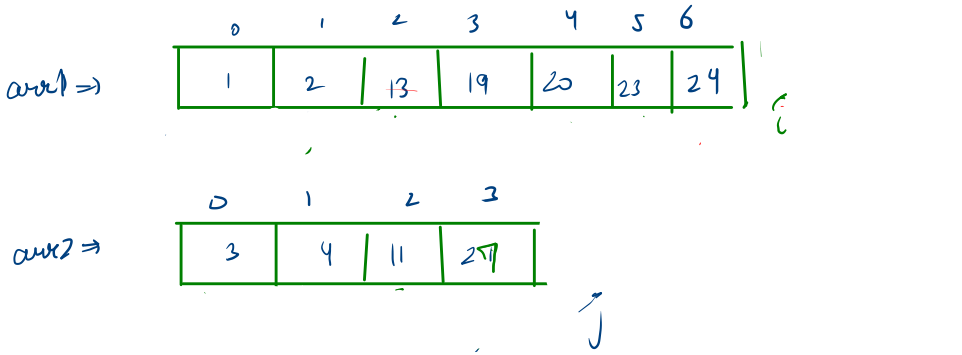
Ques Merge two sorted arrays.




```java
public static int[] merge2SortedArrays(int[] arr1, int[] arr2){
    int n = arr1.length;
    int m = arr2.length;

    int[] ans = new int[n+m];

    int i=0;
    int j=0;
    int k=0;
    int ei1 = n-1; // ei for arr1
    int ei2 = m-1; // ei for arr2

    while(i<=ei1 && j<=ei2){
        if(arr1[i] < arr2[j]){
            ans[k] = arr1[i];
            i++;
        } else {
            ans[k] = arr2[j];
            j++;
        }
        k++;
    }

    while(i<=ei1){
        ans[k] = arr1[i];
        i++;
        k++;
    }

    while(j<=ei2){
        ans[k] = arr2[j];
        j++;
        k++;
    }

    return ans;
}
```

O(n+m)

# Merge Sort

Faith

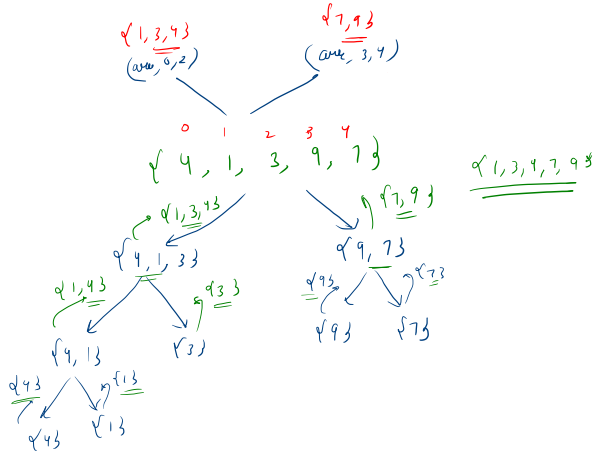
It will sort half of array

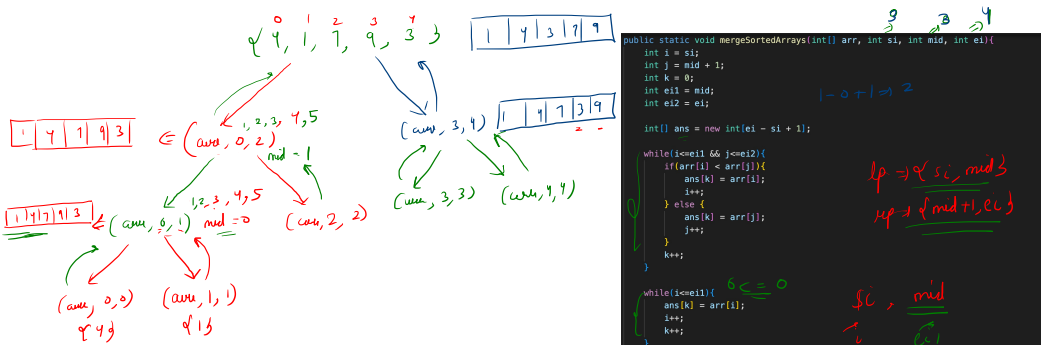
Expectation

This array will be sorted

mid = (0+4)/2 => 2

Vertically




```
public static void mergeSort(int[] arr, int si, int ei){
    if(si==ei){
        return;
    }

    int mid = (si + ei)/2;

    mergeSort(arr, si, mid);
    mergeSort(arr, mid+1, ei);
    mergeSortedArrays(arr,si,mid,ei); -> O(n)
}
```

log n

```
public static void mergeSortedArrays(int[] arr, int si, int mid, int ei){
    int i = si;
    int j = mid + 1;
    int k = 0;
    int ei1 = mid;
    int ei2 = ei;

    int[] ans = new int[ei - si + 1];

    while(i<=ei1 && j<=ei2){
        if(arr[i] < arr[j]){
            ans[k] = arr[i];
            i++;
        } else {
            ans[k] = arr[j];
            j++;
        }
        k++;
    }

    while(i<=ei1){
        ans[k] = arr[i];
        i++;
        k++;
    }

    while(j<=ei2){
        ans[k] = arr[j];
        j++;
        k++;
    }

    k=0;
    for(int idx = si; idx<=ei; idx++){
        arr[idx] = ans[k];
        k++;
    }
}
```

log n * N => N log n

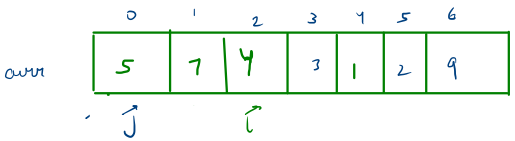
break till 10:30

Ques Partition an array on the basis of pivot elements.

0 → j-1 ⇒ smaller (blue)

j → i-1 ⇒ greater (green)

i → last ⇒ unknown




if (arr[i] <= pivot) {
swap (i

} else {
i++;
}

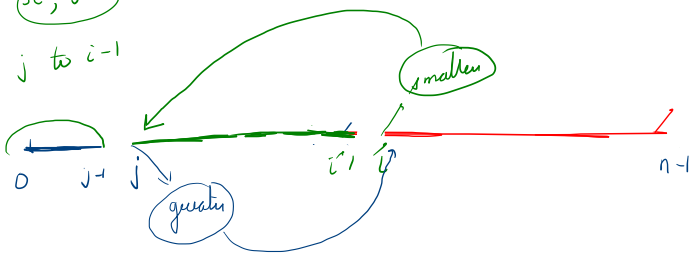
(0, -1) ⇒ X

(0, -1) ⇒ X

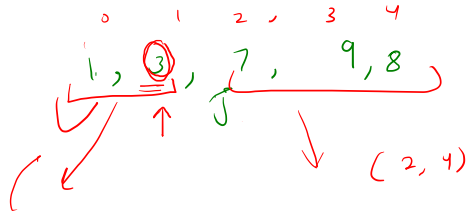
(0, n-1) ⇒ unknown

smaller ⇒ (si, j-1)

larger ⇒ j to i-1




⇒ (j-1)



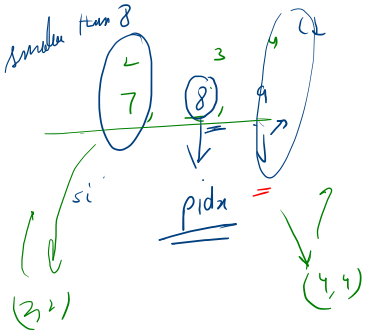

pivot = 3

merge

quick select

pivot = 8 greater than 8

smaller than 8




```java
public static int partitionPivot(int[] arr, int pivot, int si, int ei){
    int i = si;
    int j = si;

    while(i<=ei){
        if(arr[i]<=pivot){ // increase smaller area
            swap(arr,i,j);
            i++;
            j++;
        } else { // increase larger area
            i++;
        }
    }

    return j-1;
}
public static void quickSort(int[] arr, int si, int ei){
    if(si>=ei){
        return;
    }
    int pivot = arr[ei];
    int pidx = partitionPivot(arr, pivot, si, ei);
    quickSort(arr, si, pidx - 1);
    quickSort(arr, pidx + 1, ei);
}
```

average n log n

$n^2$

O(n)