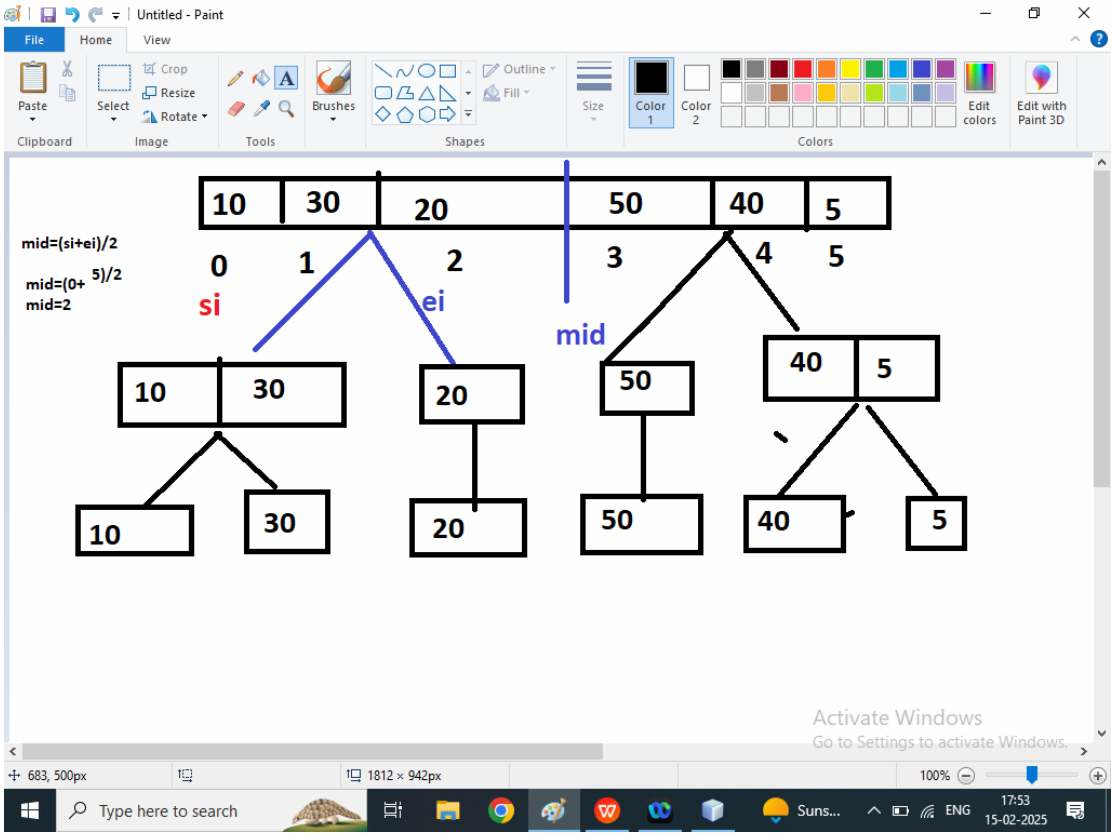
Q1. Explain Merge Sort in Data Structure?
Ans: Merge Sort is a divide and conquer algorithm that divides an array into smaller sub arrays, sorted them, and then merges them back together in sorted order

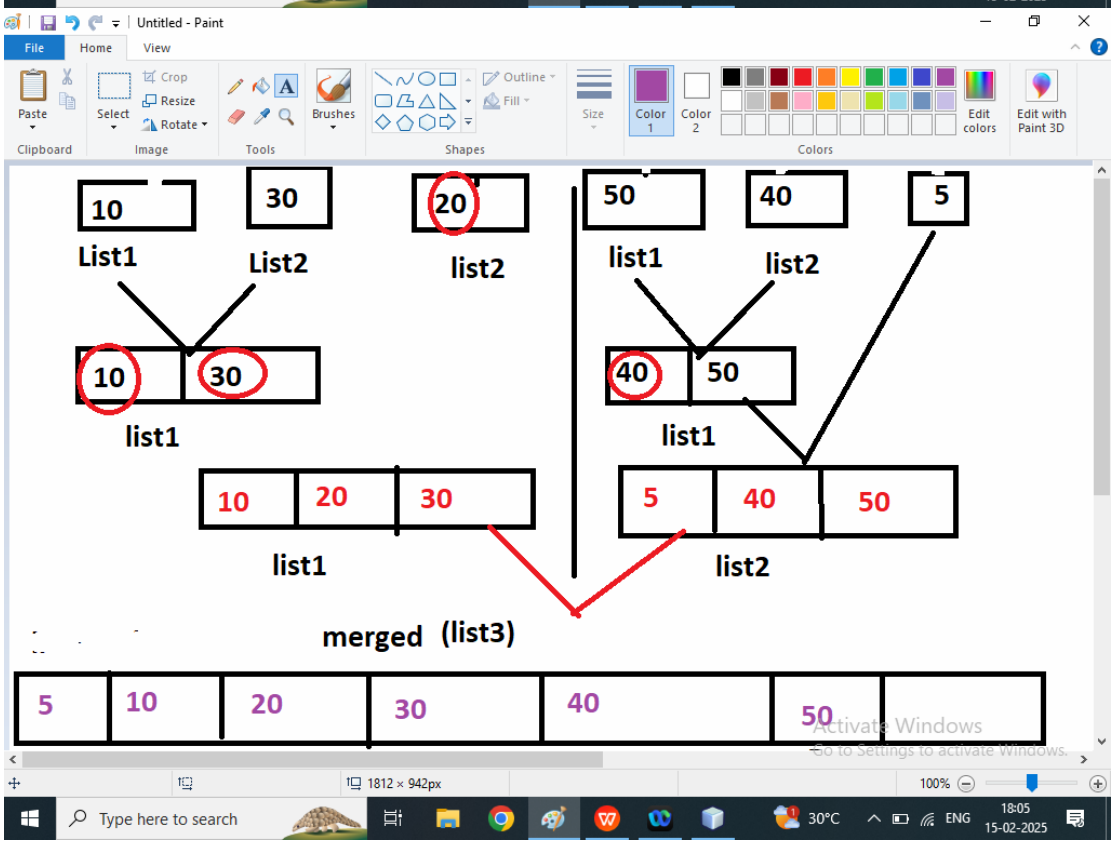
Steps of merge Sort:
1. Divide : Split the array into two parts recursively until each sub array has one element
2. Conquer : Recursively sorts the two part of the array
3. Combine : Combine the sorted parts of the array into single array

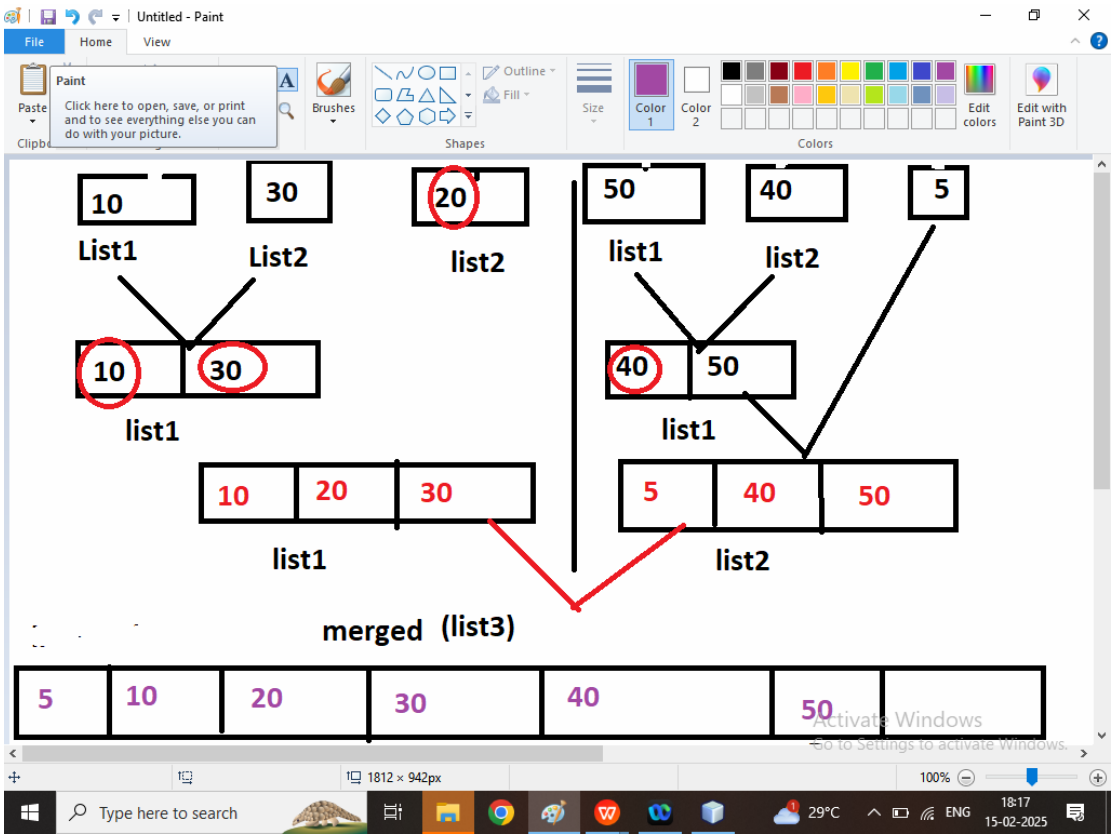
Time Complexity of Merge Sort
Best Case: O(n log n)
Average Case:O(n log n)
Worst Case: O(n log n)

Space Complexity : O(n)

Step1: Divide


Untitled - Paint
mid=(si+ei)/2
mid=(0+ 5)/2
mid=2
10 30 20 50 40 5
0 1 2 3 4 5
si
ei
mid
10 30
20
50
40 5
10
30
20
50
40
5
Activate Windows
Go to Settings to activate Windows.
683, 500px
1812 × 942px
100%
Type here to search
Suns...
ENG
17:53
15-02-2025


Untitled - Paint
10
List1
30
List2
20
list2
50
list1
40
list2
5
10 30
list1
40 50
list1
10 20 30
list1
5 40 50
list2
merged (list3)
5 10 20 30 40 50
Activate Windows
Go to Settings to activate Windows.
1812 × 942px
100%
Type here to search
30°C
ENG
18:05
15-02-2025




```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dsafeb2025;

import javafx.beans.binding.Bindings;


public class MergeSortDemo {
   public static void conquer(int arr[],int si,int mid,int ei)
   {
      int size=ei-si+1;
      int merged[]=new int[size];
      int idx1=si;

      int idx2=mid+1;
      int x=0;//x represent merged array
      while(idx1<=mid && idx2<=ei){
         if(arr[idx1]<=arr[idx2]){
            merged[x++]=arr[idx1++];
         }else{
            merged[x++]=arr[idx2++];
         }
      }
      while(idx1<=mid){
         merged[x++]=arr[idx1++];
      }
      while(idx2<=ei){
```

```
            merged[x++]=arr[idx2++];
        }

    //copy the data into original array
    for(int i1=0,j1=si;i1<size;i1++,j1++){
        arr[j1]=merged[i1];
    }
    }

    public static void divide(int arr[],int si,int ei){
        if(si>=ei){//base condition
            return;
        }
        int mid=(si+ei)/2;
        divide(arr, si, mid);//for left sub array
        divide(arr, mid+1, ei);//for right sub array
        conquer(arr,si,mid,ei);

    }
    public static void main(String[] args) {
        int arr[]={10,30,20,50,40,5};
        System.out.println("Print Before Sorting ");
        for(int i=0;i<arr.length;i++){
            System.out.print("\t"+arr[i]);
        }
        divide(arr, 0, arr.length-1);
        System.out.println("\nPrint After Sorting ");
        for(int i=0;i<arr.length;i++){
            System.out.print("\t"+arr[i]);
        }
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dsafeb2025;

import javafx.beans.binding.Bindings;


public class MergeSortDemo {
    public static void conquer(int arr[],int si,int mid,int ei)
    {
        int size=ei-si+1;
        int merged[]=new int[size];
        int idx1=si;

        int idx2=mid+1;
        int x=0;//x represent merged array
        while(idx1<=mid && idx2<=ei){
            if(arr[idx1]>=arr[idx2]){
                merged[x++]=arr[idx1++];
            }else{
```

```
            merged[x++]=arr[idx2++];
          }
        }
        while(idx1<=mid){
          merged[x++]=arr[idx1++];
        }
        while(idx2<=ei){
          merged[x++]=arr[idx2++];
        }

    //copy the data into original array
    for(int i1=0,j1=si;i1<size;i1++,j1++){
      arr[j1]=merged[i1];
    }
    }

    public static void divide(int arr[],int si,int ei){
      if(si>=ei){//base condition
        return;
      }
      int mid=(si+ei)/2;
      divide(arr, si, mid);//for left sub array
      divide(arr, mid+1, ei);//for right sub array
      conquer(arr,si,mid,ei);

    }
    public static void main(String[] args) {
      int arr[]={10,30,20,50,40,5};
      System.out.println("Print Before Sorting ");
      for(int i=0;i<arr.length;i++){
        System.out.print("\t"+arr[i]);
      }
      divide(arr, 0, arr.length-1);
      System.out.println("\nPrint After Sorting ");
      for(int i=0;i<arr.length;i++){
        System.out.print("\t"+arr[i]);
      }
    }
}
```
