Merge Sort

↳ Merge sort is also an important sorting technique used to sort the given data.

↳ Merge sort technique based on divide and conquer strategy

Divide: Divide the array into two halves.

Conquer: Sort the two sub-arrays

Combine: Merge the two sorted sub-arrays into a single sorted array.

↳ $(\log_2 n)$ pass are required to sort n elements.

↳ Best case Time complexity : $O(n \log_2 n)$

↳ Worst case Time complexity : $O(n \log_2 n)$

↳ Average case Time complexity : $O(n \log_2 n)$

In merge sort we follow the following steps.

1. we take a variable i and store the starting index of our array in this. And we take another variable j and store the last index of array in it.
2. Then we find the middle of array using the formula $(i+j)/2$ and mark the middle index as mid, and break the array into two sub-arrays, from i to mid and from mid+1 to j index.

3. Then we divide these 2 sub-arrays again, just like we divided our main array and this continues.

4. Once we have divided the main array into subarrays with single element, then we start merging the subarrays.

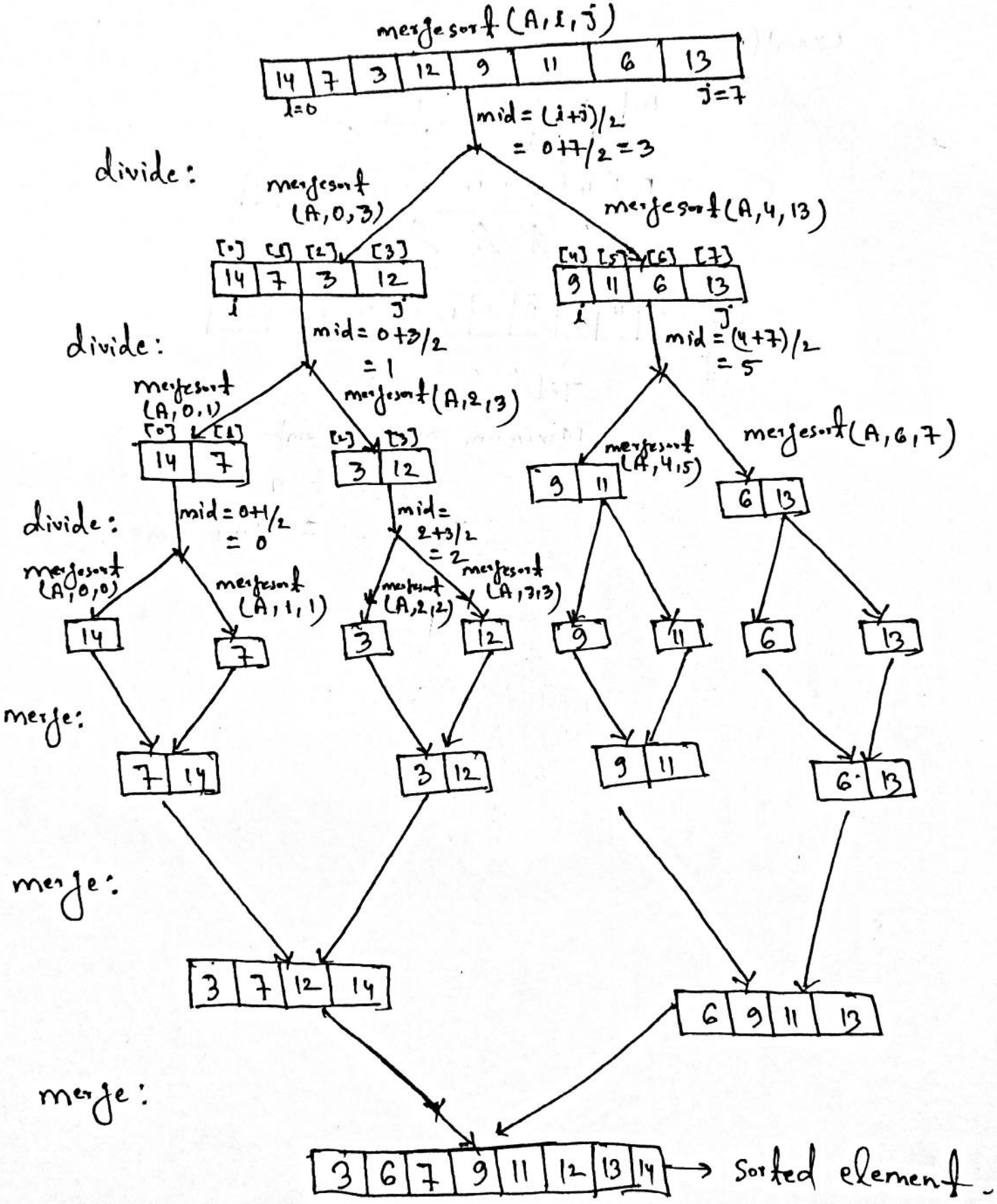
Algorithm:

```
mergesort (A, i, j)
{
  if (i>=j) return;
  else
    {
      mid = (i+j)/2;
      mergesort (A, i, mid);
      mergesort (A, mid+1, j);
      merge (A, i, j);
    }
}.
```

Let's take one example : A

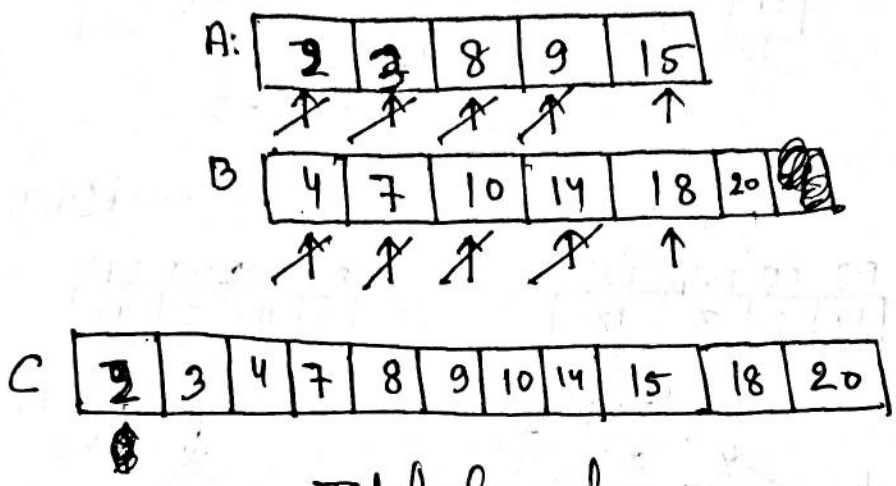
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
14	7	3	12	9	11	6	13

i=0 j=7.



Merging : Concept of merging is applicable only for Sorted Subarrays.

example :



Total element = n

Maximum No. of comparision possible = n

Time. = O(n) Time.

```
#include<stdio.h>
#include<conio.h>
merge(a, i, j)
int a[], i, j;
  {
    int k, b[100], mid, l, start;
      start = i;
      mid = (i+j)/2
        k = mid+1;
        l = i;
  /* FORM ARRAY b */
  while(i<=mid && k<=j)
    {
      if(a[i]<=a[k])
          b[l++] = a[i++];
      else
          b[l++] = a[k++];
    }
  if(i>mid)
      for(;k<=j;)
          b[l++]= a[k++];
  else
    if(k>j)
      for(;i<=mid;)
          b[l++] = a[i++];
  /* COPY BACK TO ARRAY a */
      for(l=start l<=j; l++)
          a[l]= b[l];
}
```

```
mergesort(a, i, j)
    int a[], i, j ;
  {
    int mid;
     if (i>=j) return ;
     mid = (i+j)/2;
     mergesort (a, i, mid);
     mergesort (a, mid+1, j);
      merge (a, i, j);
  }
void main()
{
  int data[100], i, j, n ;
  /* INPUT */
  Printf (" Give n : ");
   Scanf ("%d", &n);
   Printf (" n = %d \n", n);
    for (i=0, i<n; i++)
        scanf (" %d", &data[i]);
    ←Printf (" \n Numbers read are : ");
   for (i=0; i<n; i++)
    Printf ("%d ", data[i]);
    Printf (" \n");
   mergesort (data, 0, n-1);
 /* PRINT RESULT */
 Printf (" \n Sorted numbers are : ");
   for (i=0 ; i<n; i++)
     Printf ("%d", data[i]);
   } getch(); Printf (" \n");
```