Merge sort :

Befor understanding merge sort, first we should know about merging

Merging means merge two sorted array into a single array, so merging is done by need of extra array.

ex:

a = | 2 | 10 | 18 | 20 | 23 | b = | 4 | 9 | 19 | 25 |

we will merge this two array.

let 'i' points on array a.
& j points on array b.

Let the array in which merge array would store be c[];

no. of elements in c be (m+n)
where m = elements in array a.
n = elements in array b.

we will use this concept.

```
if( a[i] > b[j])      (i<m && j<n)
c[k++] = a[i++];
else
c[k++] = b[j++];
```

If still elements in array remains then we can use

```
for( ; i<m ; i++)
 c[k++] = a[i++];
for( ; i<n ; i++)
  c[k++] = b[j++];
```

from above step we get array c as.

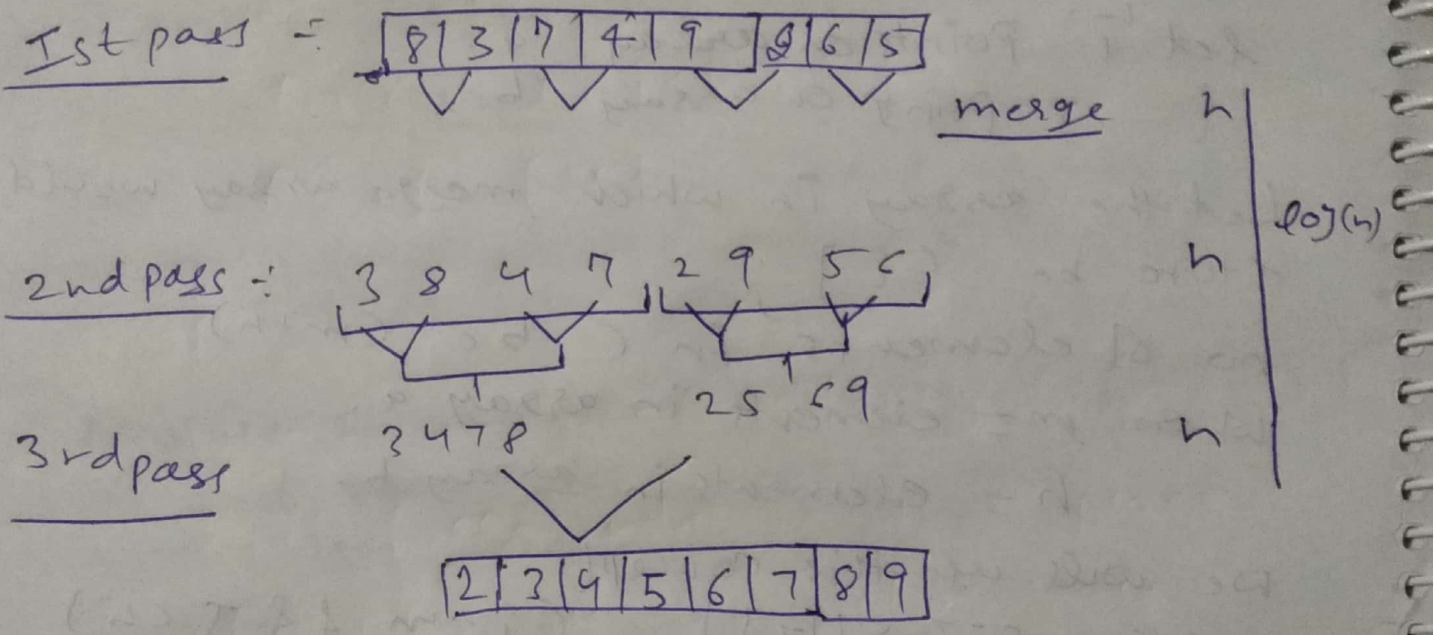
c[] = [2|4|9|10|18|19|20|23|25];

Now, Algorithm for merge sort :-

2-way merging :- can be implemented iteratively or recursively

ex:- A [8|3|7|4|9|2|6|5]

while implementing merge sort, assume each element of array is a separate list & apply merge sort.



n = no. of elements
So, we conclude time complexity of merge sort is O(nlogn)

```
   /* iterative algorithm */
void mergesort (int A[], int n)
{
     for (i=0; i+p-1≤n; i= i+p]
     {
        l=i;
```

```
            h = i+p-1;
            mid = l+h ;
                  ---
                   2

            merge (A, l, mid, h);
          }
      }
      if (p/2 < n)
      merge (A, 0, p/2 -1, n-1);

 }
  /* Recursive*/                      →0    → n-1
 void Rmergesort(int A[], int l , int h)
 {
      int mid;
      if (l<h)
      {
           Rmergesort (A, l, mid);
           Rmergesort (A, mid +1 , h);
           merge (A, l, mid, h);
      }
  }
  /* program for merging */
  void merge (int A[], int l, int mid, int h)
  {
      int i,J,k;
      i=l;  J= mid+1;  k=l;
      while (i≤mid && J≤h)
      {
        if (A[i] ≤ A[J])
      B[k++] = A[i++];
     else
      B[k++] = A[j++]
   }
   for( ; i<=mid; i++)
    B[k++] = A[i];
   for( ; j<=h; j++)
    B[k++] = A[j]
    /* copy elements of B to A again*/
    for(i=l; i<h; i++)
    A[i] = B[i];
 }
```

Now it's Time complexity O(nlogn) & it falls under out of place algorithm as an extra array is needed.