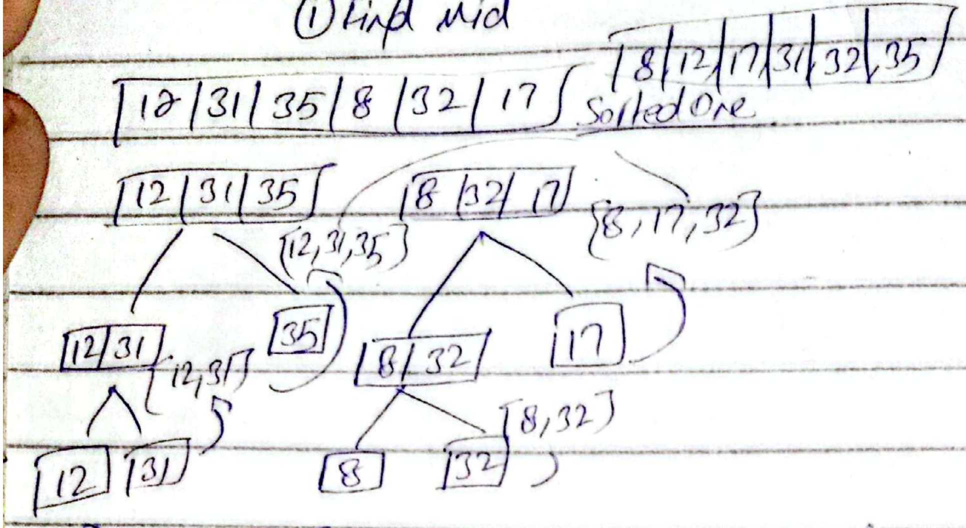
Merge Sort :-

Divide & Conquer

① Divide the array into two parts(equal)

① find mid



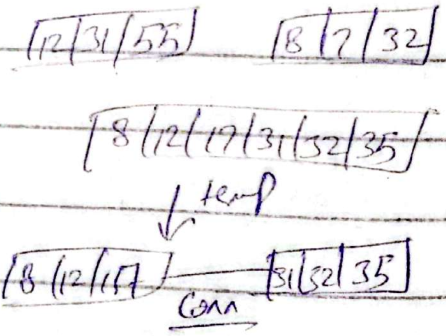
② Merge Parts to create a sorted array
use backtracking as well

Recursive Function

```
mergeSort(arr[], start, end){
  if(s<e){
  int mid= st+ (e-s)/2
  mergeSort (arr, st, mid)
  MergeSort(arr, mid+1, end)
  Merge(arr, st., mid, end
```

Merge step

S=0, mid=2, end=5



```
void merge(arr, st, mid, end){
vector<int> temp
i=st ; j=mid+1
while(i<=mid && j<=end){
 if(A[i]<=A[j]){
   temp.pb(A[i])
     i++
  else
   temp.pb(A[j])
     j++
```

Ascending <=

Descending >=

lefts

```
while(i<=mid){
    temp.pb(A[i])
    i++
}
```

Right

```
while (j<=end){
    temp.pb(A[j])
    j++
}
```

```
for(idx=0; idx<temp.size(); idx++)
A[idx+st] = temp[idx]
```

Time Complexity = O(n) of every loop

↓

total Numbers of elements in array.

Space Complexity = total call * work = log n * n

T.C = O(log n * n)

S.C = O(n)