

# Merge Sort

unsorted



sorted



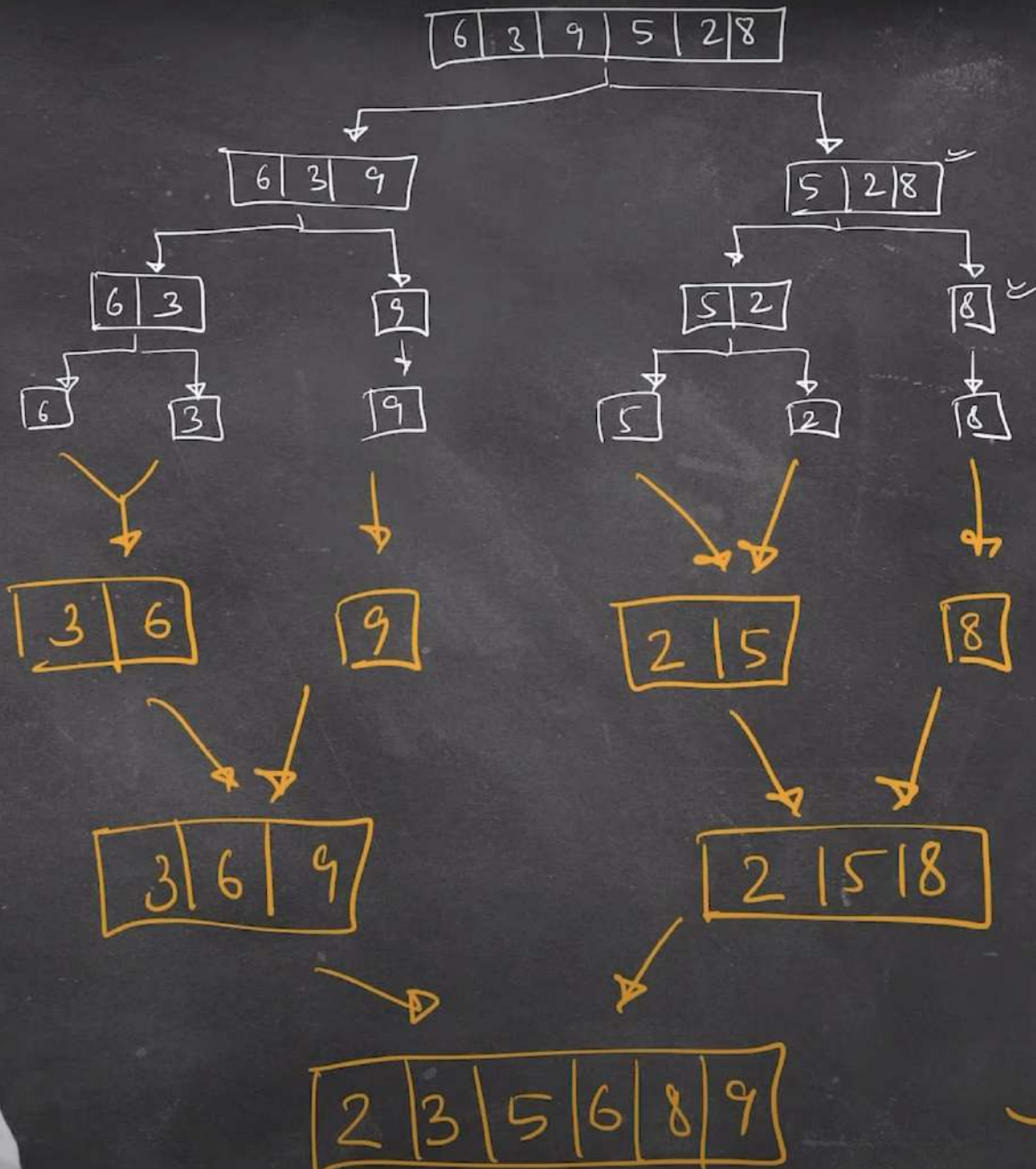
**Approach**

**Divide & Conquer**

# Merge Sort

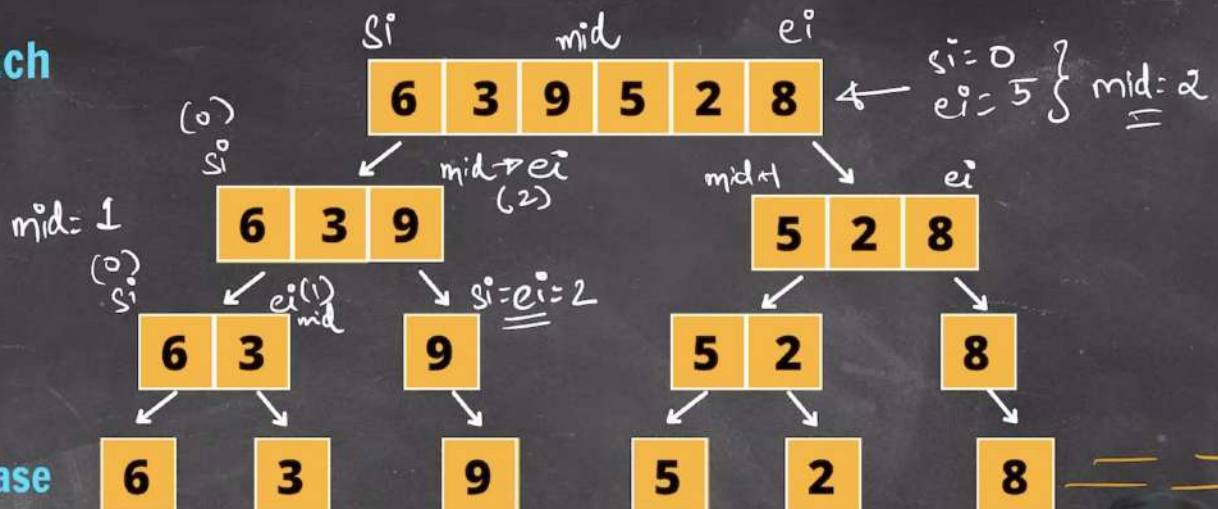
Divide

Conquer =  
SORT



## Approach

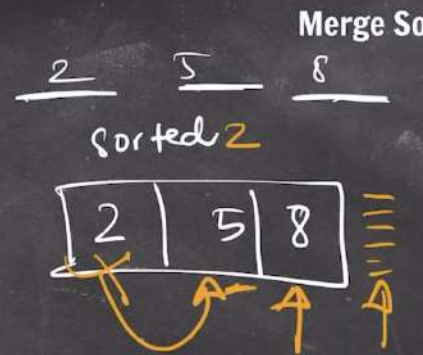
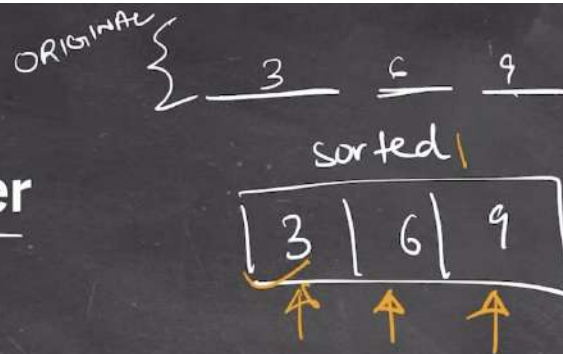
Merge Sort



## Approach

## Divide & Conquer

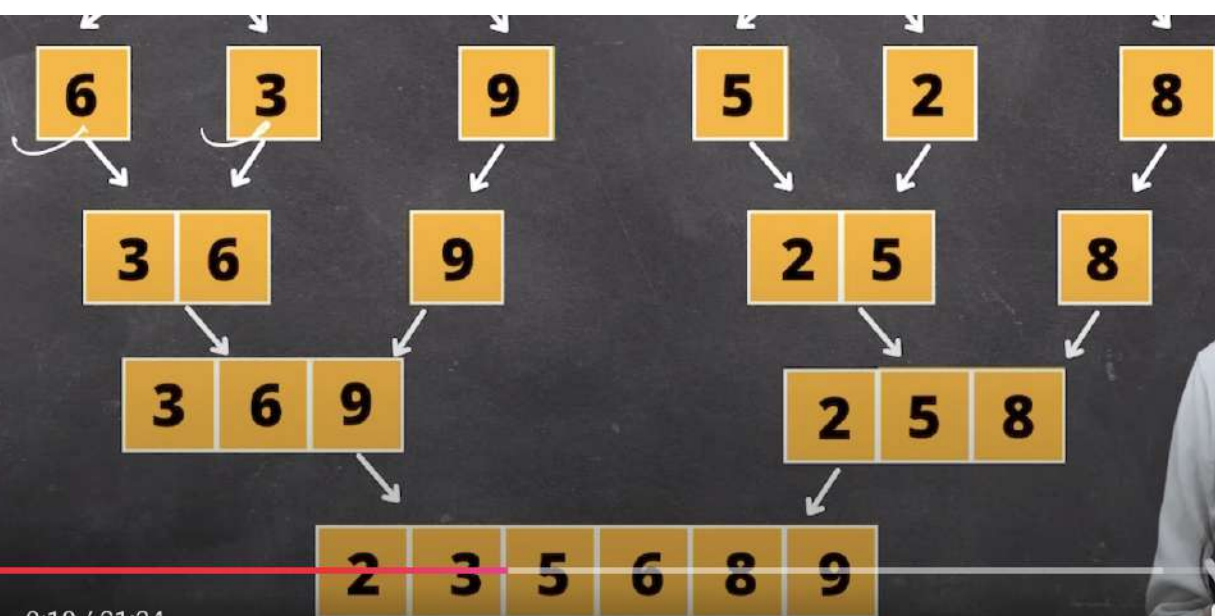
conquer =





Base Case

3/6  
n



9:10 / 21:24

```
public static void divide(int arr[], int si, int ei) {  
    if(si >= ei) {  
        return;  
    }  
  
    int mid = si + (ei-si)/2;  
    divide(arr, si, mid);  
    divide(arr, mid+1, ei);  
    conquer(arr, si, mid, ei);  
}
```

## Merge Sort

```
public static void conquer(int arr[], int si, int mid, int ei) {  
    int merged[] = new int[ei-si+1];  
    int idx1 = si;  
    int idx2 = mid+1;  
    int x = 0;  
    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++];  
    }  
  
    for(int i=0, j=si; i<merged.length; i++, j++) {  
        arr[j] = merged[i];  
    }  
}
```

**Time Complexity :  $O(n \log n)$**