

# **SNS COLLEGE OF ENGINEERING**

**Kurumbapalayam(Po), Coimbatore – 641 107**

**Accredited by NAAC-UGC with 'A' Grade**

**Approved by AICTE, Recognized by UGC & Affiliated to Anna University**

**Department of Artificial Intelligence and Data Science**

**Course Name: 23ITB201 Data structures and Algorithms**

**II Year / III semester**

**Unit III – Sorting, searching and hashing**

**Topic: Divide and conquer**

Divide and conquer algorithm is a strategy of solving a large problem by  
dividing the problem into smaller sub-problems  
solving the sub-problems, and  
combining them to get the desired output.  
In a divide and conquer algorithm, recursion is used.

# Divide and Conquer Algorithms Work?

## Steps involved:

1. Divide the given problem into sub-problems using recursion.

2. Solve the smaller sub-problems recursively. If the subproblem is small enough, solve it directly.

3. Combine the solutions of the sub-problems that are part of the recursive process to solve the actual problem.

4. Understand this concept with the help of an example.

5. Sort an array using the divide and conquer approach (ie. merge sort).

Sorting technique based on divide and conquer technique. With worst-case time complexity of  $O(n \log n)$ , it is one of the most used and approached algorithms.

It divides the array into equal halves and then combines them in a sorted manner.

### **How Merge Sort works?**

One of the most popular sorting algorithms known for its efficiency and stability. It follows the **divide and conquer** approach to sort a given array of elements.

Step-by-step explanation of how merge sort works:

1. Divide the list or array recursively into two halves until it can no more be divided.

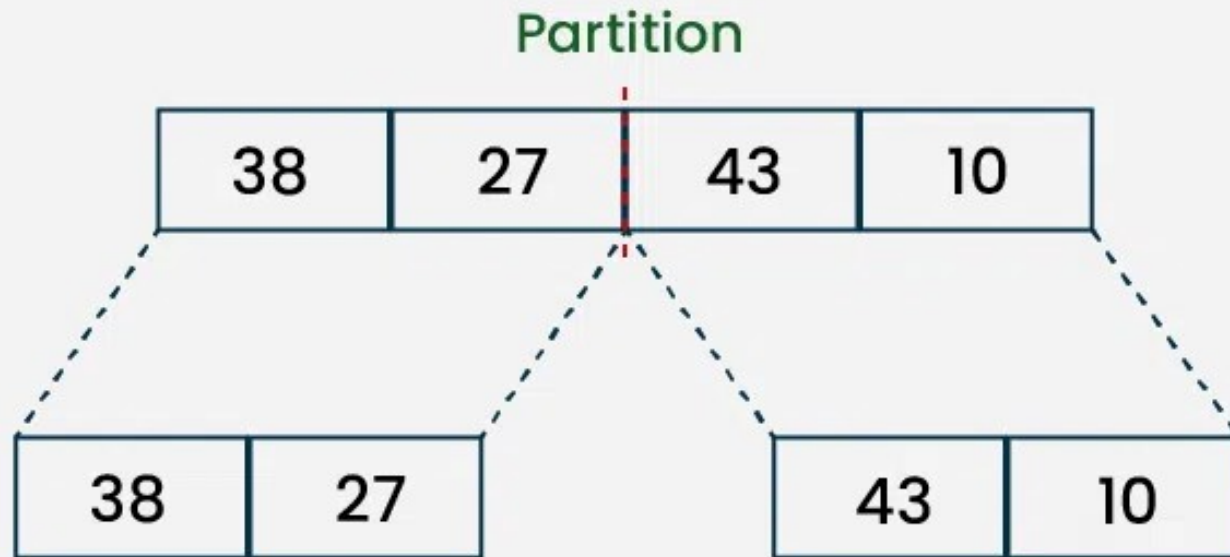
2. Recursively sort each subarray individually using the merge sort algorithm.

3. The sorted subarrays are merged back together in sorted order. The process continues until all elements from both subarrays have been merged.

Let's sort the array or list [38, 27, 43, 10] using Merge Sort

## Step 1

Splitting the Array into two equal halves



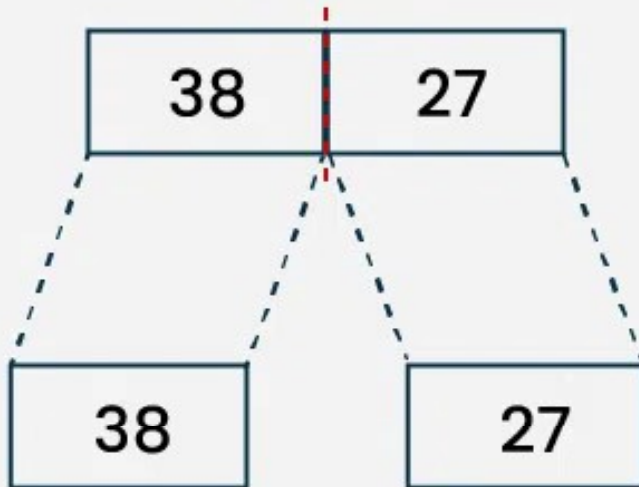
Let's sort the array or list [38, 27, 43, 10] using Merge Sort

## Step 2

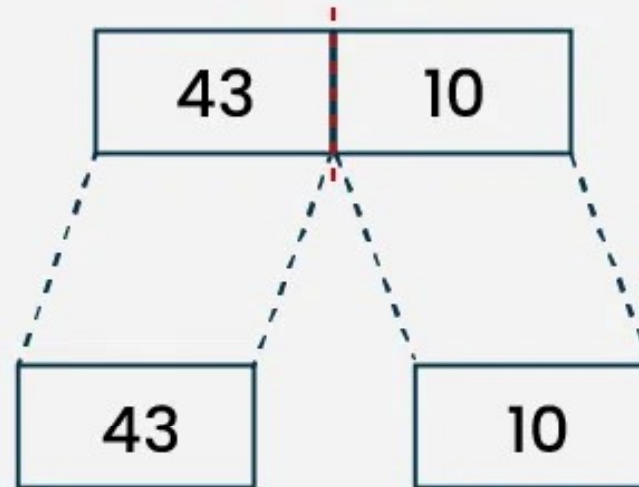
Splitting the subarrays into two halves



Partition



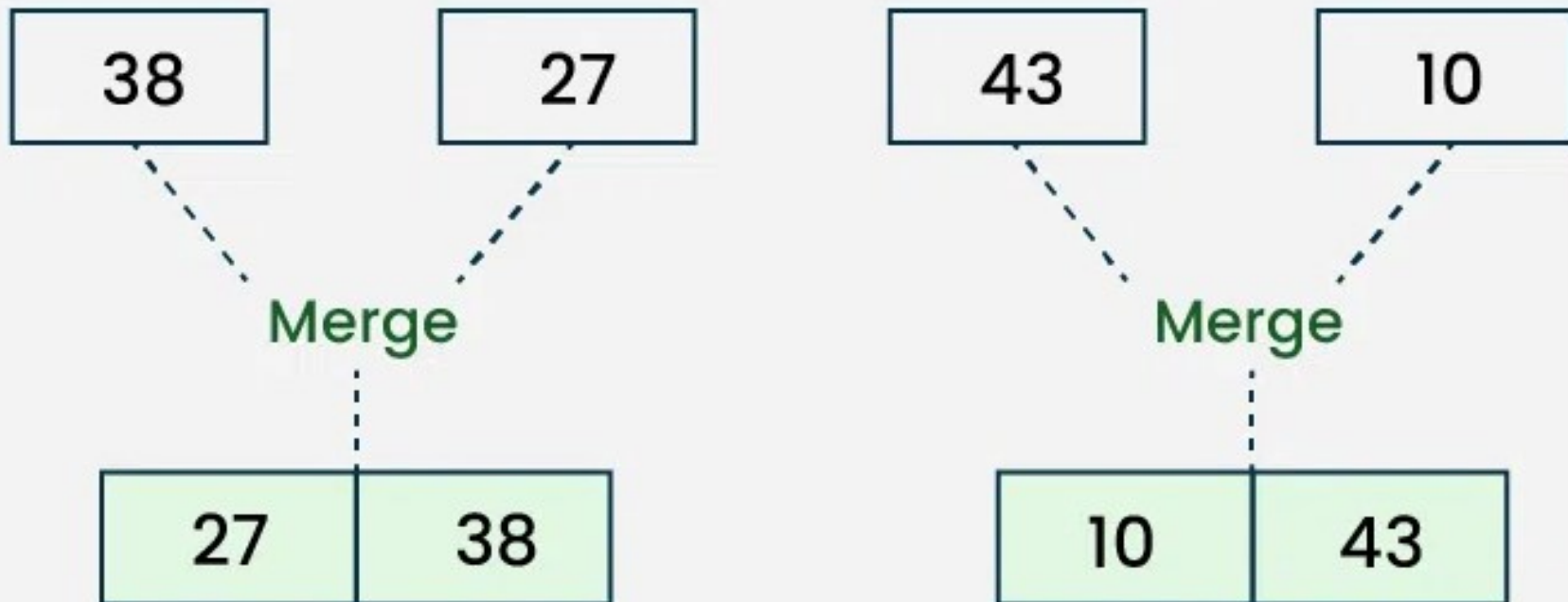
Partition



Let's sort the array or list [38, 27, 43, 10] using Merge Sort

### Step 3

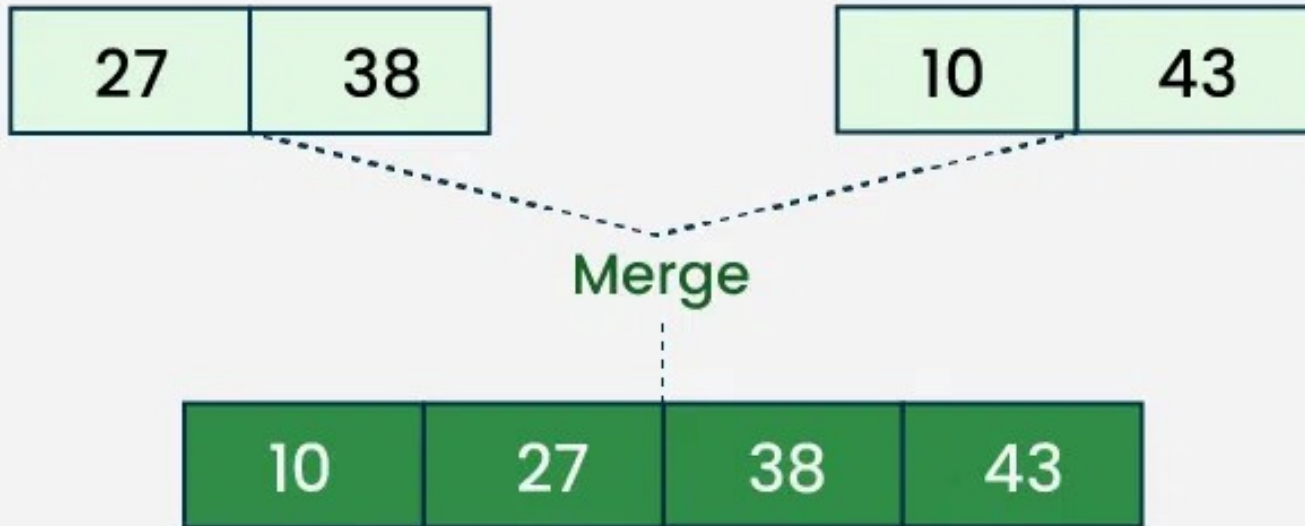
Merging unit length cells into sorted subarrays



Let's sort the array or list [38, 27, 43, 10] using Merge Sort

## Step 4

Merging sorted subarrays into the sorted array





**outline:**

```
rt(int A[], int low, int high){
```

```
igh){
```

```
= (low + high) /2;
```

```
Sort(A, low, mid);
```

```
Sort(A, mid+1, high);
```

```
Sort(A, mid, low, high);
```

•  
int mid, int low, int high)

& j <= high)

```
while (i <= mid)
{
    B[k] = A[i];
    k++;
    i++;
}
while (j <= high)
{
    B[k] = A[j];
    k++;
    j++;
}
for (int i = low; i <= high; i++)
{
    A[i] = B[i];
}
}
```