SNS COLLEGE OF ENGINEE

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna Univer

Department of Artificial Intelligence an

Course Name: 23ITB201 Data structures a

II Year / III semester

Unit III – Sorting, searching and ha

Topic: Divide and conquer

onquer algorithm is a strategy of solving a large problem by

the problem into smaller sub-problems

he sub-problems, and

g them to get the desired output.

de and conquer algorithm, recursion is used.

and Conquer Algorithms Work?

steps involved:

e the given problem into sub-problems using recursion.

ve the smaller sub-problems recursively. If the subproblem is s

solve it directly.

mbine the solutions of the sub-problems that are part of the rec

ve the actual problem.

tand this concept with the help of an example.

sort an array using the divide and conquer approach (ie. merg

orting technique based on divide and conquer technique. With worst-ca
g O(n log n), it is one of the most used and approached algorithms.

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

ge Sort work?

opular sorting algorithm known for its efficiency and stability. It follo
conquer approach to sort a given array of elements.

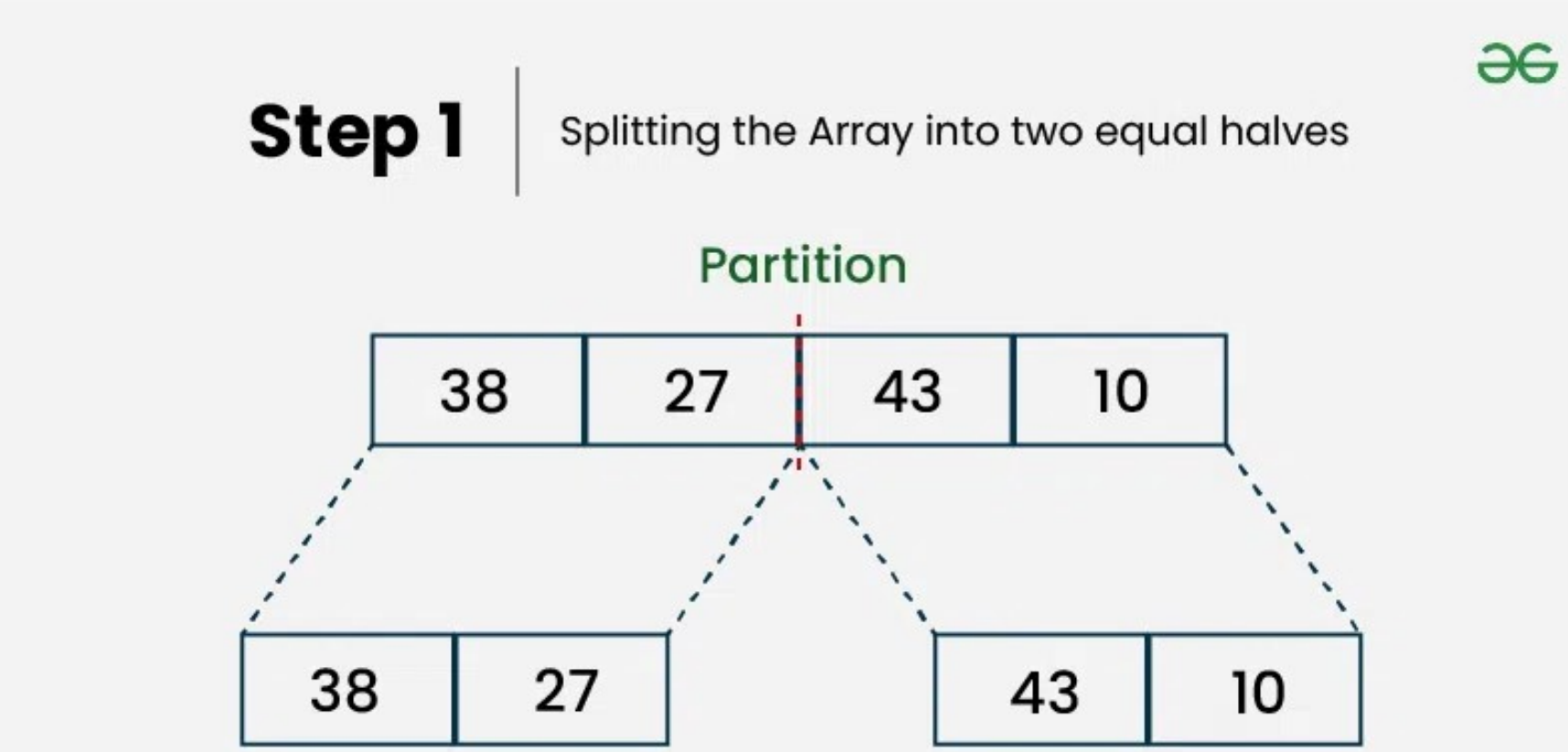
-step explanation of how merge sort works:

e the list or array recursively into two halves until it can no more be di

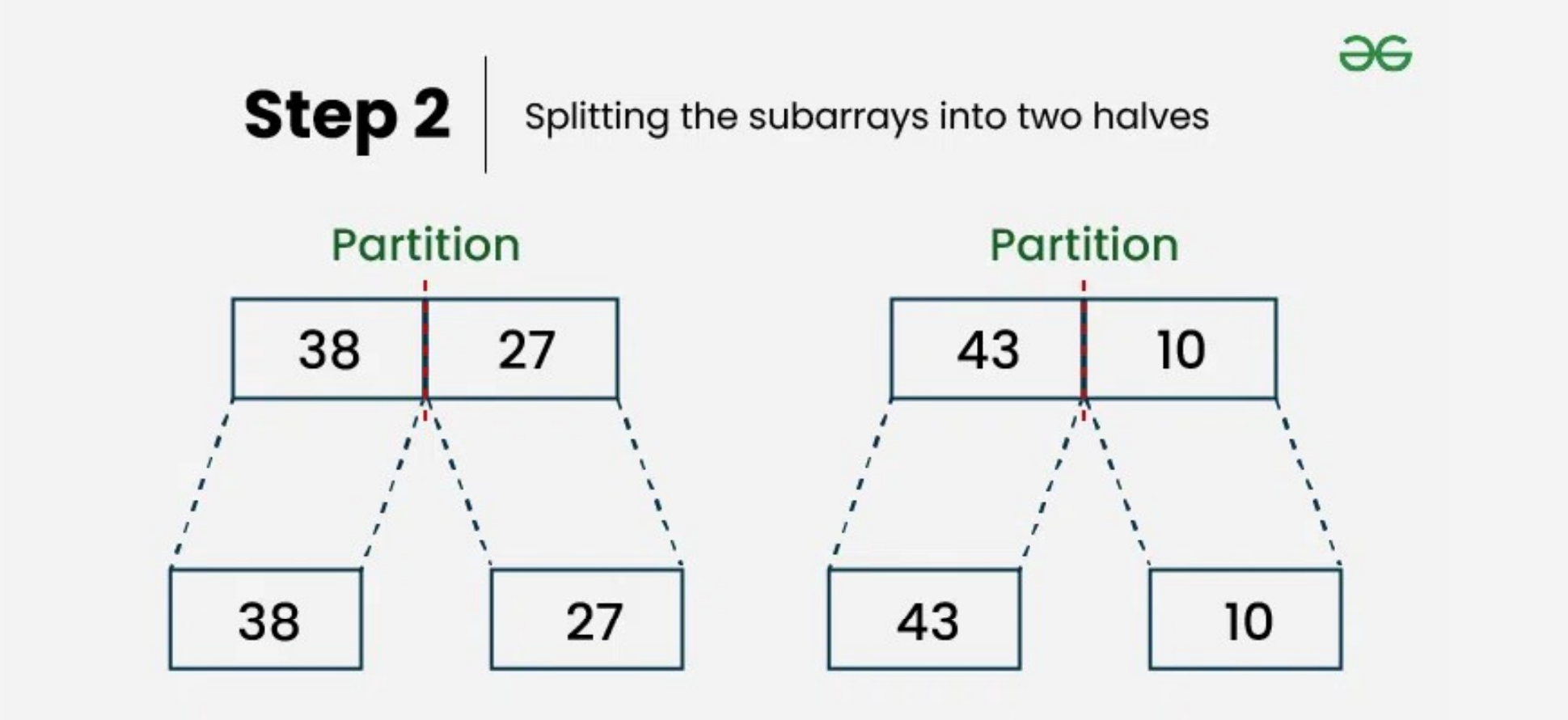
ch subarray is sorted individually using the merge sort algorithm.

orted subarrays are merged back together in sorted order. The process
ents from both subarrays have been merged.

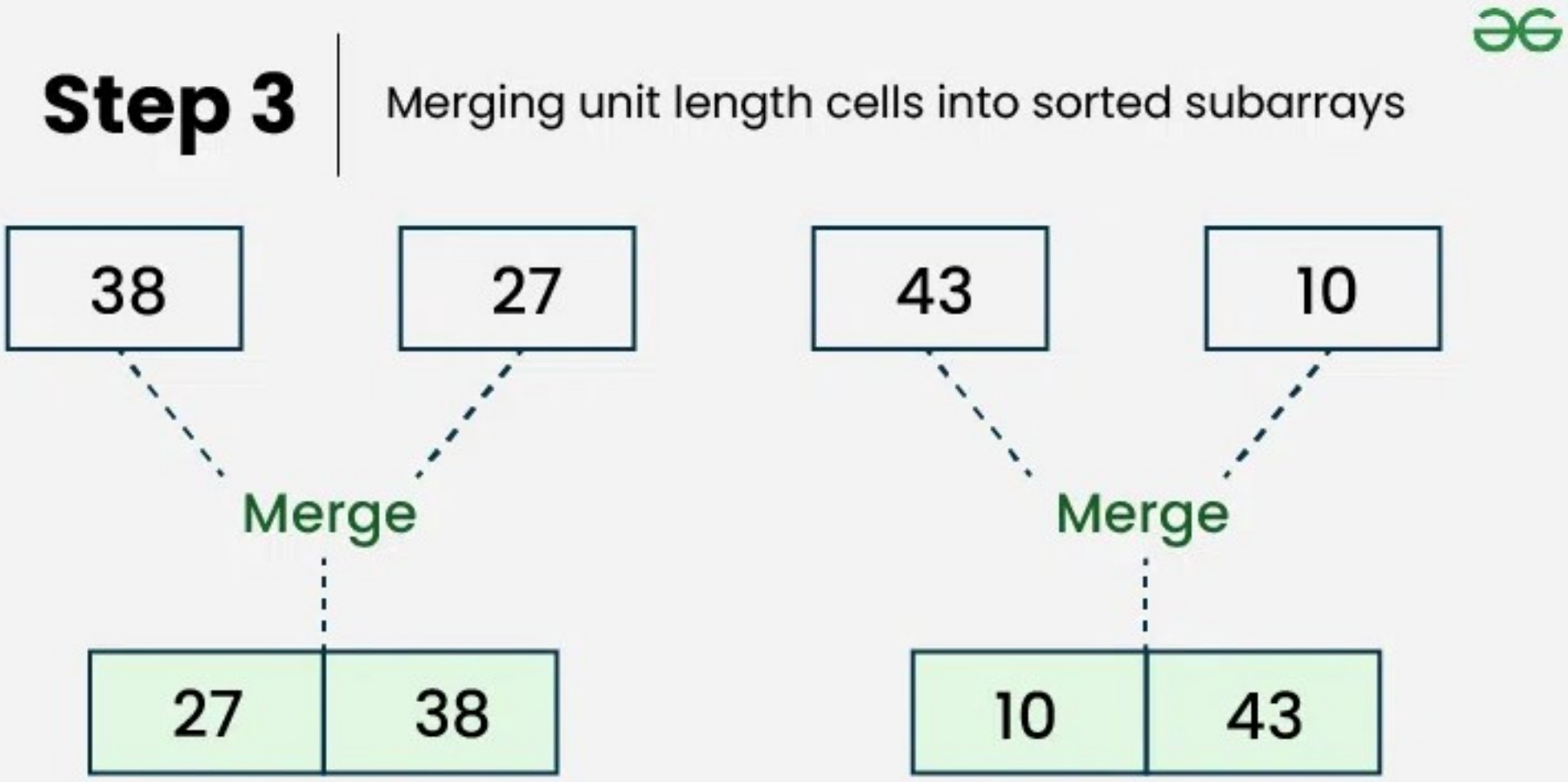
Let’s sort the array or list [38, 27, 43, 10] using Merge Sort



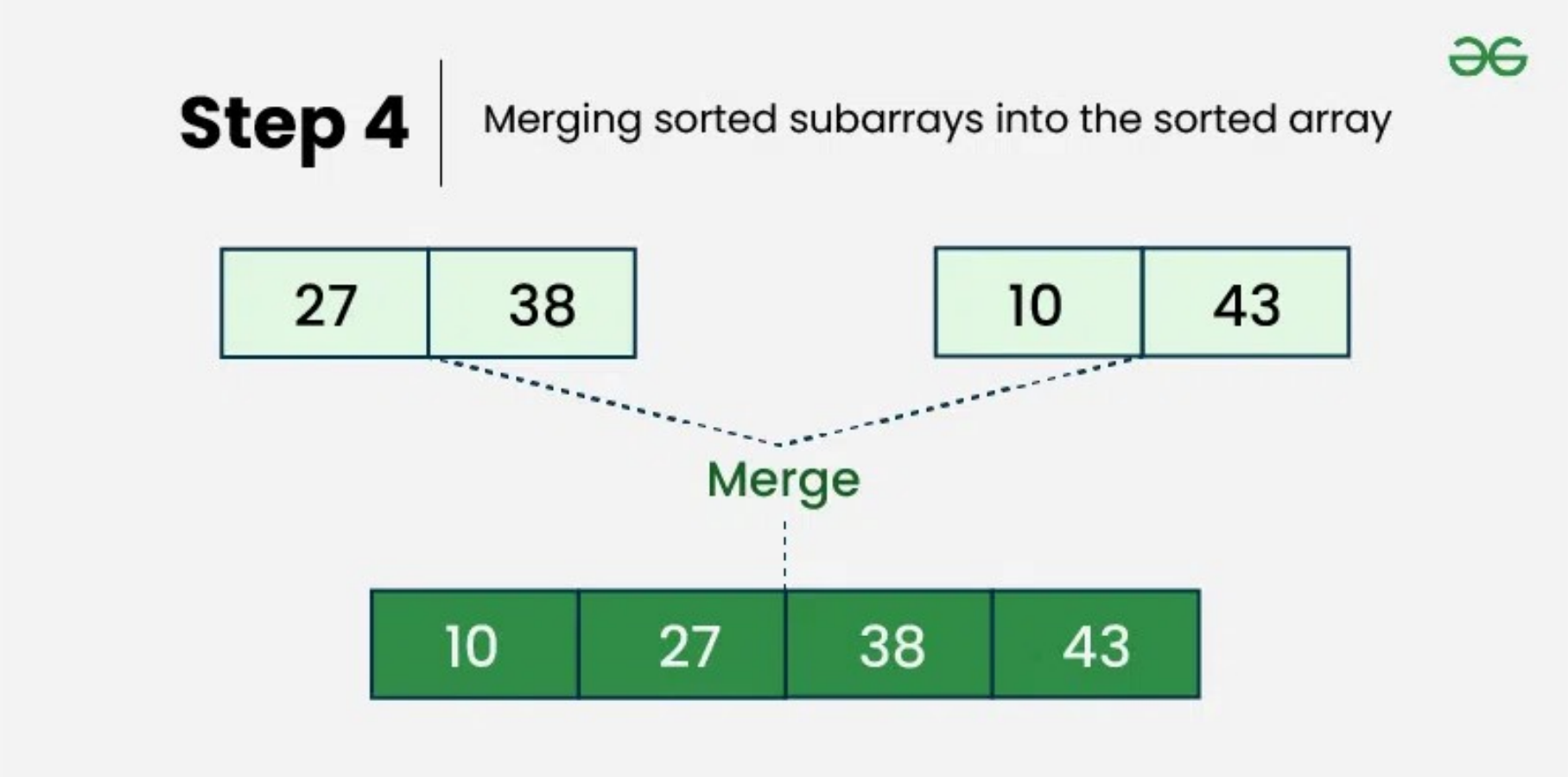
Let’s sort the array or list [38, 27, 43, 10] using Merge Sort



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



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



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


igh){
= (low + high) /2;
eSort(A, low, mid);
eSort(A, mid+1, high);
e(A, mid, low, high);
```

:

nt 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];
    }
}
```