Design and Analysis of Algorithms

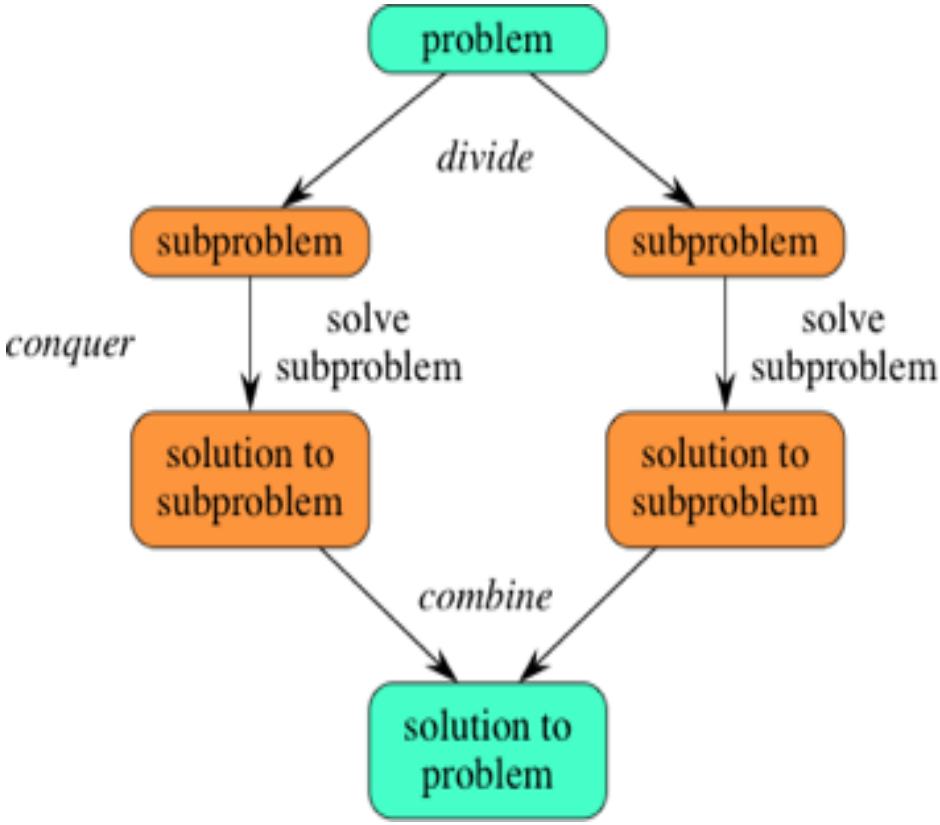
Dina El-Manakhly, Ph. D.

dina_almnakhly@science.suez.edu.eg

Divide and conquer

Divide and conquer strategy involves three steps :

1. **Divide** the given problem into sub-problems of same type. This step involves breaking the problem into smaller sub-problems. Sub-problems should represent a part of the original problem. This step generally takes a recursive approach to divide the problem until no sub-problem is further divisible.

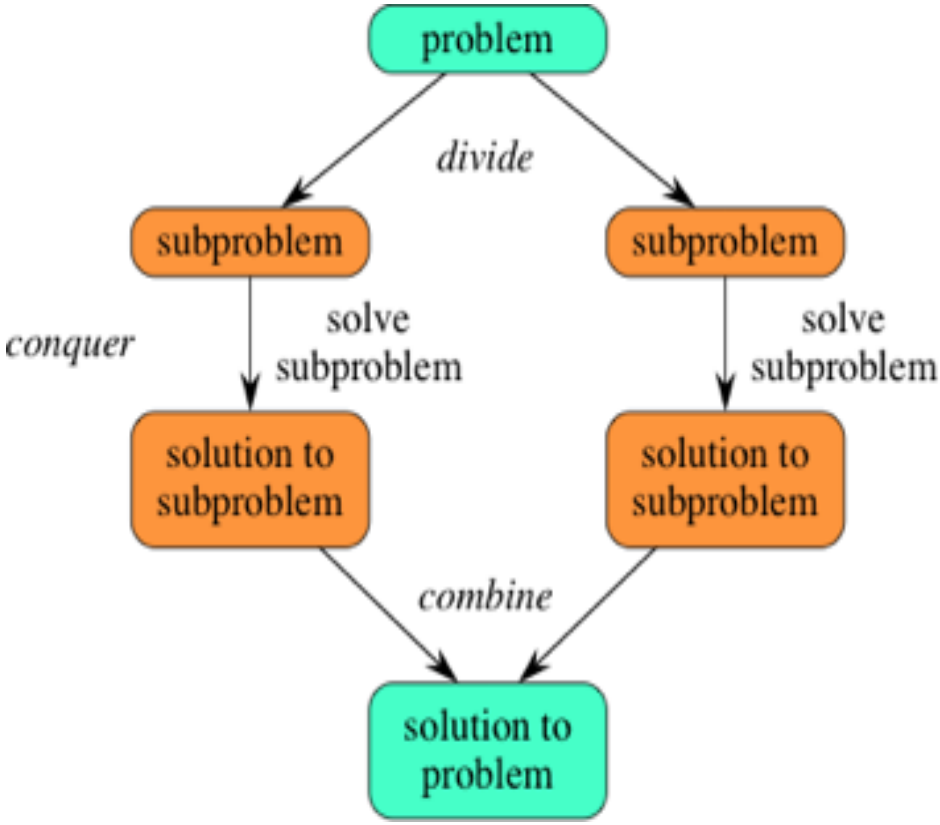


Divide and conquer

Divide and conquer strategy involves three steps :

2. Conquer the sub-problems by solving them recursively. If the sub-problem sizes are small enough, just solve the sub-problems in a straightforward manner.

3. Combine: Appropriately combine the answers. When the smaller sub-problems are solved, this stage recursively combines them until they formulate a solution of the original problem.



Binary Search

- **Problem Definition:** Given a sorted array $A=(a_1, a_2,\ldots, a_n)$ of n elements in non-decreasing order and an element k. Find the position of k in A, j, if $k=a_j$. Otherwise, return zero.

Examples

Example 1: Given A=(2,4,6,7,10,17,20) and k=7 then search(A,k)=4

Example 2: Given A=(2,4,6,7,10,17,20) and k=6 then search(A,k)=3

Example 3: Given A=(2,4,6,7,10,17,20) and k=9 then search(A,k)=0

Binary search can be implemented **only** on a **sorted list of items**. If the elements are not sorted already, we need to sort them first.

Binary Search

In this approach, the element is always searched in the middle of an array.

Main Idea:

- We compare a given element k with the middle element in the sorted array $A(1...n)$.

- $m=[(L+H)/2]$, L is the index of first element of A and H is the index of the last element of A.
- If $k = a_m$ then the element k is exist in the array A and return m.
- If $k < a_m$ then we discard $A(m...H)$ and we repeat the same process on $A(L...m\text{-}1)$. Similarly,
- if $k > a_m$ then we discard $A(L...m)$ and we repeat the same process on $A(m+1..H)$.

Binary Search Implementation

- Binary Search Algorithm can be implemented in two ways:

1. Iterative Method
2. Recursive Method

Iteration Method

Do until the pointers low and high meet each other.

```
int binarySearch(int arr[], int l, int h, int k)
{
    while (l <= h)
    {
        int m = (l+h)/2;

        if (k==arr[m]) // Check if k is present at mid
            return m;

        if (k>arr[m]) // If k greater, ignore left half
            l = m + 1;

        else // If k is smaller, ignore right half
            h = m - 1;
    }

    return -1; // If we reach here, then element was not present
}
```

Pseudo Code

Recursion

Algorithm: BinarySearch(A(L...H),k)

Begin

if $L > H$ then return 0

else

$m = \lfloor (L+H)/2 \rfloor$

if $k = a_m$ then return m

else if $k < a_m$ then return BinarySearch($A(L.....m\text{-}1),k$)

else return BinarySearch($A(m+1.....H),k$)

End.

Recursive Method

```
int binarySearch(int arr[], int l, int h, int k)
{
    if(l>h)
        return -1; // If we reach here, then element was not present
    else
    {
        int m = (l+h)/2;

        if (k==arr[m]) // Check if k is present at mid
            return m;

        if (k>arr[m]) // If k greater, Search the right half and ignore left half
            return binarySearch(arr,m + 1,h,k);

        else // If k is smaller, // Search the left half and ignore right half
            return binarySearch(arr,l,m-1,k);
    }
}
```

Some standard algorithms that follow Divide and Conquer algorithm

- Binary Search
- Merge Sort
- Quick Sort
- Closest Pair of Points
- Strassen's Algorithm (matrix multiplication)
- Finding maximum and minimum

MergeSort Algorithm

- **MergeSort** is one of the most popular sorting algorithms that is based on the principle of Divide and Conquer Algorithm.
- Here, a problem is divided into multiple sub-problems. Each sub-problem is solved individually. Finally, sub-problems are combined to form the final solution.
- **Definition**

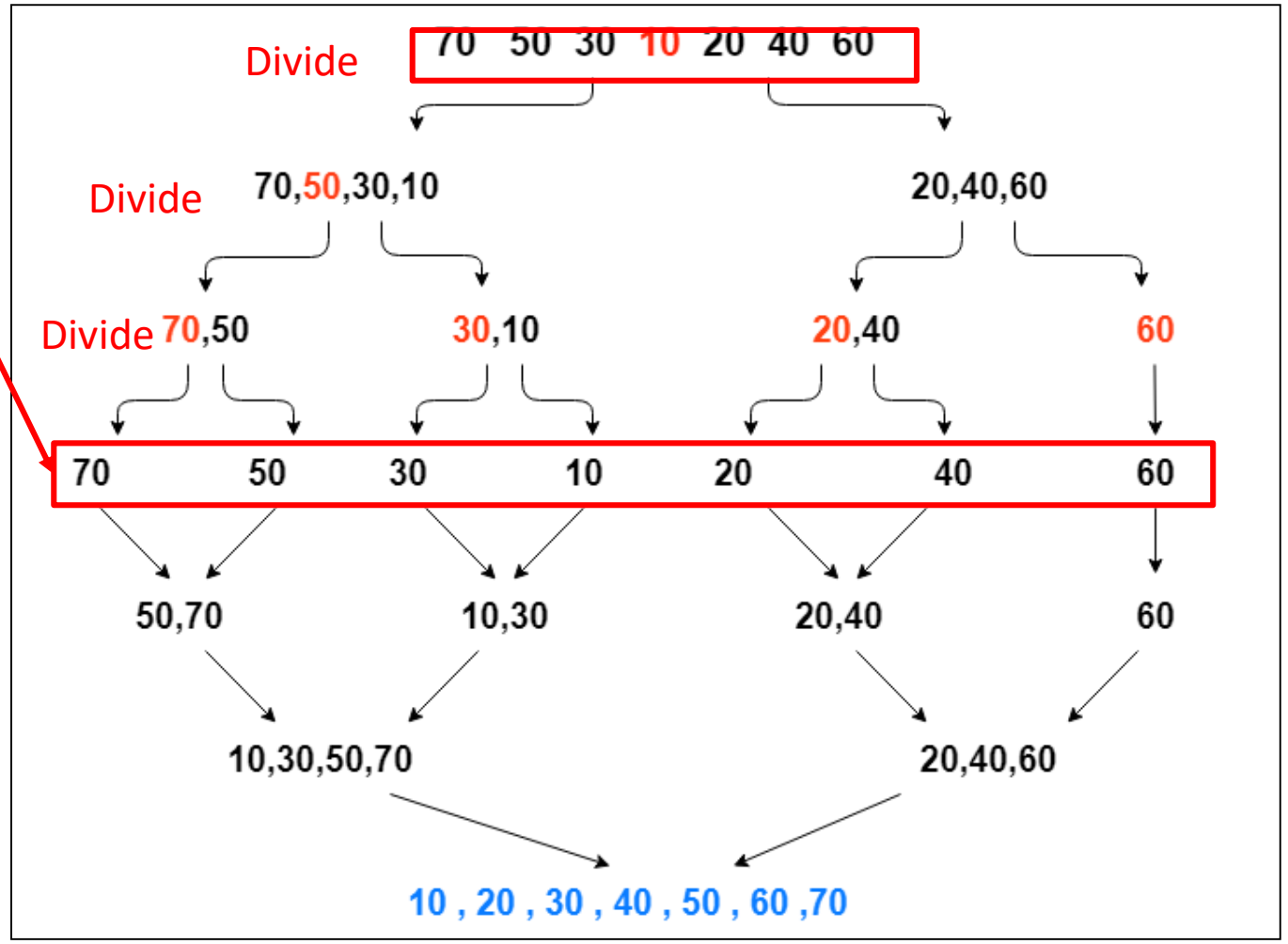
Problem Definition: Given an array A=(a_1, a_2,…, a_n) of n elements. Sorting the array is rearrangement the elements of the array such that $a_i \le a_{i+1}$, $1 \le i \le n-1$.

Example 1: Given A=(20,4,10,17,6,7,2)

Goal A=(2,4,6,7,10,17,20)

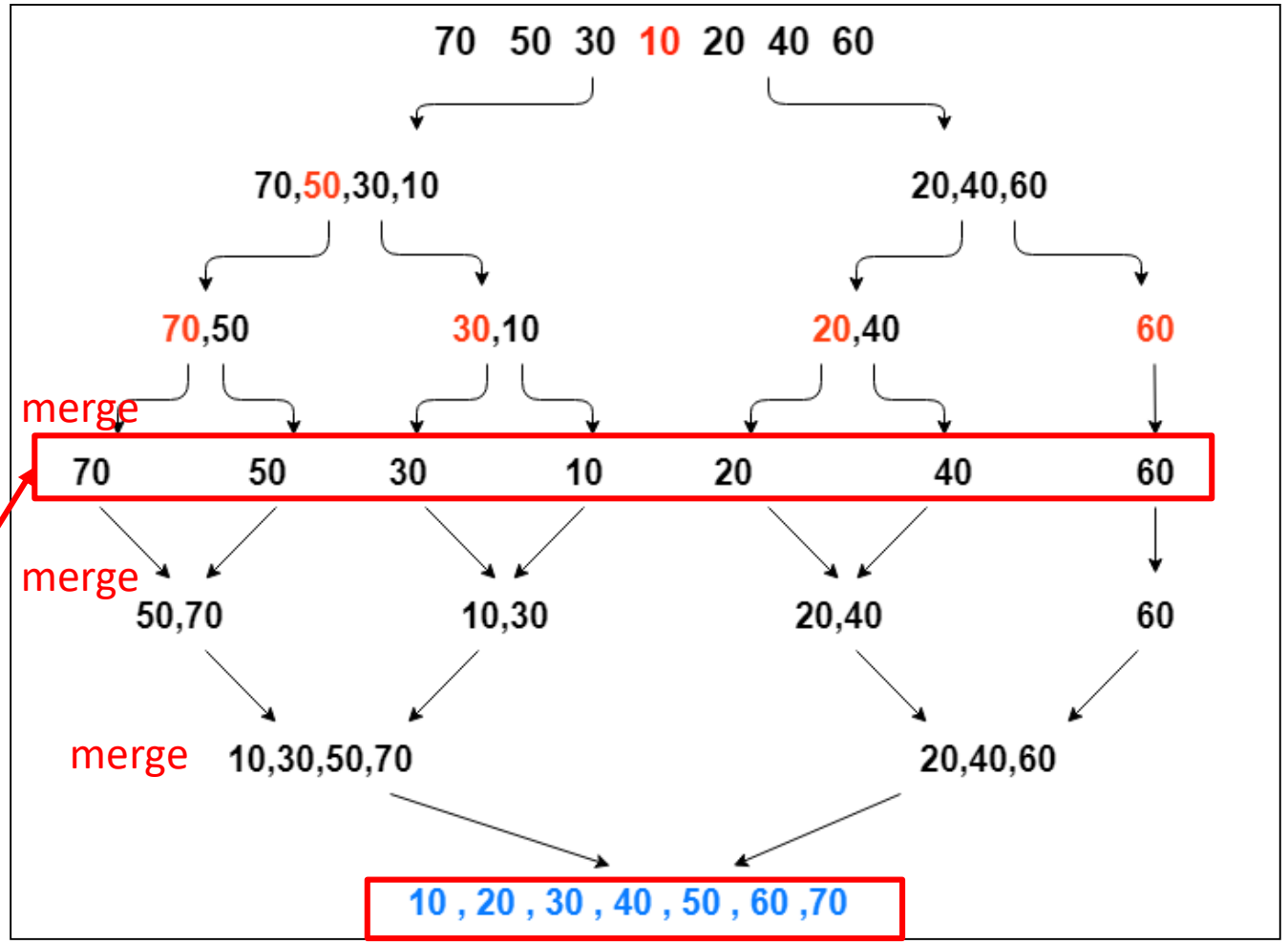
How MergeSort Works

- **MergeSort** works in the following way:
  - Initially divide the array into two equal halves, These subarrays are further divided into two halves. Now they become array of unit length that can no longer be divided and array of unit length are always sorted.



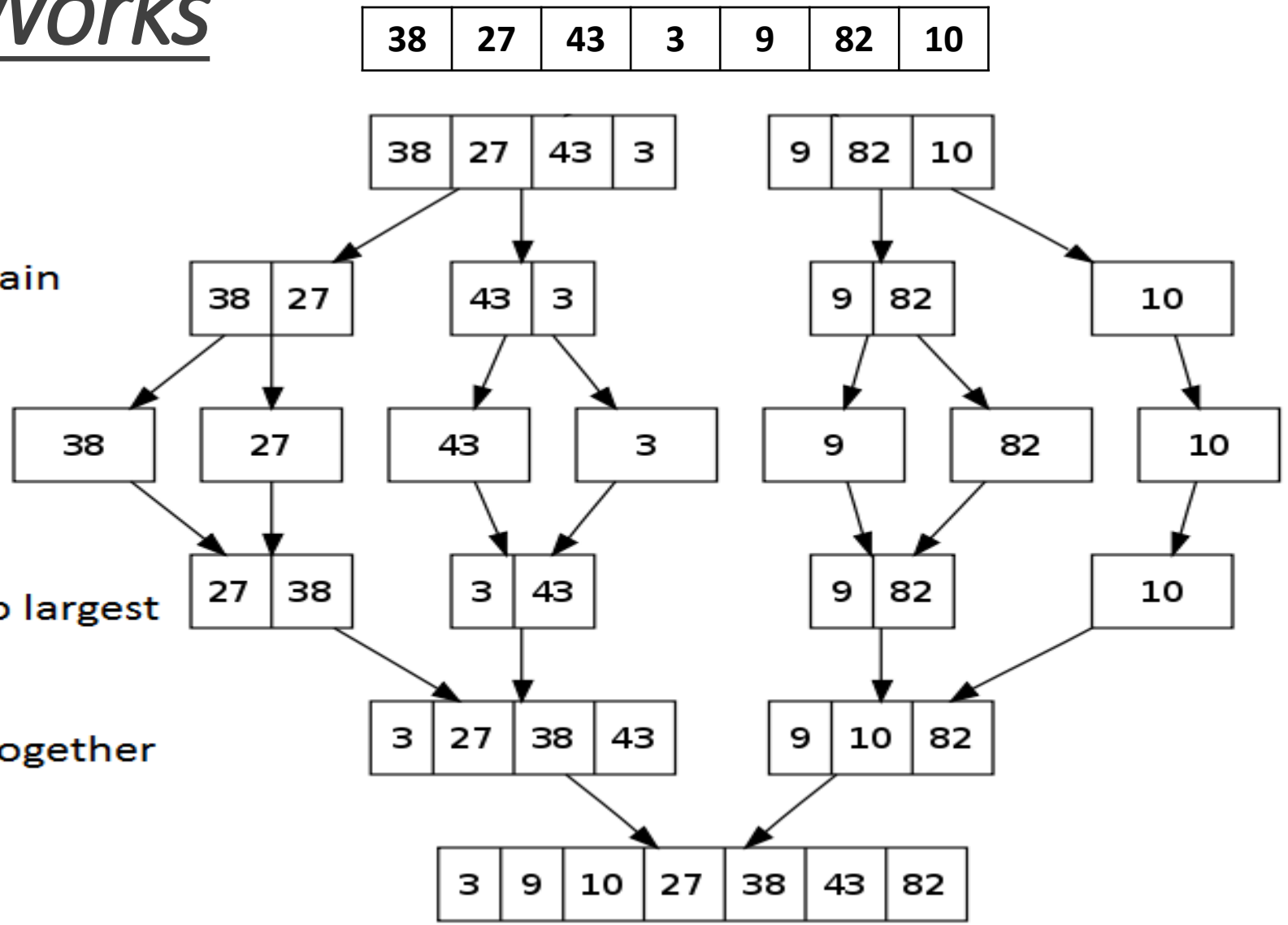
How MergeSort Works

- **MergeSort** works in the following way:
  - These sorted subarrays are merged together, and we get bigger sorted subarrays. This merging process is continued until the sorted array is built from the smaller subarrays.



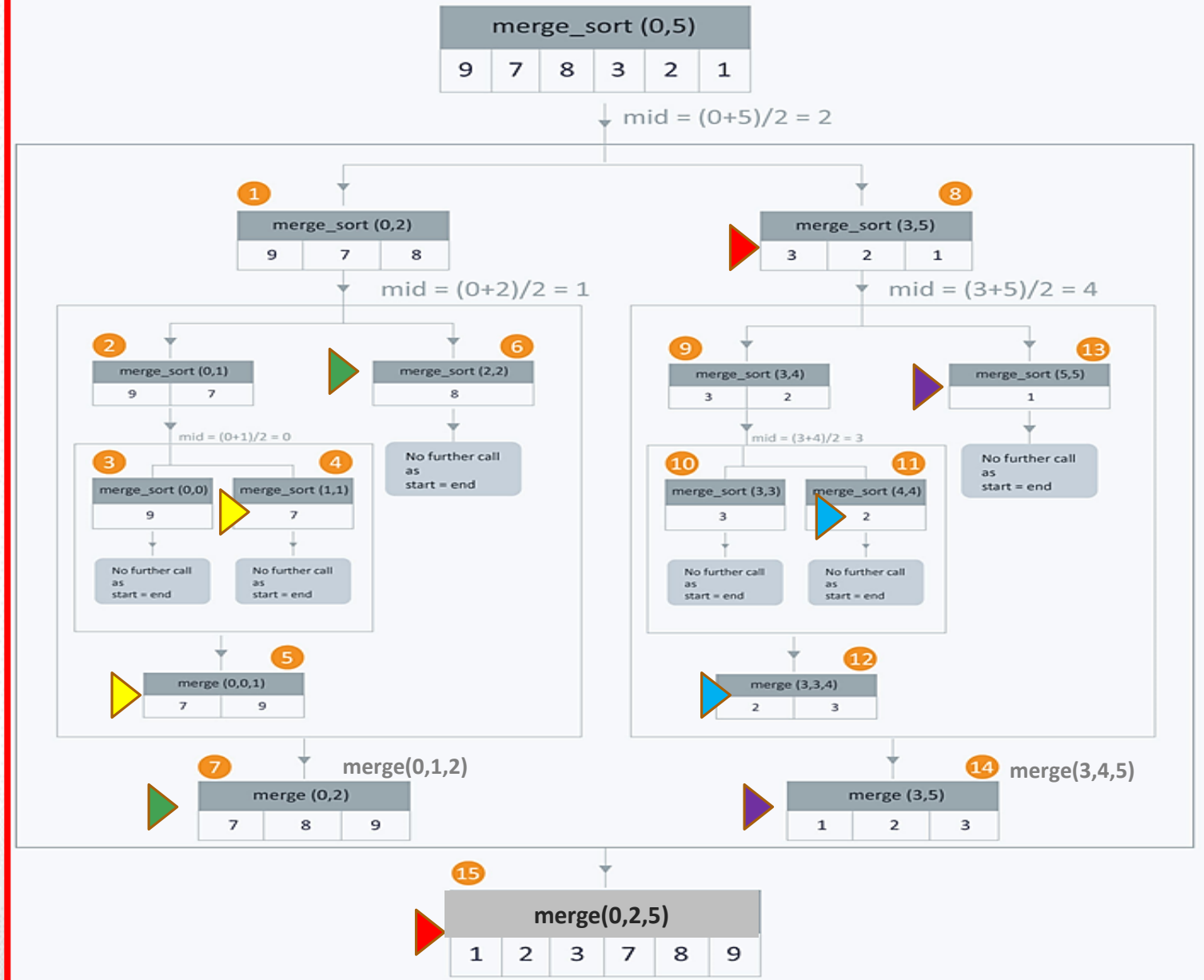
How Merge Sort Works

1. Divide the array into two parts
2. Divide the array into two parts again
3. Break each element into single parts
4. Sort the elements from smallest to largest
5. Merge the divided sorted arrays together
6. The array has been sorted

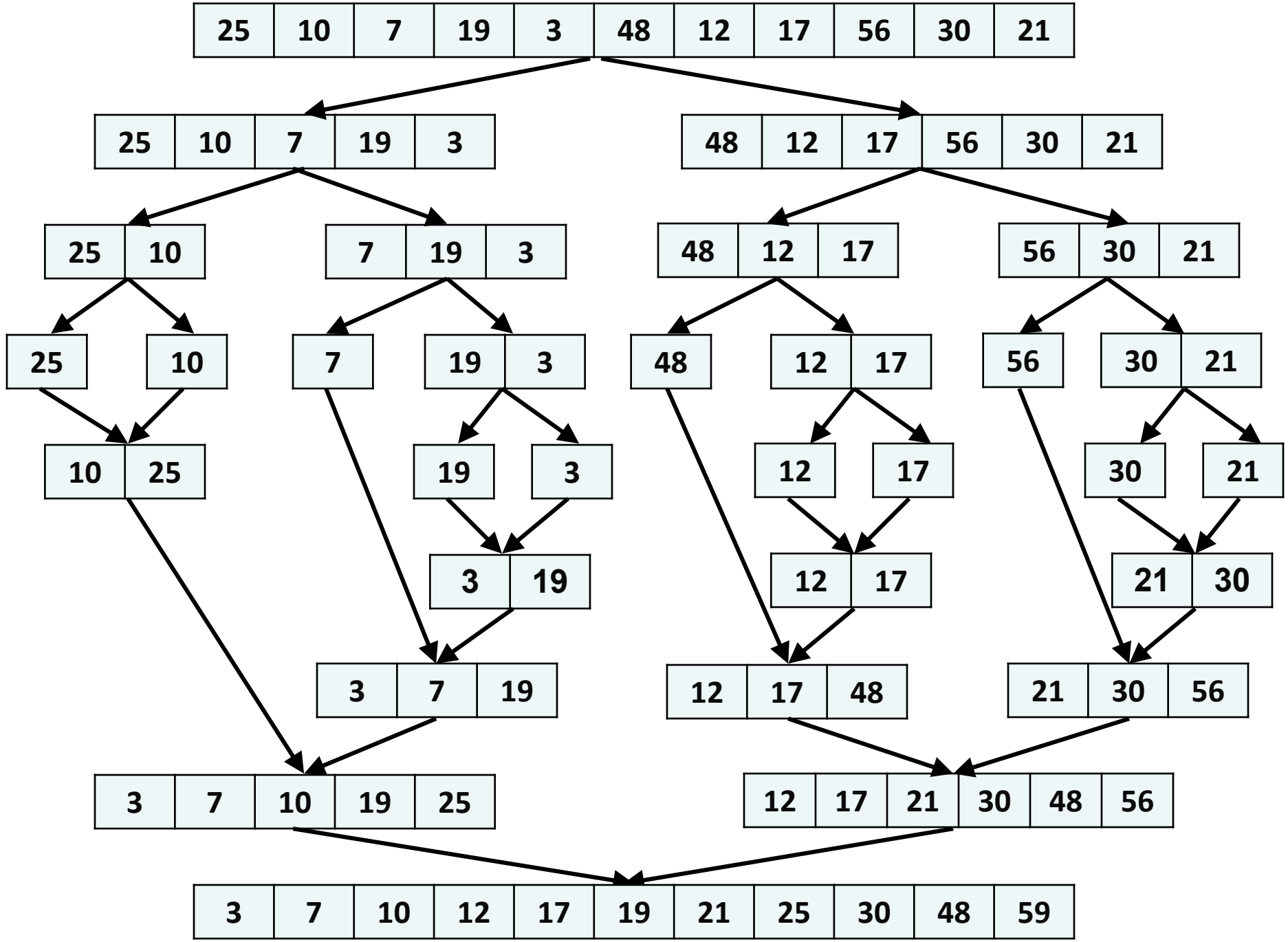


```
// merge sort sorting algorithm

// merge function
void merge(int arr[], int l, int m, int r)
{

}
```

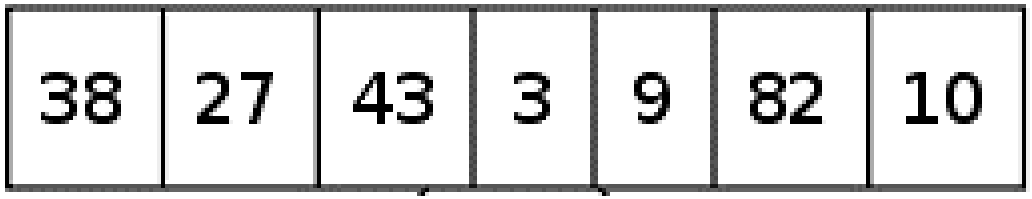
```
// merge sort function
void mergeSort(int arr[],int l, int r)
{
    if(l<r)
    {
        int m = (l+r)/2;
        mergeSort(arr,l,m);
        mergeSort(arr,m+1,r);
        merge(arr,l,m,r);
    }
}
```



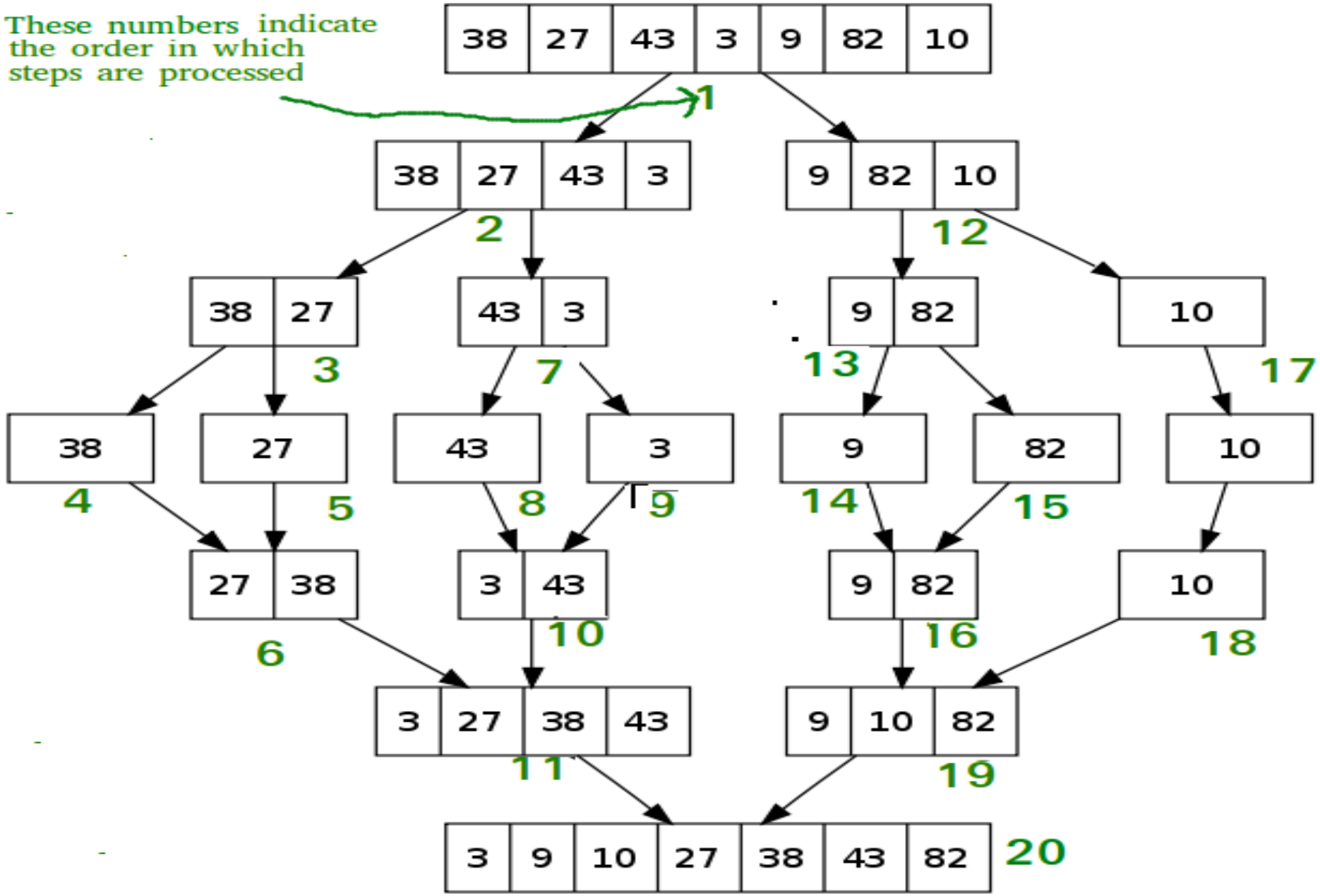
Tracing Merge Sort Algorithm


25 10 7 19 3 48 12 17 56 30 21
25 10 7 19 3
48 12 17 56 30 21
25 10
7 19 3
48 12 17
56 30 21
25
10
7
19 3
48
12 17
56
30 21
10 25
19 3
12 17
30 21
3 19
12 17
21 30
3 7 19
12 17 48
21 30 56
3 7 10 19 25
12 17 21 30 48 56
3 7 10 12 17 19 21 25 30 48 59

Indicate the order in which steps are processed in merge sort algorithm



Answer


These numbers indicate the order in which steps are processed
38 27 43 3 9 82 10
1
38 27 43 3
9 82 10
2
12
38 27
43 3
9 82
10
3
7
13
17
38
27
43
3
9
82
10
4
5
8
9
14
15
27 38
3 43
9 82
10
6
10
16
18
3 27 38 43
9 10 82
11
19
3 9 10 27 38 43 82
20

Merge Sort Complexity

Best	**O(n*log n)**
Worst	**O(n*log n)**
Average	**O(n*log n)**
Space Complexity	**O(n)**

Some standard algorithms that follow Divide and Conquer algorithm

- Binary Search
- Merge Sort
- Quick Sort
- Closest Pair of Points
- Strassen's Algorithm (matrix multiplication)
- Finding maximum and minimum