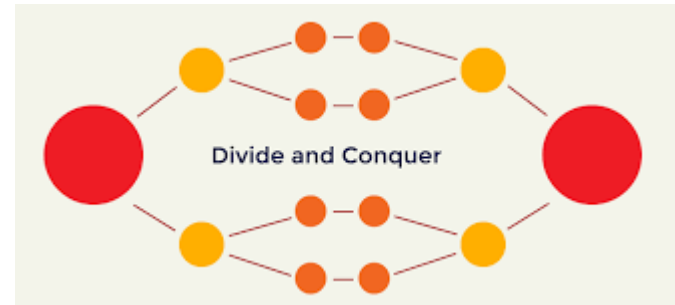
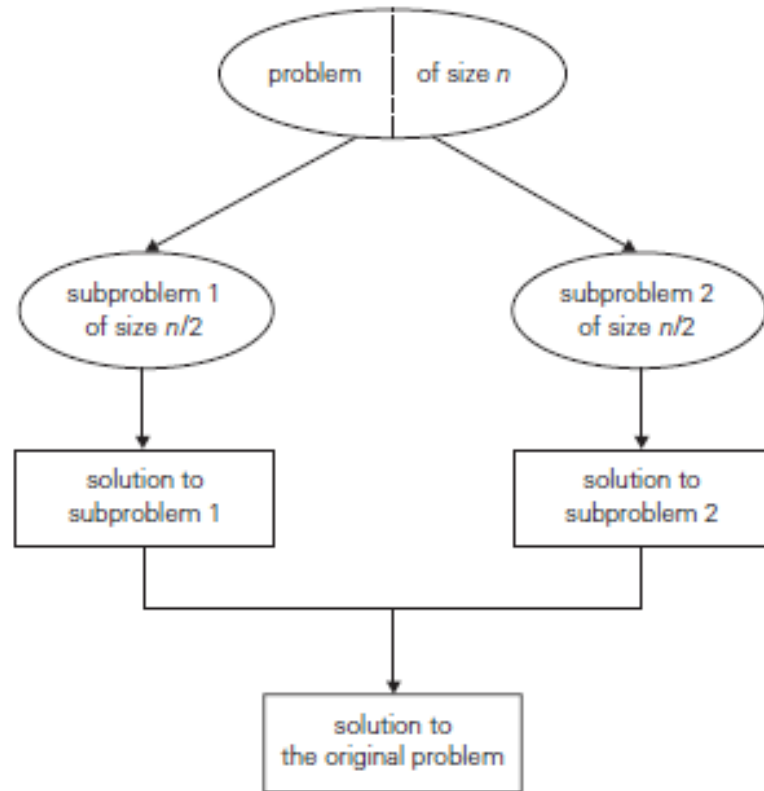


# Unit II – Divide and Conquer

- **Merge sort**
- Quick sort
- Binary search
- Multiplication of large Integers
- Strassen's Matrix Multiplication

# Divide and Conquer Design Technique



# Divide and Conquer Design Technique

1. A problem is divided into several **sub problems** of the same type, ideally of about equal size.
2. The sub problems are **solved** (typically *recursively*, though sometimes a different algorithm is employed, especially when sub problems become small enough).
3. If necessary, the solutions to the **sub problems are combined** to get a solution to the original problem.

## Algorithm for Divide and Conquer:

```
DAC(P)  
if small(P)  
    S(P)  
else  
    Divide P into P1,P2.....Pn  
    Apply DAC(P1), DAC(P2).....DAC(Pn)  
    S(DAC(P1), DAC(P2).....DAC(Pn))
```

# Divide and Conquer Design Technique

## Recurrence Relation

$$T(n) = a T(n/b) + f(n)$$

here

$T(n/b)$  – sub problem

$f(n)$  – time spent for dividing  $n$  into  $n/b$  and combining their solutions

## Masters Theorem

$$T(n) = a T(n/b) + f(n) \quad a \geq 1, b > 1, f(n) = O(n^k \log^p n)$$

Find values: 1.  $\log_b a$

2.  $k$

Case 1 : if  $\log_b a > k$ , then  $O(n^{\log_b a})$

Case 2: if  $\log_b a = k$ , then  $O(n^k \log^p n \log n)$

Case 3: if  $\log_b a < k$ , then  $O(n^k \log^p n)$

- **Masters Theorem - Example**

- $T(n) = a T(n/b) + f(n), f(n) = O(n^k \log^p n)$

- $T(n) = 2 T(n/2) + 1$

- Here  $a = 2, b = 2, f(n) = 1 = O(1) = O(n^0 \log^0 n)$

- From this  $k = 0, p = 0, a=2, b=2$

Find values: 1.  $\log_b a = \log_2 2 = 1$

2.  $k = 0$

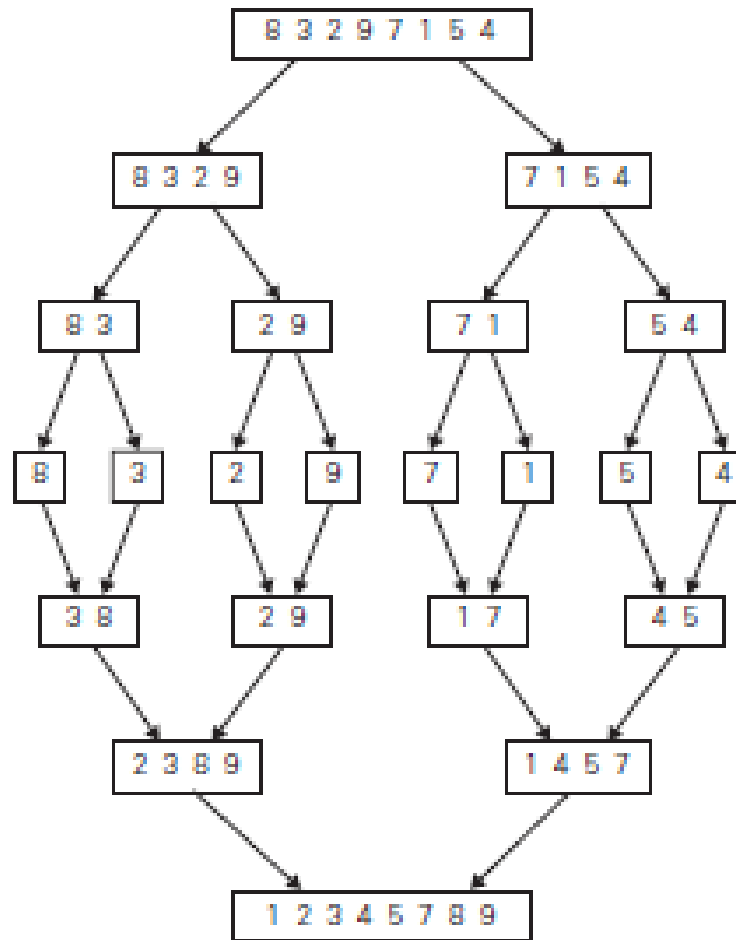
Case 1:  $\log_b a > k \rightarrow 1 > 0$

$O(n^{\log_b a})$

$O(n^1)$

# MERGE SORT - Example

- [Link](#)
- Example



# MERGE SORT - Algorithm

**ALGORITHM** *Mergesort*( $A[0..n - 1]$ )  $\longrightarrow$   $T(n)$   
//Sorts array  $A[0..n - 1]$  by recursive mergesort  
//Input: An array  $A[0..n - 1]$  of orderable elements  
//Output: Array  $A[0..n - 1]$  sorted in nondecreasing order  
**if**  $n > 1$   
    copy  $A[0..\lfloor n/2 \rfloor - 1]$  to  $B[0..\lfloor n/2 \rfloor - 1]$   
    copy  $A[\lfloor n/2 \rfloor..n - 1]$  to  $C[0..\lfloor n/2 \rfloor - 1]$   
    *Mergesort*( $B[0..\lfloor n/2 \rfloor - 1]$ )  $\longrightarrow$   $T(n/2)$   
    *Mergesort*( $C[0..\lfloor n/2 \rfloor - 1]$ )  $\longrightarrow$   $T(n/2)$   
    *Merge*( $B, C, A$ ) //see below  $\longrightarrow$   $n$

**ALGORITHM** *Merge*( $B[0..p - 1], C[0..q - 1], A[0..p + q - 1]$ )  
//Merges two sorted arrays into one sorted array  
//Input: Arrays  $B[0..p - 1]$  and  $C[0..q - 1]$  both sorted  
//Output: Sorted array  $A[0..p + q - 1]$  of the elements of  $B$  and  $C$   
 $i \leftarrow 0; j \leftarrow 0; k \leftarrow 0$   
**while**  $i < p$  and  $j < q$  **do**  
    **if**  $B[i] \leq C[j]$   
         $A[k] \leftarrow B[i]; i \leftarrow i + 1$   
    **else**  $A[k] \leftarrow C[j]; j \leftarrow j + 1$   
     $k \leftarrow k + 1$   
**if**  $i = p$   
    copy  $C[j..q - 1]$  to  $A[k..p + q - 1]$   
**else** copy  $B[i..p - 1]$  to  $A[k..p + q - 1]$

# MERGE SORT - Analysis

- $T(n) = 1$                        $n=1$
- $= 2 T(n/2) + n$        $n > 1$
- Here  $a=b=2$ ,  $f(n) = n$
- 2 values
  - $\text{Log}_b a = \log_2 2 = 1$
  - $K \rightarrow n^k = n^1$
- $\log_b a = k \rightarrow 1 = 1 \rightarrow \text{case 2} \rightarrow O(n \log n)$