UNIT 2

DIVIDE AND CONQUER STRATEGY

INTRODUCTION

- Many algorithms are recursive in nature to solve a given problem recursively dealing with sub-problems.

- In divide and conquer approach, a problem is divided into smaller problems, then the smaller problems are solved independently, and finally the solutions of smaller problems are combined into a solution for the large problem.

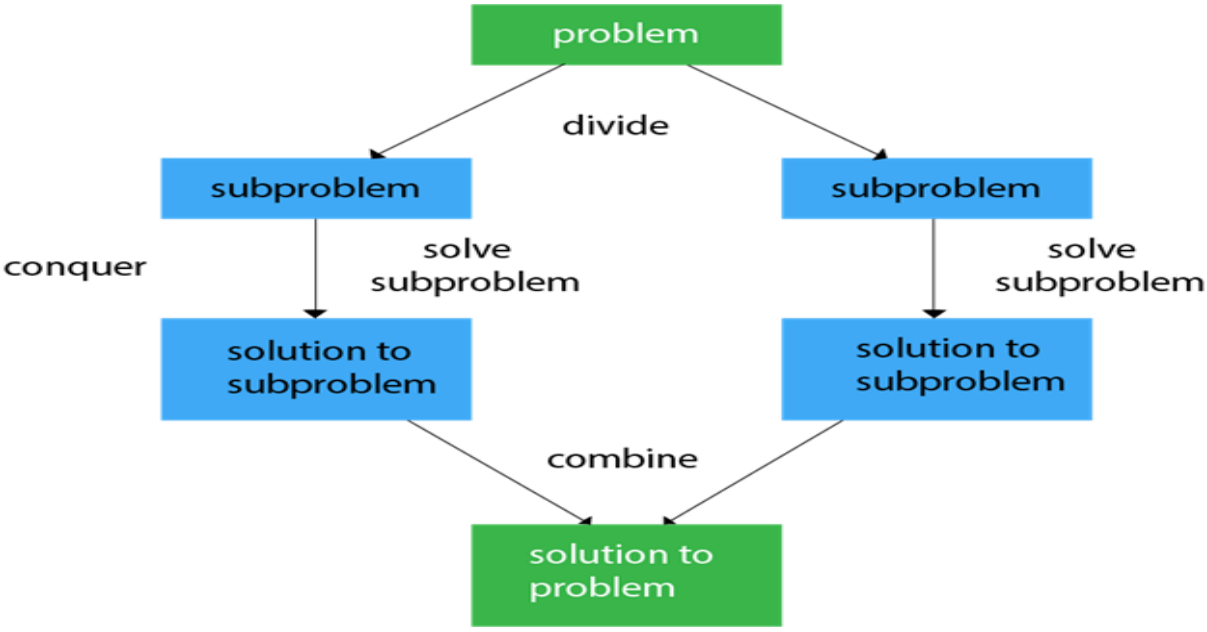
- Generally, divide-and-conquer algorithms have three parts –

- **Divide:** Divide the problem into a number of sub-problems that are smaller instances of the same problem.
- **Conquer:** Conquer the sub-problems by solving them recursively. If they are small enough, solve the sub-problems as base cases.
- **Combine:** Combine the solutions to the sub-problems into the solution for the original problem.

Application of Divide and Conquer Approach

Following are some problems, which are solved using divide and conquer approach.

- Finding the maximum and minimum of a sequence of numbers
- Strassen's matrix multiplication
- Merge sort
- Binary search
- Quick Sort



Fundamental of Divide & Conquer Strategy:

There are two fundamental of Divide & Conquer Strategy:

1. Relational Formula
2. Stopping Condition

1. Relational Formula: It is the formula that we generate from the given technique. After generation of Formula we apply D&C Strategy, i.e. we break the problem recursively & solve the broken subproblems.

2. Stopping Condition: When we break the problem using Divide & Conquer Strategy, then we need to know that for how much time, we need to apply divide & Conquer. So the condition where the need to stop our recursion steps of D&C is called a Stopping Condition.

Advantages of Divide and Conquer

- Divide and Conquer tend to successfully solve one of the biggest problems, such as the Tower of Hanoi, a mathematical puzzle. It is challenging to solve complicated problems for which you have no basic idea, but with the help of the divide and conquer approach, it has lessened the effort as it

works on dividing the main problem into two halves and then solve them recursively. This algorithm is much faster than other algorithms.

- It efficiently uses cache memory without occupying much space because it solves simple subproblems within the cache memory instead of accessing the slower main memory.
- It is more proficient than that of its counterpart Brute Force technique.
- Since these algorithms encourage parallelism, it does not involve any modification and is handled by systems incorporating parallel processing.

Disadvantages of Divide and Conquer

- Since most of its algorithms are designed by incorporating recursion, so it necessitates high memory management.
- An explicit stack may overuse the space.
- It may even crash the system if the recursion is performed rigorously greater than the stack present in the CPU.

Time Complexity

The complexity of the divide and conquer algorithm is calculated using the master theorem.

$T(n) = aT(n/b) + f(n)$,

where,

n = size of input

a = number of subproblems in the recursion

n/b = size of each subproblem. All subproblems are assumed to have the same size.

f(n) = cost of the work done outside the recursive call, which includes the cost of dividing the problem and cost of merging the solutions