TEAM ID:	NM2023TMID04432
PROJECT NAME:	AGRICULTURE DOCS CHAIN

UTILIZATION OF ALGORITHMS, DYNAMIC PROGRAMMING, OPTIMAL MEMORY UTLIZATION

Dynamic programming is a technique that breaks the problems into sub-problems, and saves the result for future purposes so that we do not need to compute the result again.

The subproblems are optimized to optimize the overall solution is known as optimal substructure property. The main use of dynamic programming is to solve optimization problems.

Here, optimization problems mean that when we are trying to find out the minimum or the maximum solution of a problem.

The dynamic programming guarantees to find the optimal solution of a problem if the solution exists.

The definition of dynamic programming says that it is a technique for solving a complex problem by first breaking

into a collection of simpler subproblems, solving each subproblem just once, and then storing their solutions to avoid repetitive computations.

Let's understand this approach through an example.

Consider an example of the Fibonacci series. The following series is the Fibonacci series:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ,...

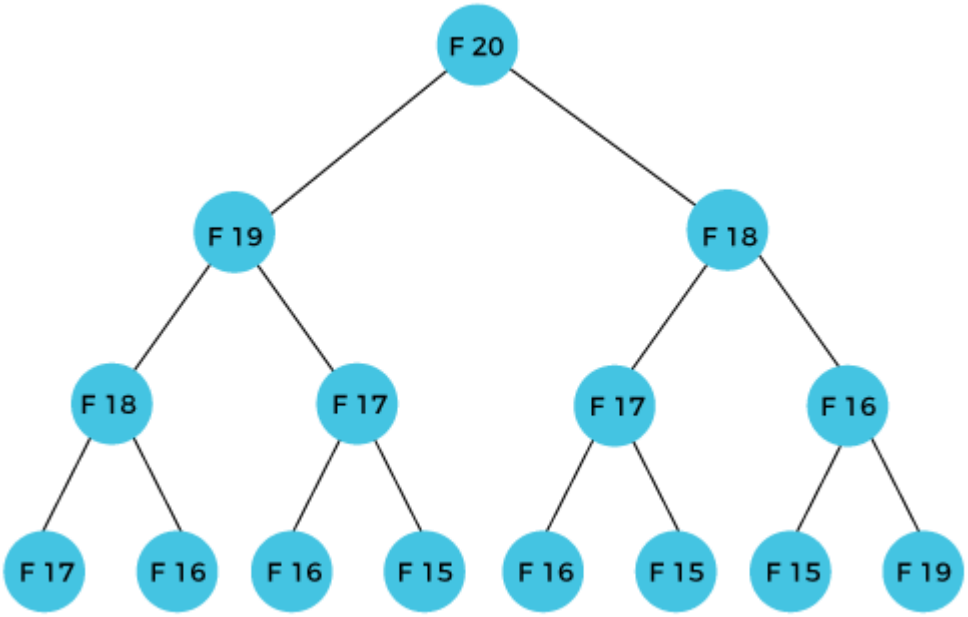
The numbers in the above series are not randomly calculated. Mathematically, we could write each of the terms using the below formula:

F(n) = F(n-1) + F(n-2),

With the base values F(0) = 0, and F(1) = 1. To calculate the other numbers, we follow the above relationship. For example, F(2) is the sum **f(0)** and **f(1),** which is equal to 1.

How can we calculate F(20)?

The F(20) term will be calculated using the nth formula of the Fibonacci series. The below figure shows that how F(20) is calculated.



Advantages

- It is very easy to understand and implement.
- It solves the subproblems only when it is required.
- It is easy to debug.

Disadvantages

It uses the recursion technique that occupies more memory in the call stack. Sometimes when the recursion is too deep, the stack overflow condition will occur.

It occupies more memory that degrades the overall performance.

Let's understand dynamic programming through an example.

```
1. int fib(int n)
2. {
3.    if(n<0)
4.    error;
5.  if(n==0)
6.  return 0;
7.  if(n==1)
8. return 1;
9. sum = fib(n-1) + fib(n-2);
10.      }
```