Staircase Problem

Problem

A person can climb one, two or three stairs at a time. Find the number of ways in which he can climb n^{th} stairs.

Example

n = 5

Ways:

{1, 1, 1, 1, 1}
{1, 1, 1, 2}
{1, 1, 2, 1}
{1, 2, 1, 1}
{2, 1, 1, 1}
{1, 2, 2}
{2, 1, 2}
{2, 2, 1}
{1, 1, 3}
{1, 3, 1}
{3, 1, 1}
{2, 3}
{3, 2}

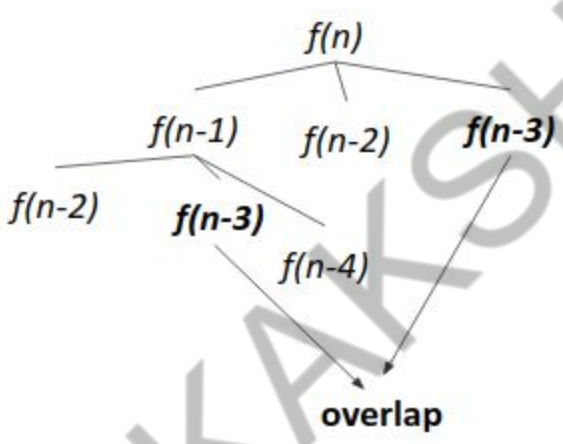
Hence total number of ways = 13.

It can be easily seen that

$$f(n) = f(n-1) + f(n-2) + f(n-3)$$

Since it can be represented as Recurrence Relation, therefore it has an optimal substructure property.

Checking whether it has overlapping subproblem property
Making recursion tree



Since f(n-3) repeats, it follows overlapping subproblem property.

Approach 1 (Memoization)

1. Write the recursive solution.
2. Memoize it.

Approach 2 (Tabulation)

1. Initialize *f(0)=1, f(1)=1, f(2)=2.*
2. Iterate from *i=3 to i=n*, and keep applying *f(i) = f(i-1) + f(i-2) + f(i-3).*
3. Output *f(n).*

Code

```
void solve()
{
    int n;
    cin >> n;

    vi dp(n+1);

    dp[0] = 1;
    dp[1] = 1;
    dp[2] = 2;

    rep(i,3,n+1)
    {
        dp[i] = dp[i-1] + dp[i-2] + dp[i-3];
    }

    cout << dp[n] << endl;
}
```