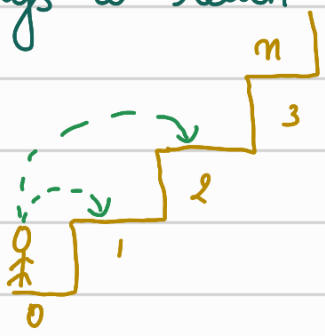
Stair Case problem → We have a staircase with 'n' no. of stairs. We can take either 1 or 2 steps So we have to find ways to reach n^{th} stair.



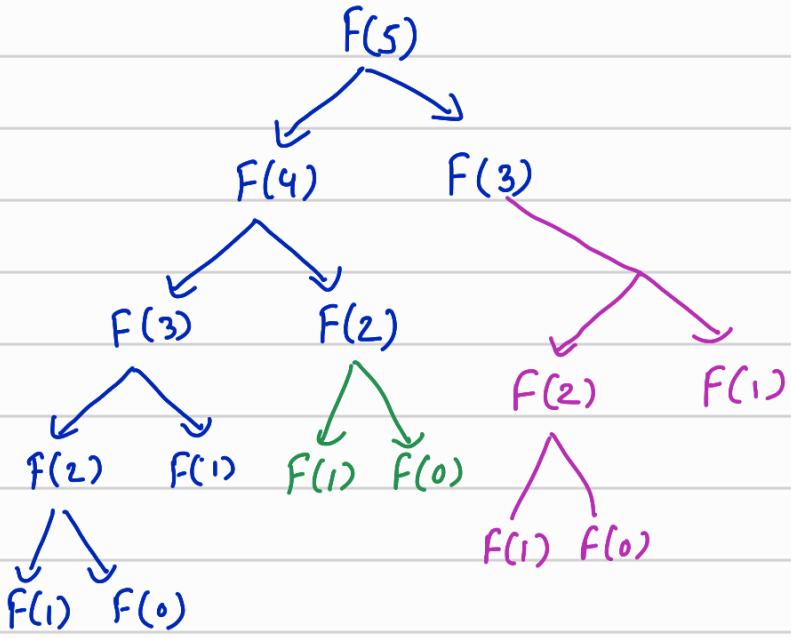
Solution → Now we can see that to reach n^{th} step we can either stand on 'n-1' stair or 'n-2' stair.

So total ways to reach n^{th} step :-

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

```cpp
#include <iostream>
using namespace std;

int climbStairs(int n){
    //base case
    if(n == 0 || n==1)
        return 1;

    int ans = climbStairs(n-1) + climbStairs(n-2);
    return ans;
}

using namespace std;

int main()
{
    int n;
    cout << "Enter the value of n" << endl;
    cin >> n;

    int ans = climbStairs(n);
    cout << "Answer is : " << ans << endl;

    return 0;
}
```

Recursion tree



Problem-2 → Print array elements using Recursion :-

```cpp

#include <iostream>
using namespace std;

void printArr(int arr[], int n ,int i){
    //base case
    if( i >= n){
        return;
    }

    cout << arr[i] << " ";
    printArr(arr,n, i+1);
}

int main()
{
    int arr[5] = {10,20,30,40,50};

    int n = 5;
    int i = 0;
    printArr(arr,n,i);

    return 0;
}

```

Problem-3 ⇒ Finding Max in an Array

```cpp
#include <iostream>
#include<limits.h>
using namespace std;

void findMax(int arr[], int n ,int i, int& maxi){
    if( i >= n){
        return;
    }

    if(arr[i] > maxi){
        maxi = arr[i];
    }

    findMax(arr,n,i+1,maxi);
}

int main()
{
    int arr[] = {10,30,21,44,32,17,19,66};
    int n =8;

    int maxi = INT_MIN;

    int i = 0;
    findMax(arr, n , i , maxi);

    cout << "Maximum number is : " << maxi << endl;

    return 0;
}
```

Problem-4 ⇒ Check 'char' in string

```cpp
#include <iostream>
#include<limits.h>
using namespace std;

int checkKey(string& str, int i, int& n, char& key){
    if(i >= n){
        return -1;
    }

    if(str[i] == key){

        //If you want to print the location of all the letter dound
        //cout << "Found at : " << i << endl;
        return i;
    }

    int ans = checkKey(str, i+1, n , key);
    return ans;
}

int main()
{
    string str = "lovebabbar";
    int n = str.length();

    char key = 'r';

    int i = 0;
    int ans = checkKey(str, i , n , key);

    cout << "Answer is : " << ans << endl;

    return 0;
}
```

Problem -5 ⇒ Print all digits of a no.

Ip → 647 , Op → 6 4 7

```cpp
#include <iostream>
#include <vector>
using namespace std;

void recurPermute(int index, vector<int> &nums, vector<vector<int>> &ans){

    if(index == nums.size()){
        ans.push_back(nums);
        return;
    }
    for(int i = index; i < nums.size(); i++){
        swap(nums[index],nums[i]);
        recurPermute(index+1, nums,ans);
        swap(nums[index], nums[i]);
    }
}

vector<vector<int>> permute(vector<int>& nums){
    vector<vector<int>> ans;
    recurPermute(0, nums, ans);
    return ans;
}

int main() {
    // Write C++ code here
    vector<int> nums = {1, 2, 3};
    vector<vector<int>> ans = permute(nums);
    for(int i = 0; i < ans.size(); i++){
         for (int j = 0; j < ans[i].size(); j++)
        {
            cout << ans[i][j];
        }
        cout << "\n";
    }
    return 0;
}
```

Note → If we pass any no. starting with 0 in above function it will not give us the proper result.

Eg → if Ip → 0647 , Op → 423

Reason → When an integer literal starts with a leading zero, it is interpreted as an octal (base 8) value.

In case of 0647 → $6 8^2 + 4 8^1 + 7 \times 8^0 = 384 + 32 + 7 = 423$