Modulo Arithmetic

Introduction

Modulo (%) operation: This operation returns us the remainder of the division. Example: 17 % 5 = 2. The modulo of a positive number **n** ranges from **0 to n - 1**. In many questions we are required to take **%(10^9 + 7)**. This is because the maximum positive number that we can store in long long is **(2^{64} - 1)**. But the solution might demand to multiply two numbers and their product may be greater than **(2^{64} - 1)**. **For example : 2^{40} * 2^{32} =2^{72}**.

%(10^9 + 7) : This specific number is chosen because:

1. It is a very large number.
2. It is a prime number.

Properties

Modulo arithmetic has the following properties:

1. **(a + b) % c = ((a % c) + (b % c)) % c**
2. **(a * b) % c = ((a % c) * (b % c)) % c**
3. **(a - b) % c = ((a % c) - (b % c)) % c**
4. **(a / b) % c = ((a % c) / (b % c)) % c**, given that a Multiplicative Modulo Inverse should exist.

The major use of modulo arithmetic is to prevent range overflow at every step, that is , in case the operations on any of the operands in the solution cause the operand to go out of range of its data type, then taking modulo at that very step makes sure that our program/code works correctly.

Now, **if (x % m) = -ve** , then we change the formula to **(x%m + m) % m**.

Number of Balanced BTs

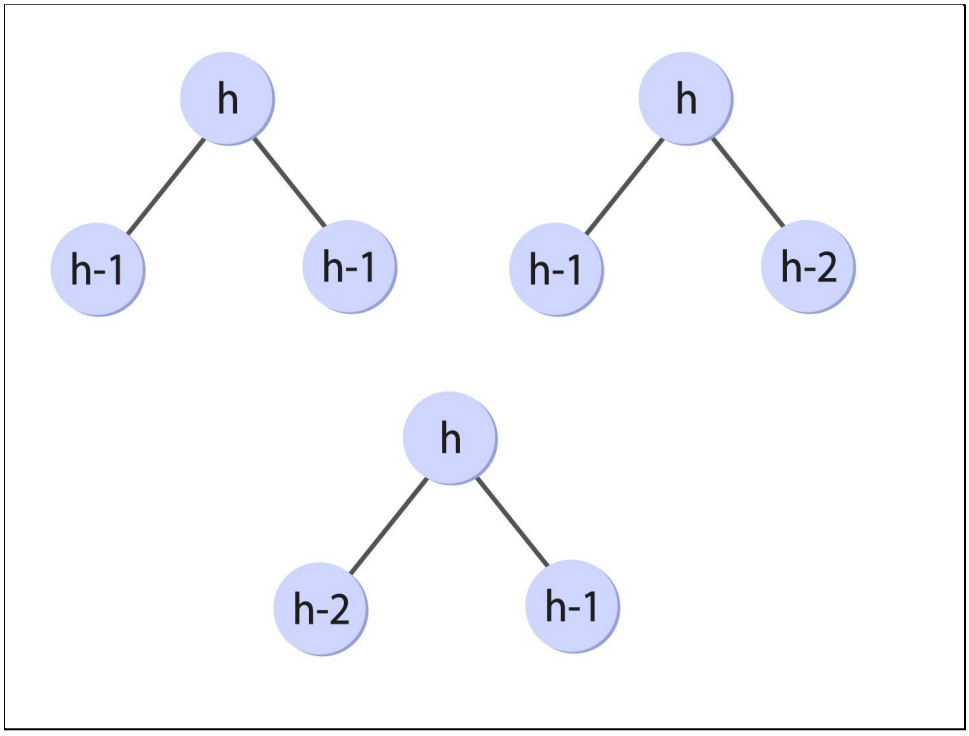
Problem Statement: Given height = **h**. Find the number of balanced binary trees possible.

Explanation:

Binary Tree: A generic tree with at most two child nodes for each parent node is known as a binary tree.
A binary tree is made of nodes that constitute a **left** pointer, a **right** pointer, and a **data** element. The **root** pointer is the topmost node in the tree. The left and right pointers recursively point to smaller subtrees on either side. A null pointer represents a binary tree with no elements, i.e., an empty tree. A formal definition is: a binary tree is either empty (represented by a null pointer), or is made of a single node, where the left and right pointers (recursive definition ahead) each point to a binary tree.

For a **Balanced Binary Tree**, **| lh - rh | <= 1** , that is the difference between left height and right height must be less than or equal to 1 **for every node**.

Possibilities: Suppose there are **x** number of trees for **h - 1** height, **y** for **h -2** number of height. Therefore we have three possibilities:



Total possibility : x*x + x*y + x*y

Code:

```
#include<bits/stdc++.h>

using namespace std;


int balancedBTs(int h){

    if(h==0 || h==1){
        return 1;
    }

    int m = pow(10,9) + 7;
    int x = balancedBTs(h-1);
    int y = balancedBTs(h-2);
```

```
    long res1 = (long)x*x;
    long res2 = (long)x*y*2;

    int ans1 = (int)(res1%m);
    int ans2 = (int)(res2%m);

    int ans = (ans1+ans2)%m;

    return ans;
}

int main(){

    int h=8;
    cout << balancedBTs(h) << endl;
    return 0;
}
```