Count Balanced Binary Trees of Height h

Last Updated : 10 Dec, 2022

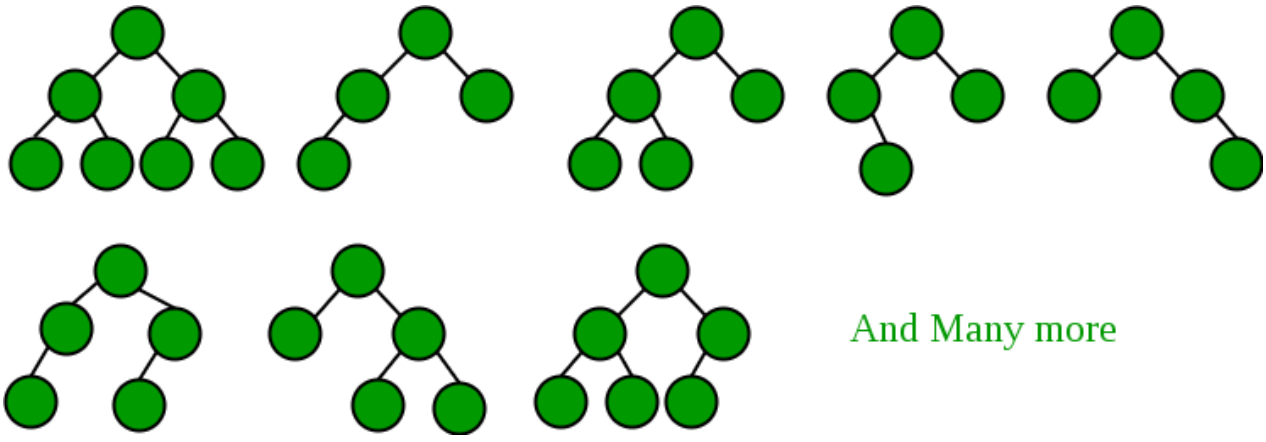
Given a height h, count and return the maximum number of balanced binary trees possible with height h. A balanced binary tree is one in which for every node, the difference between heights of left and right subtree is not more than 1.

Examples :

```
Input : h = 3
Output : 15

Input : h = 4
Output : 315
```

Following are the balanced binary trees of height 3.



Height of tree, h = 1 + max(left height, right height)
Since the difference between the heights of left and right subtree is not more than one, possible heights of left and right part can be one of the following:

3. (h-1), (h-1)

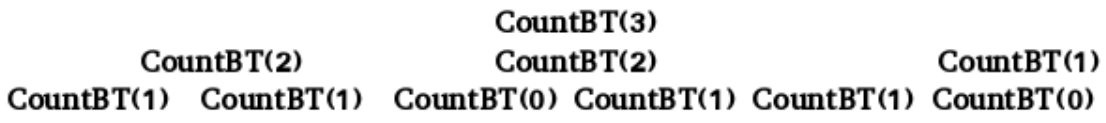
```
count(h) = count(h–1) * count(h–2) +
           count(h–2) * count(h–1) +
           count(h–1) * count(h–1)
         = 2 * count(h–1) * count(h–2) +
           count(h–1) * count(h–1)
        = count(h–1) * (2*count(h – 2) +
                        count(h – 1))
```

Hence we can see that the problem has optimal substructure property.

A **recursive function** to count no of balanced binary trees of height h is:

```
int countBT(int h)
{
    // One tree is possible with height 0 or 1
    if (h == 0 || h == 1)
        return 1;
    return countBT(h–1) * (2 *countBT(h–2) +
                           countBT(h–1));
}
```

The time complexity of this recursive approach will be exponential. The recursion tree for the problem with h = 3 looks like :



As we can see, sub-problems are solved repeatedly. Therefore we store the results as we compute them.

An efficient dynamic programming approach will be as follows :

Below is the implementation of above approach:

```
// binary trees of height h.
#include <bits/stdc++.h>
#define mod 1000000007
using namespace std;

long long int countBT(int h) {

    long long int dp[h + 1];
    //base cases
    dp[0] = dp[1] = 1;
    for(int i = 2; i <= h; i++) {
        dp[i] = (dp[i - 1] * ((2 * dp [i - 2])%mod + dp[i - 1])%mod) % mod;
    }
    return dp[h];
}


// Driver program
int main()
{
    int h = 3;
    cout << "No. of balanced binary trees"
            " of height h is: "
         << countBT(h) << endl;
}
```

Java

```
// Java program to count number of balanced
// binary trees of height h.
import java.io.*;
class GFG {

    static final int MOD = 1000000007;

    public static long countBT(int h) {
        long[] dp = new long[h + 1];

        // base cases
        dp[0] = 1;
        dp[1] = 1;

        for(int i = 2; i <= h; ++i)
            dp[i] = (dp[i - 1] * ((2 * dp [i - 2])% MOD + dp[i - 1]) % MOD)
```

```
    // Driver program
    public static void main (String[] args) {
        int h = 3;
        System.out.println("No. of balanced binary trees of height "+h+" is
    }
}
/*
This code is contributed by
Brij Raj Kishore
*/
```

Python3

```
# Python3 program to count number of balanced
# binary trees of height h.

def countBT(h) :
    MOD = 1000000007
    #initialize list
    dp = [0 for i in range(h + 1)]

    #base cases
    dp[0] = 1
    dp[1] = 1

    for i in range(2, h + 1) :
        dp[i] = (dp[i - 1] * ((2 * dp [i - 2])%MOD + dp[i - 1])%MOD) % MO

    return dp[h]

#Driver program
h = 3
print("No. of balanced binary trees of height "+str(h)+" is: "+str(countBT(

# This code is contributed by
# Brij Raj Kishore
```

C#

```
// C# program to count number of balanced
// binary trees of height h.
```

```
    public static long countBT(int h) {
        long[] dp = new long[h + 1];

        // base cases
        dp[0] = 1;
        dp[1] = 1;

        for(int i = 2; i <= h; ++i)
            dp[i] = (dp[i - 1] * ((2 * dp [i - 2])% MOD + dp[i - 1]) % MOD)

            return dp[h];
    }

    // Driver program
    static void Main () {
        int h = 3;
        Console.WriteLine("No. of balanced binary trees of height "+h+" is:
    }
    // This code is contributed by Ryuga
}
```

PHP

```
<?php
// PHP program to count
// number of balanced

$mod =1000000007;

function countBT($h)
{
    global $mod;

    // base cases
    $dp[0] = $dp[1] = 1;
    for($i = 2; $i <= $h; $i++)
    {
        $dp[$i] = ($dp[$i - 1] *
                  ((2 * $dp [$i - 2]) %
                  $mod + $dp[$i - 1]) %
                         $mod) % $mod;
    }
    return $dp[$h];
```

```
$h = 3;
echo "No. of balanced binary trees",
                " of height h is: ",
                    countBT($h) ,"\n";


// This code is contributed by aj_36
?>
```

Javascript

```
// Javascript program to count number of balanced binary trees of heigh

let MOD = 1000000007;

function countBT(h) {
    let dp = new Array(h + 1);
    dp.fill(0);

    // base cases
    dp[0] = 1;
    dp[1] = 1;

    for(let i = 2; i <= h; ++i)
        dp[i] = (dp[i - 1] * ((2 * dp [i - 2])% MOD + dp[i - 1]) % MOD)

        return dp[h];
}

let h = 3;
  document.write("No. of balanced binary trees of height h is: "+countB
```

Output

```
No. of balanced binary trees of height h is: 15
```

Time Complexity: O(n)
Auxiliary Space: O(n), since n extra space has been taken.

Memory efficient Dynamic Programming approach :

can replace dp[i],dp[i-1] and dp[i-2] with dp2, dp1 and dp0 respectively. (contributed by **Kadapalla Nithin Kumar**)

Implementation:

C++

```
// C++ program to count number of balanced
// binary trees of height h.
#include <bits/stdc++.h>
using namespace std;

long long int countBT(int h) {
      if(h<2){
      return 1;
    }
      const int BIG_PRIME = 1000000007;
    long long int dp0 = 1, dp1 = 1,dp2;

    for(int i = 2; i <= h; i++) {

          dp2 = (dp1 * ((2 * dp0)%BIG_PRIME + dp1)%BIG_PRIME) % BIG_PRIME;

          // update dp1 and dp0
        dp0 = dp1;
          dp1 = dp2;

          // Don't commit following simple mistake
          // dp1 = dp0;
        // dp0 = dp1;
    }
    return dp2;
}


// Driver program
int main()
{
    int h = 3;
    cout << "No. of balanced binary trees"
            " of height h is: "
         << countBT(h) << endl;
}
// This code is contributed by Kadapalla Nithin Kumar
```

```
// Java program to count number of balanced
// binary trees of height h.
import java.io.*;
class GFG {

    static final int BIG_PRIME = 1000000007;
    static long countBT(int h){
        if(h<2){
          return 1;
        }
        long dp0 = 1, dp1 = 1,dp2 = 3;

        for(int i = 2; i <= h; i++) {

              dp2 = (dp1 * ((2 * dp0)%BIG_PRIME + dp1)%BIG_PRIME) % BIG_PRI

              // update dp1 and dp0
            dp0 = dp1;
              dp1 = dp2;

              // Don't commit following simple mistake
              // dp1 = dp0;
            // dp0 = dp1;
        }
        return dp2;
    }
    // Driver program
    public static void main (String[] args) {
        int h = 3;
        System.out.println("No. of balanced binary trees of height "+h+" is
```



```
    }
}
/*
This code is contributed by
Brij Raj Kishore and modified by Kadapalla Nithin Kumar
*/
```

Python3

```
# Python3 program to count number of balanced
# binary trees of height h.

def countBT(h):
```

```
    dp0 = dp1 = 1
    dp2 = 3
    for _ in range(2,h+1):
        dp2 = (dp1*dp1 + 2*dp1*dp0)%BIG_PRIME
        dp0 = dp1
        dp1 = dp2
    return dp2
#Driver program
h = 3
print("No. of balanced binary trees of height "+str(h)+" is: "+str(countBT(

#This code is contributed by Kadapalla Nithin Kumar
```

C#

```
// C# program to count number of balanced
// binary trees of height h.

using System;
class GFG {

    static int BIG_PRIME = 1000000007;

    public static long countBT(int h) {
          // base case
          if(h<2){
          return 1;
        }

          long dp0 = 1;
        long dp1 = 1;
        long dp2 = 3;
        for(int i = 2; i <= h; ++i){
            dp2 = (dp1 * ((2 * dp0)% BIG_PRIME + dp1) % BIG_PRIME) % BIG_PR
              dp0 = dp1;
              dp1 = dp2;
        }
       return dp2;
    }

    // Driver program
    static void Main () {
        int h = 3;
        Console.WriteLine("No. of balanced binary trees of height "+h+" is:
```

PHP

```php
<?php
// PHP program to count
// number of balanced

$BIG_PRIME =1000000007;

function countBT($h)
{
    global $BIG_PRIME;

    // base cases
      if($h < 2){
      return 1;
    }
    $dp0 = $dp1 = 1;
    $dp2 = 3;
    for($i = 2; $i <= $h; $i++)
    {
        $dp2 = ($dp1 *
                  ((2 * $dp0) % $BIG_PRIME
                  + $dp1) %
                           $BIG_PRIME) % $BIG_PRIME;
          $dp0 = $dp1;
          $dp1 = $dp2;
    }
    return $dp2;
}


// Driver Code
$h = 3;
echo "No. of balanced binary trees",
               " of height h is: ",
                   countBT($h) ,"\n";


// This code is contributed by aj_36 and modified by Kadapalla Nithin Kumar
?>
```

Javascript

```
function countBT(h) {
    if(h<2){
        return 1;
    }
    // base cases
    let dp0 = 1;
    let dp1 = 1;
    let dp2 = 3;
    for(let i = 2; i <= h; ++i){
        dp2 = (dp1 * ((2 * dp0)% MOD + dp1) % MOD) % MOD;
        dp0 = dp1;
        dp1 = dp2;
    }
    return dp2;
}

let h = 3;
  document.write("No. of balanced binary trees of height h is: "+countB
// This code is contributed by Kadapalla Nithin Kumar
```

Output

```
No. of balanced binary trees of height h is: 15
```

Time Complexity: O(n)
Auxiliary Space: O(1)

other Geek.

Next Article

Comparison between Height Balanced Tree and Weight Balanced Tree

Similar Reads