600. Non-negative Integers without Consecutive Ones

DescriptionHintsSubmissionsSolutions

- Total Accepted: **1518**
- Total Submissions: **5741**
- Difficulty: **Hard**
- Contributors:sanxi

Given a positive integer n, find the number of **non-negative** integers less than or equal to n, whose binary representations do NOT contain **consecutive ones**.

Example 1:

Input: 5**Output:** 5**Explanation:**

Here are the non-negative integers <= 5 with their corresponding binary representations:

0 : 0

1 : 1

2 : 10

3 : 11

4 : 100

5 : 101

Among them, only integer 3 disobeys the rule (two consecutive ones) and the other 5 satisfy the rule.

Note: 1 <= n <= 109

Subscribe to see which companies asked this question.

Show Tags

Show Similar Problems

Have you met this question in a real interview?

Yes

Python dp solutions

```
class Solution(object):
    def findIntegers(self, num):
        """
        :type num: int
        :rtype: int
        """
        A=bin(num)[2:][::-1]
        # dp[i][0] is the number of integers with (i+1)bits, highest bit is 0 and without consecutive ones
        # dp[i][1] is the number of integers with (i+1)bits, highest bit is 1 and without consecutive ones
        dp=[[1,1] for _ in range(len(A))]
        # res is the number of integers less than A[:i] without consecutive ones.
        res=1 if A[0]=='0' else 2
        for i in range(1, len(A)):
            dp[i][0]=dp[i-1][0]+dp[i-1][1]
            dp[i][1]=dp[i-1][0]
            if A[i-1:i+1]=='01':
                # if A[i-1:i+1]=='01', we can append '1' after integers less than A[:i] without consecutive ones,
                # also any integer with A[i]=='0', without consecutive ones is less than A[:i+1]
                res+=dp[i][0]
            elif A[i-1:i+1]=='11':
                # if A[i-1:i+1]=='11', then any integer with i+1 bits and without consecutive ones is less than A[:i+1]
                res=dp[i][0]+dp[i][1]
            # if A[i]=='0', the number of integers  with i+1 bits, less than A[:i+1]  and without consecutive ones is the same as A[:i]
```

~~~
        return res
          ```
~~~

C++ dp solutions

This problem can be solved using Dynamic Programming. Let dp[i][0] be the number of binary strings of length i which do not contain any two consecutive 1’s and which end in 0. Similarly, let dp[i][1] be the number of such strings which end in 1. We can append either 0 or 1 to a string ending in 0, but we can only append 0 to a string ending in 1. This yields the recurrence relation:

```
#include<iostream>
#include<stdio.h>
#include<vector>
#include<unordered_set>
#include<unordered_map>
#include<limits.h>
#include<set>
#include<algorithm>
#include<sstream>
#include<stdlib.h>
#include<string.h>
using namespace std;
int dp[32][2];
void init()
{
    dp[0][1]=1; dp[0][0]=1;
    for(int i=1;i<=31;i++)
    {
        //00 and 10
        dp[i][0] = dp[i-1][0]+dp[i-1][1];
        // no consecutive 1 --> 01
        // 11 is unpermissive
        dp[i][1] = dp[i-1][0];
    }
}

int dfs(int num,int len)
```

```
{
    if(len<=0) return 1;
    int val = 1<<(len-1);
    if(num>=val)
    {
        return dp[len-1][0] + dfs(num-val,len-2);
    }else
    {
        return dfs(num,len-1);
    }
}

int findIntegers(int num) {
    init();
    int len = 0;
    int n = num;

    while (n)
    {
        len++;
        n >>= 1;
    }

    return dfs(num, len);
}

int main(int argc,char *argv[])
{
    int ans = findIntegers(5);
    cout << ans << endl;
    return 0;
}
```