**★★★☆☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：10328: Coin Toss**

**解題者：張鈞富**

**解題日期：2019年4月11日**

**題意：**

輸入兩數n, k(1 ≤ k ≤ n ≤ 100)，計算在連續投擲n次硬幣之所有情況中，有至少連續k次正面的情況的次數，並將其輸出

**題意範例：**

4 1 → 15

4 2 → 8//全部的組合-dp(4,1) = 16-8 = 8

4 3 → 3

4 4 → 1

6 2 → 43

**解法：**

計算所有情況減去最多出現連續m次正面之情形

 dp(n,m)=dp(n-1,m)\*2-dp(n-m-2,m)

dp(n,m)=丟n次硬幣其中最多連續出現m次正面

dp(n-1,m)\*2=丟n-1次硬幣時就已經達成條件\*2

 (第n次分別為正反面的情形)

 (投擲n次硬幣的組合等於投擲n-1次硬幣的組合\*2)

dp(n-m-2,m)=前n-1次中的最後m次都是正面且第n次也是正面

 (不合法的狀況，連續出現m+1次正面)

 **(dp(n-m-2),m)(反面)(m個正面) (正面)**

(連續擲硬幣n次，假設n-1次中最後連續出現m次正面，為符合規定所以連續m個正面的前一次為反面，假設投擲硬幣的總次數為n次，則剩下的情況為dp(n-m-2,m)，以上為不合規定的情況，所以要扣除。)



以4 2為例



4 2代表丟擲4次硬幣最少連續出現2次正面之情形，等於所有情況減掉丟擲4次硬幣最多連續出現1次正面之情形(dp(4,1))。而dp(4,1)=dp(3,1)\*2-dp(1,1)=5\*2-2=8(打勾處)，所以4 2會輸出8 (16-8)。

**解法範例:**

**** 輸入為4 1，因此要求dp(4,0)，而dp(4,0)=dp(3,0)\*2-dp(1,0)=1\*2-1=1(打勾處)，所以4 1會輸出15 (16-1)。

 當輸入為4 3，要求出dp(4,2)，而dp(4,2)=dp(3,2)\*2-dp(1,2)=7\*2-1=13(打勾處)，所以4 3會輸出3 (16-13)。

**討論：**

為什麼要逆向思考?

 dp的概念是把大問題切割成小問題，再由小問題推出大問題的解。

 對於此題，若是逆向思考的話，只要考慮最後幾次是否符合答案要求；而以正向思考的話，小問題會有太多形式，還需多考慮到先前符合的情況在後來是否也符合。

 簡單來說，先考慮所有情形再減去不符合的情形會比每次都要重新考慮有沒有後來才符合的情況要加入來的簡單。

 另外，此題的dp出來的資料會超過long long，所以在dp的同時要用大數處理

**程式：**

// Problem: 10328 - Coin Toss

// Author: 張鈞富

// Date: 2019/04/17

#include<iostream>

#include<cmath>

using namespace std;

#define N 101

#define M 101

#define BIG\_NUMBER 33

long long dp\_result[N\*M][BIG\_NUMBER] = { 0 };

void dp() {

 int i, j, k;//投擲i次硬幣的所有組合中，最多出現連續j次正面的次數

 for (i = 0; i<N; ++i)

 for (j = 0; j < M; ++j) {

 if (i <= j) {//i<=j，次數為投擲i次硬幣的所有組合

 if (i < 63) {

 long long temp = pow(2, i);

 for (k = 0; temp != 0 && k < BIG\_NUMBER; ++k) {//將數值轉為大數結構存入

 dp\_result[i\*N + j][k] = temp % 10;

 temp /= 10;

 }

 }

 else {//數值大於long long，分兩次存取資料

 long long temp1 = pow(2, 62);

 long long temp2 = pow(2, i - 62);

 for (k = 0; temp1 != 0 && k < BIG\_NUMBER; ++k) {//先將2的63次方轉為大數結構存入

 dp\_result[i\*N + j][k] = temp1 % 10;

 temp1 /= 10;

 }

 for (k = 0; k < BIG\_NUMBER; ++k) //乘上剩下的數

 dp\_result[i\*N + j][k] \*= temp2;

 for (k = 0; k < BIG\_NUMBER - 1; ++k)//進位

 if (dp\_result[i\*N + j][k] >= 10) {

 dp\_result[i\*N + j][k + 1] += dp\_result[i\*N + j][k] / 10;

 dp\_result[i\*N + j][k] = dp\_result[i\*N + j][k] % 10;

 }

 }

 }

 else if (i - j - 2 >= 0) {//dp(i,j)=dp(i-1,j)\*2-dp(i-j-2,j)

 for (k = 0; k < BIG\_NUMBER; ++k)

 dp\_result[i\*N + j][k] = dp\_result[(i - 1)\*N + j][k] \* 2 - dp\_result[(i - j - 2)\*N + j][k];

 for (k = 0; k < BIG\_NUMBER - 1; ++k)//進位

 if (dp\_result[i\*N + j][k] >= 10) {

 dp\_result[i\*N + j][k + 1] += dp\_result[i\*N + j][k] / 10;

 dp\_result[i\*N + j][k] = dp\_result[i\*N + j][k] % 10;

 }

 else if (dp\_result[i\*N + j][k] < 0) {

 dp\_result[i\*N + j][k + 1] += dp\_result[i\*N + j][k] / 10;

 if (dp\_result[i\*N + j][k] % 10 != 0)

 --dp\_result[i\*N + j][k + 1];

 dp\_result[i\*N + j][k] = dp\_result[i\*N + j][k] + 10;

 }

 }

 else if (i - j - 2 < 0) {//dp(i,j)=dp(i-1,j)\*2-1

 for (k = 0; k < BIG\_NUMBER; ++k)

 dp\_result[i\*N + j][k] = dp\_result[(i - 1)\*N + j][k] \* 2;

 dp\_result[i\*N + j][0] -= 1;

 for (k = 0; k < BIG\_NUMBER - 1; ++k)//進位

 if (dp\_result[i\*N + j][k] >= 10) {

 dp\_result[i\*N + j][k + 1] += dp\_result[i\*N + j][k] / 10;

 dp\_result[i\*N + j][k] = dp\_result[i\*N + j][k] % 10;

 }

 else if (dp\_result[i\*N + j][k] < 0) {

 dp\_result[i\*N + j][k + 1] += dp\_result[i\*N + j][k] / 10;

 if (dp\_result[i\*N + j][k] % 10 != 0)

 --dp\_result[i\*N + j][k + 1];

 dp\_result[i\*N + j][k] = dp\_result[i\*N + j][k] + 10;

 }

 }

 }

}

int main(void) {

 int n, k, i, j, answer[BIG\_NUMBER];

 dp();

 while (cin >> n >> k) {

 //算出投擲n次硬幣總共會有幾種組合(2的n次方)

 for (i = 0; i < BIG\_NUMBER; ++i)

 answer[i] = 0;

 answer[0] = 1;

 for (i = 0; i < n; ++i) {

 for (j = 0; j < BIG\_NUMBER; ++j)

 answer[j] \*= 2;

 for (j = 0; j < BIG\_NUMBER - 1; ++j)//進位

 if (answer[j] >= 10) {

 answer[j + 1] += answer[j] / 10;

 answer[j] = answer[j] % 10;

 }

 }

 //全部的組合減掉dp()

 for (i = 0; i < BIG\_NUMBER; ++i)

 answer[i] = answer[i] - dp\_result[n\*N + k - 1][i];

 for (j = 0; j < BIG\_NUMBER - 1; ++j)//檢查每個位數有每有大於0

 if (answer[j] < 0) {

 answer[j] += 10;

 --answer[j + 1];

 }

 for (i = BIG\_NUMBER - 1; answer[i] == 0 && i >= 0; --i);//輸出答案

 for (; i >= 0; --i)

 cout << answer[i];

 cout << endl;

}

 return 0;

}