You are climbing a stair case. It takes *n* steps to reach to the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

**Example 1:**

**Input:** 2

**Output:** 2

**Explanation:** There are two ways to climb to the top.

1. 1 step + 1 step

2. 2 steps

爬塔,每次跨一个台阶，那么爬到第n格，一共有几个台阶

首先我们要意识到这个是recursion递推公式，又有重复的地方，那么毫无疑问，动态规划

class Solution {

public int climbStairs(int n) {

if (n == 1) {

return 1;

}

int dynamicArray[] = new int[n + 1];

dynamicArray[1] = 1;

dynamicArray[2] = 2;

for (int i = 3; i <= n; i++) {

dynamicArray[i] = dynamicArray[i - 1] + dynamicArray[i - 2];

}

return dynamicArray[n];

}

}

dynamicArray[i] = dynamicArray[i - 1] + dynamicArray[i - 2];

这句话解释：

想要到达第十阶，一共有两种方法，一个是从第八阶跨两格，一个是从第九阶跨一格（第八阶跨1格再跨一格是陷阱，相当于第九阶跨一格）

然后把每次的结果记载dynamicArray里

方法2，fibonacci法

class Solution {

public int climbStairs(int n) {

if (n == 1) {

return 1;

}

int first = 1;

int second = 2;

for (int i = 3; i <= n; i++) {

int third = first + second;

first = second;

second =third;

}

return second;

}

}

当我们总结出公示的时候，我们发现他就是经典fibonacci

那么完全可以用first,second,third解决，

n==1 return 1

first=1

second=2 // 跨两格加上一格一格两种方案

for (int i=3;i<n;i++){

int third=first+second

first=second //我们已经可以舍去first了

second=third //我们可以舍去second了

}