假设你正在爬楼梯。需要 *n* 阶你才能到达楼顶。

每次你可以爬 1 或 2 个台阶。你有多少种不同的方法可以爬到楼顶呢?

注意：给定 *n* 是一个正整数。

示例 1：

```
输入： 2
输出： 2
解释： 有两种方法可以爬到楼顶。
1. 1 阶 + 1 阶
2. 2 阶
```

示例 2：

```
输入： 3
输出： 3
解释： 有三种方法可以爬到楼顶。
1. 1 阶 + 1 阶 + 1 阶
2. 1 阶 + 2 阶
3. 2 阶 + 1 阶
```

理解题目：

输入是：台阶数：n （相当于和）

输出是：得到n的方法：m

最笨的方法：

n = 1 --> m =1;

n = 2 --> m =2;

n = 3 --> m =3; (n=2 + 1节台阶 = 2 +1) ；

n = 4 --> m= 5; （n=3 + 1节台阶 或者是 n=2 + 2节台阶 = 3 +2)

n = 5 --> m=8; (n =4 + 1节台阶;或者 n =3 + 2节台阶 ; = 5 + 3)

所以相当于斐波那契数列数列

f(n) = f(n-1) + f(n -2)

f(1) = 1;

f(2) = 2;

方法一：递推 时间复杂度（O（n）？）

```c
int climbStairs(int n){
    if (n < 0)
        return 0;
    if (n <= 2){
        return n;
    }
    int n1 = 1,n2 =2, num;
    for(int i = 2; i < n ; i++)
    {
         num = n1 + n2;
         n1 = n2;
         n2 = num;
     }
     return num;
}
```

2 递归

```c
int _climb(int i, int *arr){
    if (arr[i] != 0)
        return arr[i];
    arr[i] = _climb(i-1, arr) + _climb(i-2, arr);
    return arr[i];
}
```

```c
int climbStairs(int n){
    int num ;
    if (n <0)
        return 0;
     if (n <=2)
        return n;
     int *arr = (int*)calloc(n+1, sizeof(int));
     arr[1] = 1;
     arr[2] = 2;
     num = _climb(n, arr);
     free(arr);
     return num;
}
```

python的解法：

1 递归：

```python
def climbStairs(self, n: int) -> int:
        if n == 1: return 1
        if n == 2: return 2
        return self.climbStairs(n-1) + self.climbStairs(n-2)
```

但是这样很慢，很容易超时，因此可以加一个缓存装饰器

```python
import functools
@functools.lru_cache(100)  # 缓存装饰器
def climbStairs(self, n: int) -> int:
        if n == 1: return 1
        if n == 2: return 2
        return self.climbStairs(n-1) + self.climbStairs(n-2)
```

直接DP，新建一个字典或者数组来存储以前的变量，空间复杂度O(n)

```python
def climbStairs(self, n: int) -> int:
        cs = {}
        cs[1] = 1
        cs[2] = 2
        for i in range(3, n+1):
            cs[i] = cs[i-1] + cs[i-2]
        return cs[n]
```

还是DP，只不过是只存储前两个元素，减少了空间，空间复杂度O(1)，后一个数 = 前两个数之和

```python
def climbStairs(self, n: int) -> int:
        if n == 1 or n == 2:
            return n
        a , b , tmp = 1,2,0
        for i in range(3, n+1):
            tmp = a + b
            a = b
            b = tmp
        return tmp
```

直接斐波那契数列的计算公式喽

log10（F(n)）=-0.5*log10（5.0）+（（double）n）*log（s）/log（10.0）+log10（1-（（1-√5）/（1+√5））^n)

s = (1+sqrt(5.0))/2.0;

```python
class Solution:
    def climbStairs(self, n: int) -> int:
```

```
        import math
        sqrt5=5**0.5
        fibin=math.pow((1+sqrt5)/2,n+1)-math.pow((1-sqrt5)/2,n+1)
        return int(fibin/sqrt5)
```

面向测试用例编程

```
class Solution:
    def climbStairs(self, n: int) -> int:
        a = [1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,28657,463
        return a[n-1]

```