20180322_大富翁游戏掷骰子

大富翁游戏，玩家根据骰子的点数决定走的步数，即骰子点数为1时可以走一步，点数为2时可以走两步，点数为n时可以走n步。求玩家走到第n步（n<=骰子最大点数且是方法的唯一入参）时，总共有多少种投骰子的方法。

输入描述:

```
输入包括一个整数n,(1 ≤ n ≤ 6)
```

输出描述:

```
输出一个整数,表示投骰子的方法
```

输入例子1:

```
6
```

输出例子1:

```
32
```

我开始只想到了：是一种遍历问题，将情景简化为。假设要达到5，那么问题变成，5，4，3，2，1分别到5有几种方式。加起来的和就是答案。

但我这种思路还没有那么清晰：于是看了一下大神的思路：将答案看做是f(n),它由f(n-1),f(n-2),f(n-3)....f(1)分别到f(n)的总和组成。那么f(n-1)到f(n)有一种情况（例如：5到6，只有+1一种情况），f(n-2)到f(n)同样是一种（例如：4到6，需要+2，这里为什么不说4+1+1再到6，不也可以吗？不是的，这是两步了，完成第一步后，4变成5，重复了5到6，因此，只有一种情况就是4+2），以此类推，直到f(1),并且可以知道f(1)和f(2)是可以直接得到的。所以f（n）=f（n-1）* 1+f（n-2）* 1+.....+f（1）*1——》(解释下这里的 *1是什么意思，因为f(n-1)到f(n) 有一种情况，所以这里就是f(n-1)所有情况乘以f(n-1)到f(n)的情况就是所有倒数第二项是n-1的情况的集合了)。于是这里就自然而然想到了递归的情况：代码如下：

一种菲波那切数列。

```
int getNum(int num){
    int count = 0;
    if( num == 1 ){
        count = 1;
    }else if( num == 2 )
        count = 2;
    else{
        for( int i = 1;i < num ; i++ ) {
            count += getNum(num - i);
        }
        /// 这里的加1，是说如果自己到自己，例如，num=4，一次就投出个4，那么就是从0到4，这个情况要加上
        count+=1;
    }
    return count;
```

```
}
#include<iostream>
int main(){
    int num;
    std::cin >> num;
    std::cout << getNum(num);
}
```

java 版本。

```
import java.util.Scanner;
public class Test{
  public static void main(String[] args){
     Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        System.out.println(getNum(n));
  }
   public int getNum(int num){
    int count=0;
    if (num==1){
        count=1;
    }
    else if (num==2){
        count=2;
    }
    else{
        for(int i=1;i<num;i++){
            count+=getNum(num-i);
        }
        //这里的加1，是说如果自己到自己，例如，num=4，一次就投出个4，那么就是从0到4，这个情况要加上
        count+=1;
    }
    return count;
}
}
```