1.一个台阶总共n级，如果一次可以跳1级，也可以跳2级。求总共有多少总跳法，并分析算法的时间复杂度。

这个题目就是Fibonacci序列

```
public class Jump {

    public int JumpFloor(int target) {
        if(target == 0) {
            return 0;
        }

        if(target == 1)
            return 1;

        if(target == 2)
            return 2;

        return JumpFloor(target - 1) + JumpFloor(target - 2);
    }

}
```

2.一个台阶总共有n级，如果一次可以跳1级，也可以跳2级，。。。。，也可以跳n级，此时该青蛙跳上一个n级的台阶总共有多少种跳法。

分析：用Fib(n)表示青蛙跳上n阶台阶的跳法数，青蛙一次性跳上n阶台阶的跳法数1(n阶跳)，设定Fib(0) = 1；

当n = 1 时， 只有一种跳法，即1阶跳：Fib(1) = 1;

当n = 2 时， 有两种跳的方式，一阶跳和二阶跳：Fib(2) = Fib(1) + Fib(0) = 2;

当n = 3 时，有三种跳的方式，第一次跳出一阶后，后面还有Fib(3-1)中跳法； 第一次跳出二阶后，后面还有Fib(3-2)中跳法；第一次跳出三阶后，后面还有Fib(3-3)中跳法

Fib(3) = Fib(2) + Fib(1)+Fib(0)=4;

当n = n 时，共有n种跳的方式，第一次跳出一阶后，后面还有Fib(n-1)中跳法； 第一次跳出二阶后，后面还有Fib(n-2)中跳法.....第一次跳出n阶后，后面还有 Fib(n-n)中跳法.

Fib(n) = Fib(n-1)+Fib(n-2)+Fib(n-3)+.....+Fib(n-n)=Fib(0)+Fib(1)+Fib(2)+.....+Fib(n-1)

又因为Fib(n-1)=Fib(0)+Fib(1)+Fib(2)+.....+Fib(n-2)

两式相减得：Fib(n)-Fib(n-1)=Fib(n-1) =====》 Fib(n) = 2*Fib(n-1) n >= 2

递归等式如下：

$$Fib(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ 2*Fib(n-1) & n>2\end{cases}$$

```
public class Jump {

    public int JumpFloor(int target) {

        if(target <= 0)
            return 1;

        if(target == 1)
            return 1;

        return 2 * JumpFloor(target - 1);
    }

    public static void main(String [] args) {
        System.out.println(new Jump().JumpFloor(3));
    }
}
```