1. 题目
2. 题干

写一个函数，输入 n ，求斐波那契（Fibonacci）数列的第 n 项。斐波那契数列的定义如下：

F(0) = 0,   F(1) = 1

F(N) = F(N - 1) + F(N - 2), 其中 N > 1.

斐波那契数列由 0 和 1 开始，之后的斐波那契数就是由之前的两数相加而得出。

答案需要取模 1e9+7（1000000007），如计算初始结果为：1000000008，请返回 1。

1. 示例

**示例 1：**

**输入：**n = 2

**输出：**1

**示例 2：**

**输入：**n = 5

**输出：**5

1. 题解
2. 思路

首先想到的是递归思路，但是由于递归过于臃肿，测试用例有的超时太严重。因此，想到使用一个数组记忆之前的每个数，但是这种方法需要O(n)的空间复杂度。通过题目的提示，可以想到用动态规划来解决，由于状态转移方程是F(n) = F(n-1) + F(n-2)，因此我们只需要动态地维护三个数a, b, sum即可，用这三个数不断迭代。最终获得需要的数。

1. 代码实现

Java：

1. 递归

class Solution {

    public int fib(int n) {

         //递归

         if(n == 0){

             return 0;

         }else if(n == 1){

             return 1;

        }else{

             return fib(n-1) + fib(n-2);

        }

    }

}

1. 记忆数组

class Solution {

    public int fib(int n) {

        //记忆数组

        if(n == 0){

            return 0;

        }else if(n == 1){

            return 1;

        }

        List<Integer> num = new ArrayList<Integer>();

        num.add(0);

        num.add(1);

        for(int i = 2; i <= n ; ++i){

            int newNum = (num.get(i-1) + num.get(i-2)) % 1000000007;

            num.add(newNum);

        }

        //0 1 1 2 3 5

        return num.get(n);

    }

}

1. 动态规划

class Solution {

    public int fib(int n) {

        //动态规划

        int a = 0, b = 1, sum = 0;

        for(int i = 0; i < n ; ++i){

            sum = (a+ b) % 1000000007;

            a = b;

            b = sum;

        }

        return a;

    }

}