

LintCode领扣题解 (/problem) / 斐波纳契数列 · Fibonacci

斐波纳契数列 · Fibonacci

中文

NetEase (/problem/?tags=netease)

非递归 (/problem/?tags=non-recursion)

数学 (/problem/?tags=mathematics)

枚举法 (/problem/?tags=enumeration)

描述

查找斐波纳契数列中第 N 个数。

所谓的斐波纳契数列是指：

- 前2个数是 0 和 1 。
- 第 i 个数是第 $i-1$ 个数和第 $i-2$ 个数的和。

斐波纳契数列的前10个数字是：

0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ...

❗ 在测试数据中第 $*N*$ 个斐波那契数不会超过32位带符号整数的表示范围

样例

样例 1:

输入: 1
输出: 0

样例解释:

返回斐波那契的第一个数字，是0。

样例 2:

输入: 2
输出: 1

样例解释:

返回斐波那契的第二个数字是1。

在线评测地址: <https://www.lintcode.com/problem/fibonacci/> (<https://www.lintcode.com/problem/fibonacci/>)

收起题目描述 ^

语言类型

ALL (14)

python (5)

java (4)

cpp (3)

javascript (2)

上传题解



令狐冲

更新于 7/28/2020, 2:28:55 PM

纯用递归会超时，如果用带有记忆化的递归就可以，使用循环和记忆化递归的时间复杂度一样，都是 $O(n)$ 。

不超出Integer的斐波那契数很少，仅有50个左右。但是使用纯的递归，复杂度会是 $O(2^n)$ 。因此会超时。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    def fibonacci(self, n):
        a = 0
        b = 1
        for i in range(n - 1):
            a, b = b, a + b
        return a
```

👍 获赞 3

💬 1 条评论

你的口袋题库

2000+ 算法真题、国内外名企题库免费开放



九章算法APP



令狐冲

更新于 6/9/2020, 7:03:48 AM

纯用递归会超时, 如果用带有记忆化的递归就可以, 使用循环和记忆化递归的时间复杂度一样, 都是 $O(n)$ 。

不超出Integer的斐波那契数很少, 仅有50个左右。但是使用纯的递归, 复杂度会是 $O(2^n)$ 。因此会超时。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
    /**
     * @param n: an integer
     * @return an integer f(n)
     */
    public int fibonacci(int n) {
        int a = 0;
        int b = 1;
        for (int i = 0; i < n - 1; i++) {
            int c = a + b;
            a = b;
            b = c;
        }
        return a;
    }
}

//递归版本, 会超时
public class Solution {
    /**
     * @param : an integer
     * @return: an ineger f(n)
     */
    public int fibonacci(int n) {
        // write your code here
        if (n == 1) {
            return 0;
        }

        if (n == 2) {
            return 1;
        }

        return fibonacci(n - 1) + fibonacci(n - 2);
    }
};
```

👍 获赞 3

💬 4 条评论

**DDBear**

更新于 12/3/2020, 10:56:01 AM

解题思路

这道题可以说是十分经典的入门题目, 但是却有很多种做法。

刚开始学习编程语言的同学应该会用一个for循环来解决这个问题。

逐渐地学会递归后, 这道题也可以用递归来解, 但是效率却太差了, 所以需要使用记忆化搜索来优化。

最后我们可能会遇到一种求下标很大的fibonacci数并取模数, 例如下面这道题就让我们求得fibonacci数列的第1,000,000,000项的后四位。

<https://www.lintcode.com/problem/fibonacci-ii/description> (<https://www.lintcode.com/problem/fibonacci-ii/description>)

在看完了本篇题解之后, 我相信这道进阶版的题目也不会难住你。

算法一：递推法

这种算法的朴素写法是所有人都可以写出来的, 因此不做赘述。

它的时间复杂度和空间复杂度均为 $O(n)$ 。

下面的三份代码依次为Java、C++、Python。

java c++ python

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    def fibonacci(self, n):
        fib = [0, 0, 1]
        for i in range(3, n + 1, 1):
            fib.append(fib[i - 1] + fib[i - 2])
        return fib[n]
```

但是我们发现, 这道题并不需要存储那么多的fibonacci数, 因为是返回第n项, 并且第n项只和前面的两个数字有关, 所以利用一个长度为2的空间记录前两个数字即可。

此时时间复杂度不变, 但是空间复杂度降为 $O(1)$ 。

这种节省空间的方法其实就是动态规划的滚动数组思想。

下面的三份代码依次为Java、C++、Python。

java c++ python

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    def fibonacci(self, n):
        fib = [0, 1]
        for i in range(2, n + 1, 1):
            fib[i % 2] = fib[0] + fib[1]
        return fib[(n + 1) % 2]
```

算法二：递归法

这道题是我们学递归时一定会学到的例题, 因为 $\text{fibonacci}[i] = \text{fibonacci}[i - 1] + \text{fibonacci}[i - 2]$, 故递归式为: $\text{thisFibonacci} = \text{dfs}(i) + \text{dfs}(i - 1)$ 。

但是这么做的时间复杂度难以接受, 因为有很多被重复计算的数字, 比如我们在求解fib(10)的时候, 会找到fib(9)和fib(8)共两个, 然后下一层会是fib(8)和fib(7), fib(7)和fib(6)共四个。这是一个呈指数增长的曲线, 其底数为2, 是稳定超时的代码。

时间复杂度为 $O(2^n)$, 空间复杂度为 $O(1)$ 。

下面的三份代码依次为Java、C++、Python。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    def dfs(self, n):
        if n <= 2:
            return n - 1
        return self.dfs(n - 1) + self.dfs(n - 2)

    def fibonacci(self, n):
        return self.dfs(n)
```

这时候就要用到一种经常被用来以空间换时间的优化算法——记忆化搜索。

顾名思义, 它将计算出的结果存储下来, 在计算到指定值的时候, 先判断这个值是否已经计算过, 若没有, 才进行计算, 否则读取已经存储下来的值。这样就把一个指数级复杂度变成了线性复杂度, 代价是空间复杂度从常数级上升至线性级。

时间复杂度为 $O(n)$, 空间复杂度为 $O(n)$ 。

下面的三份代码依次为Java、C++、Python。

java c++ python

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    def dfs(self, n, fib):
        if fib[n] != -1:
            return fib[n]
        if n <= 2:
            fib[n] = n - 1
            return fib[n]
        fib[n] = self.dfs(n - 1, fib) + self.dfs(n - 2, fib)
        return fib[n]

    def fibonacci(self, n):
        result = [-1] * (n + 1)
        self.dfs(n, result)
        return result[n]
```

算法三: 矩阵快速幂

先修知识1: 快速幂: <https://www.lintcode.com/problem/fast-power/description> (<https://www.lintcode.com/problem/fast-power/description>)

先修知识2: 矩阵的乘法运算原理。

在先修知识掌握之后, 我们不禁要问:

为什么求一个fibonacci还能和矩阵、求幂扯上关系?

我们首先来看一个例子:

假设我们有一个22的矩阵 $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ 和一个21的矩阵 $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ ，将上面两个矩阵相乘会变成 $\begin{bmatrix} 3 \\ 2 \end{bmatrix}$ 对吧，再用 $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ 和新的矩阵 $\begin{bmatrix} 3 \\ 2 \end{bmatrix}$ 继续相乘又会变成 $\begin{bmatrix} 5 \\ 3 \end{bmatrix}$ ，继续运算就是 $\begin{bmatrix} 8 \\ 5 \end{bmatrix}$ ， $\begin{bmatrix} 13 \\ 8 \end{bmatrix}$神奇的事情出现了，当我们不断地用这个 $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ 乘上初始矩阵，得到的新矩阵的上面一个元素就会变成的fibonacci数列中的一个数字，下面的元素则是上面元素的前一项，而且每多乘一次，这个数字的下标就增加一。

那么这个矩阵是怎么来的呢？

从刚才的推理中我们发现：某个矩阵A乘上 $\begin{bmatrix} \text{fib}(n+1) \\ \text{fib}(n) \end{bmatrix}$ 会变成 $\begin{bmatrix} \text{fib}(n+2) \\ \text{fib}(n+1) \end{bmatrix}$ 。现在设矩阵A为 $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ ，（为什么矩阵A是一个22的矩阵？因为只有22的矩阵乘一个21的矩阵才会得到一个21的矩阵。）那么可以列出下面的等式：

$$a \text{fib}(n+1) + b \text{fib}(n) = \text{fib}(n+2)$$

$$c \text{fib}(n+1) + d \text{fib}(n) = \text{fib}(n+1)$$

很容易地，我们得到：

$$1 \text{fib}(n+1) + 1 \text{fib}(n) = \text{fib}(n+2)$$

$$1 \text{fib}(n+1) + 0 \text{fib}(n) = \text{fib}(n+1)$$

也就是说矩阵A就是 $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ 。

现在我们知道了原矩阵M连续多次乘上某个矩阵A会得到新的矩阵M'，并且M'的第一个元素就是我们想要的值。

根据矩阵的运算法则，中间的若干次相乘可以先乘起来，但是矩阵乘法的复杂度是 $O(n^3)$ ，是不是一次的乘有点慢呢？这时候就是快速幂出场的时候了，我们可以使用快速幂来优化矩阵乘法的速度，这就是矩阵快速幂算法。

值得注意的是，在快速幂中，我们有一步操作是：int result = 1。那么如何使用矩阵来实现这个单位1呢，我们要借助单位矩阵。所谓的单位矩阵是一个从左上角到右下角对角线上都是1，其余位置都是0的边长相等的矩阵（方阵）。比如 $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 。单位矩阵E的特性在于满足矩阵乘法的任意矩阵AE一定等于A，EA一定等于A。

所以本题需要将初始矩阵设置为 $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ，这样我们只需要将中间矩阵 $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ 使用快速幂连乘n-2次，再和 $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ 相乘，矩阵就变成了 $\begin{bmatrix} \text{fib}(n) \\ \text{fib}(n-1) \end{bmatrix}$ 。

矩阵快速幂算法常常被应用在递推式的加速中，可以很轻松的递推至下标相当大的位置，而不用担心超时的问题。

但是要注意以下两点：

第一，矩阵快速幂使用的过程中要注意是否应该取模，因为C++和Java会有数值溢出，如果题目要求递推式取模，那么有很大概率是一道矩阵快速幂题目。

第二，矩阵乘法是没有交换律的（ $AB \neq BA$ ），因此我们一定要注意乘法顺序。

因为矩阵乘法的复杂度是矩阵长度L的三次方，需要乘logn次。所占的空间一般只有矩阵的空间，为L的平方。

因此时间复杂度为 $O(L^3 \log n)$ ，空间复杂度为 $O(L^2)$ 。

下面的三份代码依次为Java、C++、Python。

java

c++

python

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    def quickPow(self, a, n):
        base = a
        resultMatrix = Matrix(2, 2)
        resultMatrix.numbers[0][0] = 1
        resultMatrix.numbers[1][1] = 1
        while n > 0:
            if n % 2 == 1:
                resultMatrix = resultMatrix.multiply(base)
            base = base.multiply(base)
            n //= 2
        return resultMatrix

    def fibonacci(self, n):
        if n == 1:
            return 0

        startMatrix = Matrix(2, 1)
        rollingMatrix = Matrix(2, 2)

        startMatrix.numbers[0][0] = 1
        startMatrix.numbers[1][0] = 0
        rollingMatrix.numbers[0][0] = 1
        rollingMatrix.numbers[0][1] = 1
        rollingMatrix.numbers[1][0] = 1
        rollingMatrix.numbers[1][1] = 0
        rollingMatrix = self.quickPow(rollingMatrix, n - 2)
        startMatrix = rollingMatrix.multiply(startMatrix)
        return startMatrix.numbers[0][0]

class Matrix:
    def __init__(self, row, column):
        self.row = row;
        self.column = column;
        self.numbers = []
        for i in range(0, row, 1):
            self.numbers.append([0] * column)

    def multiply(self, a):
        newMatrix = Matrix(self.row, a.column)
        for i in range(0, newMatrix.row, 1):
            for j in range(0, newMatrix.column, 1):
                for k in range(0, newMatrix.column, 1):
                    newMatrix.numbers[i][j] = newMatrix.numbers[i][j] + self.numbers[i][k] * a.numbers[k][j]
        return newMatrix
```

👍 获赞 2

💬 1 条评论



令狐冲

更新于 6/9/2020, 7:03:58 AM

纯用递归会超时, 如果用带有记忆化的递归就可以, 使用循环和记忆化递归的时间复杂度一样, 都是 $O(n)$ 。

不超出Integer的斐波那契数很少, 仅有50个左右。但是使用纯的递归, 复杂度会是 $O(2^n)$ 。因此会超时。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution{
public:
    int fibonacci(int n) {
        int a, b;
        a = 0;
        b = 1;
        for(int i = 1; i < n; i++) {
            int c = a + b;
            a = b;
            b = c;
        }
        return a;
    }
};
```

👍 获赞 1 💬 添加评论



华助教

更新于 6/9/2020, 7:04:27 AM

纯用递归会超时, 如果用带有记忆化的递归就可以, 使用循环和记忆化递归的时间复杂度一样, 都是 $O(n)$ 。

不超出Integer的斐波那契数很少, 仅有50个左右。但是使用纯的递归, 复杂度会是 $O(2^n)$ 。因此会超时。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
const fibonacci = function (n) {
    var i, a, b, c;
    a = 0;
    b = 1;
    for (i = 0; i < n - 1; i++) {
        c = a + b;
        a = b;
        b = c;
    }
    return a;
};
```

👍 获赞 0 💬 1 条评论



Senko Hanabi

更新于 9/24/2020, 1:53:32 PM

DP, 利用滚动数组优化空间。感觉比用两个变量的方法更好理解。


```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param n: an integer
    @return: an integer f(n)
    """
    def fibonacci(self, n):
        fib = [0, 1, 1]

        for i in range(3, n):
            fib[i % 3] = fib[(i - 1) % 3] + fib[(i - 2) % 3]

        return fib[(n - 1) % 3]
```

 获赞 2 添加评论**Tin**

更新于 6/9/2020, 7:03:54 AM

拓扑排序用于 Fibonacci

你见过吗? Fibonacci 说起来有N种解法, 我知道的有: 1. 递归定义, 自顶向下, 复杂度 $O(N!)$, 只是教学用的反例 2. 开数组, 从下到上, 初学者能独立发明 3. 三个变量, 交替遍历, 编程小技巧 4. 递归加DP, 为了DP教学用的, 属于硬做DP 5. 代数方法, $O(\log N)$, 最优解, 非算法类, 知道就好 6. 矩阵相乘法, 也是为了在教学中介绍矩阵相乘优化而硬用 7. 这个拓扑排序法, 虽是硬用, 倒是一个练手拓扑排序的好例子

欢迎添加其它的可硬塞在这里的算法

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:

    def fibonacci(self, n):
        """ topological sorting
        """
        if n is 1: return 0
        if n is 2: return 1

        fibs = {i : 0 for i in range(1, n+1)}
        indegrees = {i : 2 for i in range(1, n+1)}

        fibs[1], fibs[2] = 0, 1
        indegrees[1], indegrees[2] = 0, 1

        queue = collections.deque([1])

        while queue:
            k = queue.popleft()
            if k + 1 > n:
                continue
            fibs[k+1] += fibs[k]
            indegrees[k+1] -= 1
            if indegrees[k+1] == 0:
                queue.append(k+1)
            if k + 2 > n:
                continue
            fibs[k+2] += fibs[k]
            indegrees[k+2] -= 1

        return fibs[n]
```

👍 获赞 1

💬 添加评论



九章令狐冲

更新于 10/28/2020, 11:37:12 PM

java

python

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    def fibonacci(self, n):
        if n <= 2:
            return n - 1


        dp = [0] * 3

        # initialize
        dp[1 % 3] = 0
        dp[2 % 3] = 1

        for i in range(3, n + 1):
            dp[i % 3] = dp[(i - 1) % 3] + dp[(i - 2) % 3]

        return dp[n % 3]
```

👍 获赞 0 💬 添加评论



九章用户VW2945

更新于 6/9/2020, 7:04:26 AM

此程序使用了递归的算法，它的时间复杂度比较低（自我感觉），特别是在计算 n 值比较大的时候。此递归解法是通过寻找 fibonacci 数列中的更进一步的规律。例如：F8 = F4/F4 + F5/F5 F7 = F5/F4 + F4/F3。



invitation/sh:



```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
public:
    /**
     * @param n: an integer
     * @return: an ineger f(n)
     */
    int fibonacci(int n) {
        // write your code here
        if(n == 1){
            return 0;
        }
        if(n == 2 || n == 3){
            return 1;
        }
        int ave = n / 2;
        if(n % 2 == 0){
            return fibonacci(ave)*fibonacci(ave)
                + fibonacci(ave + 1)*fibonacci(ave + 1);
        }
        return fibonacci(ave)*fibonacci(ave + 1)
            + fibonacci(ave + 2)*fibonacci(ave + 1);
    }
};
```

👍 获赞 0

💬 添加评论

**九章用户SRECUN**

更新于 6/9/2020, 7:04:11 AM

利用es6 array destruction解构赋值简化代码。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
const fibonacci = function (n) {
    let [a, b] = [0, 1];
    for (let i = 1; i < n; i++) {
        [a, b] = [b, a + b];
    }
    return a;
}
```

👍 获赞 0

💬 添加评论



九章用户GH2SB3

更新于 6/9/2020, 7:04:10 AM

有点慢, 尾递归

每次递归栈都能直接返回你要的结果

时间复杂度O(1)

空间O(n)

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    public int fibonacci(int n) {
        return fib(n, 0, 1);
    }

    private int fib(int n, int a, int b){
        if(n == 1) return a;
        if(n == 2) return b;
        return fib(n-1, b, a+b);
    }
}
```

👍 获赞 0

💬 添加评论

进阶课程

视频+互动

直播+互动

直播+互动

互动课

九章算法班 2021 版

8周时间精通 57 个核心高频考点, 9 招击破 FLAG、BATJ 算法面试。22...

系统架构设计 System Design 2021 版

成为百万架构师必上。30 课时带你快速掌握 18 大系统架构设计知识点与面...

九章算法面试高频题冲刺班

每期更新 15% 题目, 考前押题, 一举拿下 FLAG & BATJ Offer

面向对象设计 OOD

应届生及亚马逊面试必考, IT 求职必备基础

