矩阵乘法在斐波那契数列计算中的应用

周卫星，陈思，张帆

（中国移动（深圳）有限公司，广东 深圳 邮编518048）

［摘　要］：该文介绍了斐波那契数列的一些算法思路，对递归算法、自底向上、比内公式等算法的时间复杂度进行了分析，给出了利用矩阵乘法升维计算降低时间复杂度的方法，对比测试了各算法实现在不同计算量下的执行时间，针对数据溢出，改进为BigInteger的数据类型,给出大数计算下的执行时间对比。

［关键词］：斐波那契；递归；比内公式；矩阵

Matrix Multiplication Apply To Calculation Of Fibonacci Sequence

WEIXING ZHOU ,SI CHEN, FAN ZHANG

(CHINA MOBILE (SHENZHEN) LIMITED, Shenzhen, 518048, GuangDong）

**【Abstract】** This paper introduces some algorithms of calculation Fibonacci sequence, and analyzes the time complexity of recursive, bottom-up and Binet formula algorithms, and gives a method to reduce the time complexity by using two dimension matrix multiplication. Compared those algorithms’ execution time of different amount of calculation. Modified the Long data type to BigInteger data type to resolve the data overflow problem, and compared different algorithms’ execution time for large number calculation.

**【Keywords】** Fibonacci; recursive; Binet formula; matrix

1．引言

斐波那契数列（Fibonacci sequence），又称黄金分割数列、因数学家列昂纳多·斐波那契（Leonardoda Fibonacci）以兔子繁殖为例子而引入，故又称为“兔子数列”，指的是这样一个数列：1、1、2、3、5、8、13、21、34、……这个数列从第3项开始，每一项都等于前两项之和。在数学上，斐波纳契数列以如下被以递归的方法定义：

2．直接递归实现

使用公式f(n)=f(n-1)+f(n-2)，依次递归计算，递归结束条件是f(1)=1，f(2)=1。

核心代码如下：

public static Long recursiveCompute(int n) {

if (n <= 2) return 1L;

else return recursiveCompute(n - 1) + recursiveCompute(n - 2);}

这种方式可以以非常少的代码实现第N+1项的计算，但是与之相对的，这种算法的代价也非常的高，计算机每次调用函数时，为了保证函数执行完毕，都必须要将函数调用时的程序现场入栈保存，递归存在大量的自身调用，势必会产生非常庞大的函数现场栈来记录数据

以求解f(10)作为例子来分析递归求解的过程。要求得f(10)，需要求得f(9)和f(8)。同样，要求得f(9)，要先求得f(8)和f(7)……用树形结构来表示这种依赖关系：

         f(10)  
                /         \  
             f(9)            f(8)  
           /       \        /      \  
        f(8)    f(7)   f(7)   f(6)  
       /    \      /    \   
   f(7)   f(6)  f(6) f(5)

不难发现在这棵树中有很多结点会重复的，而且重复的结点数会随着n的增大而急剧增加。这意味这计算量会随着n的增大而急剧增大。用递归方法计算的时间复杂度是以n的指数的方式递增的。

CPU为i5 2.5GHZ，8G内存，windows7 64位操作系统，使用System.nanoTime()来计算消耗时间，一轮计算十次取平均耗时，三轮的平均耗时如下：（后续方法相同，不再赘述）

表1 直接递归实现在不同计算值下的执行时间

|  |  |
| --- | --- |
| 计算值 | 纳秒 |
| f(10) | 67615 |
| f(40) | 577718143 |
| f(80) | 大于8小时 |

当计算f(80)时，超过8小时未计算出结果。

3．自底向上实现

上述直接递归算法，为了求解f(n)，需要同时递归求解f(n - 1)和f(n - 2)，显然这样就做了大量的重复工作。采用自底向上的算法即可避免这样的冗余。从最小需计算的f（3）开始，逐步向上计算f(4)、f(5)等，将每步计算结果都保存下来，作为下次计算的初始值，要计算f(n)，则依次计算f0，f1，f2...fn，这时计算fn只需要利用前两个结果即可，这样算法效率提高到了O(n)。

f1

f2

f3

f4

…

图1 自底向上实现示意图

3.1　动态数组

Arraylist能动态的增加和减少元素，使用它可以保存每次计算的结果，但是它每次执行add等添加元素的方法，都会检查内部数组的容量是否不够了，如果是，它会以当前容量的两倍来重新构建一个数组，将旧元素copy到新数组，然后丢弃旧数组。因此效率不如数组，其核心代码如下：

public static Long arraylist\_cache(int n) {

if (n <= 2) return 1L;

List<Long> results = new ArrayList<Long>();

results.add(1L);

results.add(1L);

int length = -1;

while ((length = results.size()) < n)

results.add(results.get(length - 2) + results.get(length - 1));

return results.get(n - 1);}

3.2　数组实现

与arraylist类似，不过是将每次的计算结果保存到数组结构中，不再赘述。

3.3　swap实现

注意到以上两种方式缓存空间仅用于两次计算，因此可以使用新计算结果覆盖原有空间，节省空间。其核心算法如下：

while (index < n ) {

l = l2;

l2 = l1 + l2;

l1 = l;

index++;}

f1

f2

f3

f2

f3

f4

图2 swap实现示意图

自底向上算法各实现方式的消耗时间对比情况如下表：

表2 自底向上实现在不同计算值下的执行时间

|  |  |  |  |
| --- | --- | --- | --- |
| 计算值 | 动态数组 | 数组 | swap |
| f(40) | 76620 | 1464 | 2463 |
| f(80) | 104838 | 2395 | 3216 |
| f(92) | 111393 | 2723 | 3448 |

对比简单递归实现，计算时间减少了不少，而且随着计算值增大，节约时间越明显。

同时也可以看到不同的实现方式耗时不同，时间复杂度与实际执行时间并不是绝对正比关系。

4．比内公式实现

f(n)=f(n-1)+f(n-2)线性递推数列的特征方程为：

解得,

则

∵f(1)=f(2)=1

∴

解得 ，

∴

核心代码为

double temp = Math.sqrt(5.0);

return (long) ( (1/temp)\*(Math.pow((1+temp)/2,n)-Math.pow((1-temp)/2, n)) );

代码复杂度为O(1)，由于double类型的精度还不够，所以程序算出来的结果会有误差。

5．矩阵乘法实现

斐波那契数列的表达式可以写成如下两个形式：

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

f(n-1) =1\*f(n-1)+0\*f(n-2)

可以转换为矩阵表达：

因此，成为了矩阵的乘方问题。

若A为n×k矩阵，B为k×m矩阵，则它们的乘积AB(有时记做A•B)将是一个n×m矩阵。前一个矩阵的列数应该等于后一个矩阵的行数，得出的矩阵行数等于前一个矩阵的行数，列数等于后一个矩阵的行数。

其乘积矩阵AB的第i行第j列的元素为：

通过分治法或快速幂，算法效率提高到了O(logn)。所谓分治法就是：

快速幂就是利用结合律快速计算幂次的方法，应用到矩阵即可：

public static long[][] matrix\_pow(int n) {

long [][] res = { { 1, 1 }, { 1, 1 } };

long [][] a=UNIT;

while(n>0){

// matrixMultiply是矩阵乘法函数

if((n&1)==1) res = matrixMultiply(res,a);

a=matrixMultiply(a,a);

n=(n>>1);

}

return res;}

当n越大，相比较其他算法会节省更多时间。核心代码为矩阵乘法与分治法，此处不细述

表3 比内公式和矩阵乘法实现在不同计算值下的执行时间

|  |  |  |
| --- | --- | --- |
| 计算值 | Matrix | Formula |
| f(40) | 6569 | 2038 |
| f(80) | 7348 | 2422 |
| f(92) | 10687 | 2627 |

以上使用基本类型long，当n>92时，会超出long的表达范围，因此结果不正确，此时需要用BigInteger解决溢出问题，但会比long多一些耗时。

表3 各算法实现实现在BigInteger类型下的执行时间

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 计算值 | List | array | swap | Matrix | Formula |
| f(80) | 82997 | 48689 | 42518 | 100103 | 1207930 |
| f(800) | 506591 | 306645 | 383252 | 238591 | 7871161 |
| f(8000) | 5446817 | 3900261 | 4076532 | 1218439 | 107777421 |

可以看到计算值从f(80)到f(800)到f(8000)，10倍的增长，但矩阵法耗时并没有随着线性增长，性能最佳。

由于公式法小数需要使用BigDecimal，效率降低了很多，f(800)结果达到了，公式法精度存在问题，取整后结果已然不对，保留10000位小数时仅能保证前13位正确，f(8000)结果更是达到了的数量级。

6．结束语

直接使用递归算法虽能快速写出，且容易理解，但其耗费空间和时间都很大，使用自底向上算法可以将时间复杂度降低到O(n)，使用推导出来的公式虽能将时间复杂度降低到O(1)，但幂方运算依然耗时，而且在大数下无法保证精度。通过升维到二维矩阵计算，可以将时间复杂度降低到O(logn)，在数据比较大的情况下，节省时间比较明显。

参考文献(黑体，小五号)：

[1] 作者.书名[M].出版地：出版者，出版年. (字号:六号，楷体)

[2] 作者.文章题名[J].期刊名，年，卷(期)：起止页码. (字号:六号，楷体)

[3] 作者.文章题名[A].会议论文集名[C].出版地：出版社，出版年.起止页码. (字号:六号，楷体)

[4] 作者.学位论文名称[D].出版地：出版社，出版年. (字号:六号，楷体)

[5] 作者.报告名称[R].出版地：出版者，出版年. (字号:六号，楷体)

[6] 标准编号，标准名称[S].

[7] 专利所有者. 专利题名[P].专利国别: 专利号，出版日期.

[8] 主要责任者. 电子文献题名[EB/OL].出处或地址，发表日期/引用日期.

[9] 主要作者. 文献题名[Z].出版地：出版社，出版年.

[1]孔庆新. Fibonacci数的若干性质(Ⅲ)[J]. 青海师范大学学报(自然科学版),1991,(02):7-16. [2017-09-02]. DOI：10.16229/j.cnki.issn1001-7542.1991.02.002

[2]王瑾瑜. 斐波那契数列的几种解法介绍及优缺点分析[J]. 科技创新导报,2008,(30):241. [2017-09-02]. DOI：10.16660/j.cnki.1674-098x.2008.30.190

[3]宋庭武. 用特征方程推导斐波那契数列的通项公式[J]. 安庆师范学院学报(自然科学版),2010,16(04):91-92. [2017-09-02].

[4]斐波那契数列通项公式的求法[J]. 张新娟. 高等数学研究. 2009(04)

[1]李亚梅. 递归算法的分析与应用[J]. 中国新通信,2012,14(15):89-90. [2017-09-02].

[1]吴川,江海宁. 递归算法在程序设计中的应用分析[J]. 科技资讯,2010,(31):16+18. [2017-09-02]. DOI：10.16661/j.cnki.1672-3791.2010.31.150

[1]朱振元,朱承. 递归算法的非递归化实现[J]. 小型微型计算机系统,2003,(03):567-570. [2017-09-02].

[1]程基沛. 比内公式新推导[J]. 数学通报,1958,(07):12. [2017-09-02].

[5]王立波. 斐波那契查找算法的改进[J]. 杭州电子科技大学学报,2011,31(03):54-57. [2017-09-02]. DOI：10.13954/j.cnki.hdu.2011.03.020

[6]数据结构与算法[M]. 清华大学出版社 , 齐德昱编著, 2003

作者简介(黑体，六号)：

周卫星，1991，男，湖南永州人，硕士研究生，软件开发工程师。研究方向：J2EE架构，图像识别，计算机网络。

□□姓名（出生年－），性别，具体籍贯，学历，职称。研究方向：□□□□，□□□□，□□□□。(字号:六号，楷体)

（所有的数学算式，请用office word公式编辑器编写。）

深圳市福田区新洲南国通大厦11楼，518048，17896445100，[zwx1991jay@sina.com](mailto:zwx1991jay@sina.com)

联系地址、邮编、电话、E-mail。