参赛队员姓名：周　　尚

中学：北京师范大学附属实验中学国际部

省份：北京市

国家/地区：中国

指导教师姓名：黄彩英

论文题目：斐波那契字符串前缀和的 O(1) 算法及其证明

本参赛团队声明所提交的论文是在指导老师指导下进行的研究工作和取得的研究成果。尽本团队所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果。若有不实之处，本人愿意承担一切相关责任。

参赛队员：周尚　　指导老师：黄彩英

2020 年 9 月 11 日

斐波那契字符串前缀和的0(1)算法及其证明[1]

周　尚
北京师范大学附属实验中学

[摘要]本人在编写斐波那契字符串前缀和算法程序过程中，通过具体观察、抽象思维和程序验证等方式，结合斐波那契数列特点，提出了一种简单而奇妙的算法，即$S_n = \lfloor n\phi \rfloor$，$\lfloor\ \rfloor$表示下取整，$\phi$为黄金分割比$\left(\sqrt{5}-1\right)/2$，将计算的时间复杂度从$0(\log^2 n)$降为0(1)，运用数学归纳法予以证明，并得出了任意一段字符串的求和公式、任意一个字符是"0"还是"1"的计算公式等相关推论。

[关键词]斐波那契字符串；前缀和；0(1)算法

[Abstract] Through specific observation, guessing upon a hunch and program verification during programming the algorithm of prefix sum of Fibonacci words, the author puts forward a simple but fantastic formula, i.e. $S_n = \lfloor n\phi \rfloor$ **, where** $\lfloor x \rfloor$ **means the maximum integer not exceeding the number x,** ϕ **is the golden ratio i.e.**$\left(\sqrt{5}-1\right)/2$**. Using the formula, the time complexity of the algorithm is reduced from O(log^{2}n) to O(1). The main body of the paper is focused on proving the formula by mathematical induction. The author also deduces related results, such as the summation formula for any subword of the Fibonacci words, the formula for identification any character of the Fibonacci words, etc.**

[Keywords] Fibonacci words, prefix sum, O(1)algorithm

[1] 本文已刊发于《数学学习与研究》（2020.12 期）。

目录

1. 斐波那契字符串前缀和常见算法的时间复杂度

1.1 关于算法的时间复杂度

一个算法的语句执行总次数 T(n)是关于问题规模 n 的函数，算法的时间复杂度就是分析 T(n)随 n 的变化情况，确定 T(n)的数量级，通常记为：T(n)=O(g(n))，其中，g(n)是问题规模 n 的某个函数。它表示随问题规模 n 的增大，算法执行时间的增长率和 g(n)的增长率相同。一般地，T(n)关于 n 的数量级越低，算法越优。

1.2 斐波那契字符串前缀和常见算法的时间复杂度

众所周知，斐波那契数列是：1，1，2，3，5，8，…，即 $f_1 = 1$，$f_2 = 1$，$f_i = f_{i-1} + f_{i-2}$，$i > 2$.

与此相关的斐波那契字符串的定义如下：

$f(0) =″0″$，$f(1) =″1″$，$f(i) = f(i-2) \oplus f(i-1)$，$i > 1$，

$F = f(0) \oplus f(1) \oplus f(2) \oplus \ldots =″0101101011011010110 1\ldots″$ 称为无限斐波那契字符串，其中，双引号中的 0、1 均为字符，⊕表示字符串的拼接。

令 $S(l, r)$ 为无限斐波那契字符串 F 的第 l 个字符至第 r 个字符中″1″的个数，则前 n 个字符中″1″的个数为 $S(1, n)$ 简记为 S_n，即前缀和。

要求以尽量低的时间复杂度计算前缀和 S_n。目前常见算法[2]的时间复杂度为 $O(\log^2 n)$。

[2] https://blog.csdn.net/qq_39440588/article/details/8151678; https://www.cnblogs.comYYC-304/p9500001.html.

2. 时间复杂度为 O(1)的算法的提出

笔者先用时间复杂度为 $0(\log^2 n)$的常用方法编写程序，运行程序得出了 S_5、S_{10}、S_{100}等具体的数值，当观察到$S_{1000} = 618$时，就猜测这可能与黄金分割点有关。然后发挥找规律的特长，结合斐波那契数列的特性，提出并证明了一种时间复杂度为 0(1)的奇妙算法，即$S_n = \lfloor n\phi \rfloor$，$\lfloor\ \rfloor$表示取整，$\phi$为黄金分割比$\left(\sqrt{5}-1\right)/2$。该公式从形式上就有些不可思议，左边肯定是一个正整数，而有右边是一个无理数与 n 相乘，再进行一个下取整运算。

如果该算法能够得到证明，它将极大地降低计算的时间复杂度，成为时间复杂度最低的算法。不妨以 n=16 为例，前 16 个字符"0101101011011010"中有 9 个"1"，代入公式

$$S_{16} = \lfloor 16 * (\sqrt{5}-1)/2 \rfloor = 9$$

得到验证。笔者通过计算机程序验证$n \le 10^{18}$情形均成立，计算程序见附录。

3. 算法证明

定义：

令f_i为斐波那契数列第 i 项,

F_i为斐波那契数列前 i 项之和，即$\sum_{j=1}^{i} f_j$，

显然，$f_{i-2} + f_{i-1} = f_i$，$f_{i+2} - 1 = F_i$.

令 $\overline{f}_i = \max_{f_j<i}(f_j), \forall j$,

$\overline{F}_i = \max_{F_j<i}(F_j), \forall j$，不妨记$\overline{F}_i$在数列$F_j$中对应于$F_{c_i}$.

引理 1： $S_n = \begin{cases} \sum_{i=1}^{t}(S_{F_{c_i}} - S_{F_{c_i-2}}) + S_1 \text{.....若} a_n = 0 \\ \sum_{i=1}^{t}(S_{F_{c_i}} - S_{F_{c_i-2}}) + S_2 \text{.....若} a_n = 1 \end{cases}$，t 为最大分解次数，

a_n 为斐波那契字符串前 n 个字符的最后一位，对于$\forall i \neq j, i, j \in [1, t]$,有$F_{c_i} \neq F_{c_j}$。

证明：

根据前述定义及内在关系，可将S_n作如下分解：

$$\begin{aligned} S_n &= S_{\overline{F}_n} + S(\overline{F}_n + 1, n) \\ &= S_{F_{c_n}} + S(F_{c_n} + 1, n) \\ &= S_{F_{c_n}} + S(F_{c_n-2} + 1, n - f_{c_n-1} - f_{c_n}) \\ &= S_{F_{c_n}} + S(F_{c_n-2} + 1, n - f_{c_n+1}) \\ &= S_{F_{c_n}} + S_{n-f_{c_n+1}} - S_{F_{c_n-2}} \\ &= S_{n-f_{c_n+1}} + (S_{F_{c_n}} - S_{F_{c_n-2}}) \end{aligned} \quad \cdots\cdots\cdots\cdots\cdots\cdots (1)$$

对（1）式中$S_{n-f_{c_n+1}}$进行重复分解，令$n - f_{c_n+1} = m$，则

$$S_{n-f_{c_n+1}} = S_m = S_{m-f_{c_m+1}} + (S_{F_{c_m}} - S_{F_{c_m-2}}).$$

假设经过 t 次分解后出现S_1（若$a_n = 0$）或S_2（若$a_n = 1$），分解结束，可得

$$S_n = \begin{cases} \sum_{i=1}^{t} (S_{F_{c_i}} - S_{F_{c_i-2}}) + S_1 \cdots\cdots 若 a_n = 0, \\ \sum_{i=1}^{t} (S_{F_{c_i}} - S_{F_{c_i-2}}) + S_2 \cdots\cdots 若 a_n = 1. \end{cases}$$

显然，对于$\forall i \neq j, i, j \in [1, t]$,有$F_{c_i} \neq F_{c_j}$.

引理 1 得证。

定义：令$r_i = f_{i+2}\phi - f_{i+1}$，$R_i = \{F_i\phi\}$，$\{\ \}$表示取小数部分。

引理 2：数列r_i的奇数项单调递减，偶数项单调递增，且$\lim_{i\to\infty} r_i = 0$.

证明：由定义可知，

$r_0 = \phi - 1$，

$r_1 = f_3\phi - f_2 = 2\phi - 1$，

$r_2 = f_4\phi - f_3 = 3\phi - 2$，

$r_i = r_{i-1} + r_{i-2}$　当 i ≥ 2 时，

下面用数学归纳法证明 $r_i = (-\phi)r_{i-1}$.

当 i=2 时， $r_2 = 3\phi - 2 = (-\phi)r_1$,

假设 i=k-1 时， $r_{k-1} = (-\phi)r_{i-2}$ 成立，则

$$
\begin{aligned}
r_k &= r_{k-1} + r_{k-2} \\
&= (-\phi)r_{k-2} + r_{k-2} \\
&= (1 - \phi)r_{k-2} \\
&= \phi^2 r_{k-2} \\
&= (-\phi)\,(-\phi)r_{k-2} \\
&= (-\phi)r_{k-1}.
\end{aligned}
$$

即当 i=k 时也成立。

由此可知， $r_i = (-\phi)^i(\phi - 1)$.

可见，数列 r_i 的奇数项单调递减，偶数项单调递增，且 $\lim_{i\to\infty} r_i = 0$ ，引理 2 得证。

引理 3：数列 R_i 的奇数项单调递减，偶数项单调递增，且 $\lim_{i\to\infty} R_i = 1 - \phi$.

证明：由引理 2 可得

$\phi - 1 < r_i < \phi$，即 $\phi - 1 < f_{i+2}\phi - f_{i+1} < \phi$,

每一项同时减去 $(\phi - 1)$，得

$$0 < (f_{i+2} - 1)\phi - f_{i+1} + 1 < 1$$

由于 $F_i = f_{i+2} - 1$，上式可变为

$$0 < F_i\phi - f_{i+1} + 1 < 1$$

因为 $- f_{i+1} + 1$ 是整数，所以

$$
\begin{aligned}
\{F_i\phi\} &= F_i\phi - f_{i+1} + 1 \\
&= (f_{i+2} - 1)\phi - f_{i+1} + 1 \\
&= f_{i+2}\phi - f_{i+1} + 1 - \phi
\end{aligned}
$$

即 $R_i = r_i + 1 - \phi$.

因此，数列 R_i 的奇偶项单调性同数列 r_i， $\lim_{i\to\infty} R_i = 1 - \phi$，引理 3 得证。

算法证明

下面用第二数学归纳法证明$S_n = \lfloor n\phi \rfloor$.

当 k=1 时，$S_1 = 0 = \lfloor \phi \rfloor$,

当 k=2 时，$S_2 = 1 = \lfloor 2\phi \rfloor$,

假设 k<n 时，$S_k = \lfloor k\phi \rfloor$成立，

当$k = n$时，由引理 1 可知，

$$\lfloor n\phi \rfloor - S_n = \lfloor n\phi \rfloor - (\sum_{i=1}^{t}(S_{F_{c_i}} - S_{F_{c_i-2}}) + S_1 \text{or} S_2)$$

$$= \lfloor n\phi \rfloor - (\sum_{i=1}^{t}(\lfloor F_{c_i}\phi \rfloor - \lfloor F_{c_i-2}\phi \rfloor) + \lfloor \phi \rfloor \text{or} \lfloor 2\phi \rfloor)$$

$$= n\phi - \{n\phi\} - (\sum_{i=1}^{t}(F_{c_i}\phi - F_{c_i-2}\phi) + \phi or 2\phi - \sum_{i=1}^{t}(\{F_{c_i}\phi\} - \{F_{c_i-2}\phi\}) + \{\phi\} or \{2\phi\})$$

由于分解过程中每次拆分出的 S 下标和相等，因此上式可变为

$$\lfloor n\phi \rfloor - S_n = n\phi - \{n\phi\} - (n\phi - \sum_{i=1}^{t}(\{F_{c_i}\phi\} - \{F_{c_i-2}\phi\}) - \{\phi\} \text{or} \{2\phi\})$$

$$= -\{n\phi\} + (\sum_{i=1}^{t}(\{F_{c_i}\phi\} - \{F_{c_i-2}\phi\}) + \{\phi\} \text{or} \{2\phi\})$$

$$= -\{n\phi\} + (\sum_{i=1}^{t}(R_{c_i} - R_{c_i-2}) + \{\phi\} \text{or} \{2\phi\})$$

利用引理 3，对上式进行缩放可得，

$$\sum_{i=1}^{t}(R_{c_i} - R_{c_i-2}) < \sum_{\substack{c_i\text{为偶数} \\ \text{且} c_i \geq 2}}^{\infty}(R_{c_i} - R_{c_i-2}) < \lim_{i\to\infty} R_i - R_0 = 1 - \phi,$$

$$\sum_{i=1}^{t}(R_{c_i} - R_{c_i-2}) > \sum_{\substack{c_i\text{为奇数} \\ \text{且} c_i \geq 3}}^{\infty}(R_{c_i} - R_{c_i-2}) > \lim_{i\to\infty} R_i - R_1 = 1 - 2\phi,$$

即，$1 - 2\phi < \sum_{i=1}^{t}(R_{c_i} - R_{c_i-2}) < 1 - \phi$.

又因为，$\{\phi\} = \phi > 2\phi - 1 = \{2\phi\}$

所以 $0 < \sum_{i=1}^{t}(R_{c_i} - R_{c_i-2}) + \{\phi\} \text{or} \{2\phi\} < 1$

因为$0 \leq \{n\phi\} < 1$，所以$0 \leq |\lfloor n\phi \rfloor - S_n| < 1$

因为$\lfloor n\phi \rfloor$，S_n皆为整数，所以$\lfloor n\phi \rfloor - S_n = 0$，即$S_n = \lfloor n\phi \rfloor$

得证。

4. 相关推论

4.1 斐波那契字符串第 1 个字符至第 r 个字符中"1"的个数为 $S(l,r) = S_r - S_{l-1} = \lfloor r\phi \rfloor - \lfloor (l-1)\phi \rfloor$.

4.2 斐波那契字符串中第 n 个字符 a_n，即当 l=r=n 时，$a_n = S(n,n) = S_n - S_{n-1} = \lfloor n\phi \rfloor - \lfloor (n-1)\phi \rfloor$.

4.3 任取 n，前 n 项的平均数 $S_n / n = \lfloor n\phi \rfloor / n < \phi$，且 $\lim_{n\to\infty}(S_n / n) = \phi$.

4.4 任取 n，数列 a_i 前 n 项的方差 σ_n^2 小于 $2\phi - 1$，且趋于 $2\phi - 1$.

证明：

$$\sigma_n^2 = \sum_{i=1}^{n} (a_i - S_n / n)^2 / n$$

$$= \frac{\sum_{i=1}^{n} a_i^2 - 2(S_n / n)\sum_{i=1}^{n} a_i + n(S_n / n)^2}{n}$$

由于 a_i=0 或 1，且前 n 项中有 S_n 个 1，所以上式可化为

$$\sigma_n^2 = \frac{\sum_{i=1}^{n} a_i^2 - 2(S_n / n)\sum_{i=1}^{n} a_i + n(S_n / n)^2}{n}$$

$$= \frac{S_n - 2(S_n / n)S_n + S_n^2 / n}{n}$$

$$= \frac{S_n - S_n^2 / n}{n}$$

$$= \frac{S_n}{n} - (\frac{S_n}{n})^2$$

$$\lim_{n\to\infty} \sigma_n^2 = \phi - \phi^2 = 2\phi - 1.$$

5. 附录：时间复杂度为 O(1)算法程序

```
#include<iostream>
using namespace std;
const long long G[3]={61803398,87498948,48204587},P=1e8;
long long n,f[3],A[5];
int main()
{
  cin>>n;
  f[0]=n/P/P,f[1]=n/P%P,f[2]=n%P;
  for(int i=0;i<3;i++)
      for(int j=0;j<3;j++)
          A[i+j]+=f[i]*G[j];
  for(int i=4;i>0;i--)
      A[i-1]+=A[i]/P,A[i]%=P;
  cout<<A[0]*P+A[1];
  return 0;
}
```

6. 致谢

本项研究是本人独立完成的。但在学习研究过程中，胡伟栋老师作为启蒙老师，将我引入信息学的殿堂，让我有机会接触信息学与数学的交叉学科问题；毕业于英国拉夫堡大学数学专业的张曦征博士对论文结构的完整性和论证的规范性提出了专业意见；数学老师王彩英给了我很多精神上的鼓励，激发了数学学习兴趣，并对论文的终稿进行了最终检查和润色。在此，本人对他们的专业指导和无私奉献表示衷心感谢！

参考文献：

无直接相关参考文献。

参赛队员介绍

周尚，男，北京师范大学附属实验中学国际部高三学生。思维能力突出，擅长数学和计算机编程，尤其擅长组合类问题，多次参加竞赛并获奖，逐渐养成用数学和计算机思维解决问题的习惯。擅长围棋，业余 5 段、弈城 9 段，担任学校棋社社长。

•2020 年 8月中国信息学奥林匹克竞赛决赛（NOI）铜牌。

•2020 年 8月亚洲和太平洋地区信息学奥林匹克竞赛（APIO）银牌。

•2020 年 3 月美国信息学奥林匹克竞赛(USACO)公开赛全球第 8 名。

•2019 年 11月CCF CSP-S2 组认证 一等奖。

•2019 年 3 月加拿大计算机竞赛中国赛区（CCC）第 14 名。

•2018 年 11 月中国信息学联赛（NOIP）北京市提高组一等奖。

•2017 年 3 月全国初中数学联赛二等奖