投稿類別:資訊類

篇名:

Fibonacci series 對質數 K 次方模運算之循環關係

作者:
古珉和。臺南一中。高二 18 班

指導老師:
高英耀老師

壹●前言

一、 研究動機

本篇的研究動機為，在參加青年程式設計競賽時，有一題目其要求為求解 Fibonacci series 第 N 項對 2 的 k 次方取餘數之值，而此要求之解法必須假定費波納契數列對 2 的 k 次方取餘數之值會循環，且此循環存在一定規律，故現場使用窮舉法將循環規律解出，並用動態規劃法求出其正解。

在回程途中，突然萌生了一個想法，既然最小質數之循環規律存在，且已經被證明為正確，且質數之間時常有類似的性質，那其他質數有無類似的循環規律，即成為了一個值得研究之議題，於是就開始此議題之研究，此即為此篇之研究動機。

二、 研究範圍及目的

在研究範圍方面，第一部分先探討各質數本身有無循環個數上關係

$$F(N) \equiv F(N+R) \pmod{P}$$

P 為輸入之質數，R 為本篇要探討之循環值。

如有關係，找出敘述其規律之一般式，並加以證明。

第二部分探討單一質數各次方之間有無循環個數上關係

$$F(N) \equiv F(N+R) \pmod{P^k}$$

P 為輸入之質數，R 為本篇要探討之循環值，K 為輸入之次方。

如有關係，找出敘述其規律之一般式，並加以證明。

第三部分討論各質數及各次方間之關係，以期能夠解得其一般式。

本篇之研究目的即為找出各質數之循環個數之規律，再進而找出單一質數各次方間之關係，從而推導出能表示其循環個數狀態之一般式，進而證明其一般式之正確性。

三、 研究方法

(一) 流程

先了解 Fibonacci Series 之性質，參考 Fibonacci Series 之定義及運用 C++程式語言編寫一程式，接著編寫質數建表之函式，將結果紀錄至一陣列之中，並使用一巢狀迴圈，分別呼叫出質數 P 以及次方 k，使其枚舉出 Fibonacci series 各項對 P 的 k 次方取餘數之值，並在出現循環時，跳出迴圈，最後將結果輸出至一文字檔中以方便觀察。

(二) 工具

1. 硬體
   本專題使用於編寫程式及撰寫本文之硬體為個人電腦。
2. 軟體
   本專題使用於編寫程式之 IDE(整合式開發環境)為 Dev-C++
3. 程式語言
   本專題所提及之所有程式，使用語言均為 C++語言，語言標準為 C++11。

貳●正文

一、背景知識

(一) Fibonacci series 及其數學之美

1. Fibonacci series 介紹

Fibonacci series 根據維基百科之敘述，首先提出此數列之學者為印度數學家 Gopala 和金月，他們在西元 1150 年在研究箱子物長寬剛好為 1 和 2 的可行方法數目時，首先描述這個數列。而在歐洲，Leonardo Pisano Bigollo 在描述兔子生長時首先使用了此數列。

在數學上，Fibonacci series 之定義為

$$\begin{cases} f(0)=0 \\ f(1)=1 \\ f(n)=f(n-1)+f(n-2)\ ,n\geq 2 \end{cases}$$

用文字來說，就是 Fibonacci series 由 0 和 1 開始，之後的 Fibonacci series 就是由之前的兩數相加而得出。

2. 數學之美

Fibonacci series 從第二項開始除以前一項之值會漸漸趨近於黃金比例 1.618:1 或是 1:0.618(即$2\cos 36°$)，也就是說

$\lim_{n\to\infty}\frac{F(n)}{F(n-1)}$=1.618，而在自然界中，人體從腳底至頭頂之距離和從肚臍至腳底之距趨近於黃金比例，向日葵的種子螺旋排列有 99%是 F(n)。

(二) 模運算及其應用

1. 模運算

根據培養與鍛鍊程式設計的邏輯腦一書在第 126 頁所述計算除以 m 的餘數又稱為「對 m 取 mod」或「對 m 取模」或「以 m 取模」。下面將一律採用「以 m 取 mod」這種說法。為了讓程式碼更簡潔，以 m 取整數 a 與 b 取餘數若相等則寫成$a \equiv b(\mathrm{mod}\ m)$。另外，以 m 除 a 的餘數要寫成 a mod m。透過簡單的計算，只要

$a \equiv c(\text{mod } m)$、$b \equiv d(\text{mod } m)$，就可以說

$$a + b \equiv c + d(\text{mod } m)、a - b \equiv c - d(\text{mod } m)$$
$$a \times b \equiv c \times d(\text{mod } m)$$

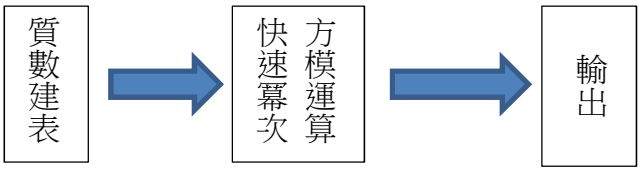
是成立的。」

2. 模運算應用

根據台大電機系鄭振牟副教授在密碼學與模算術一文中所述：「**RSA 的核心運算為模指數運算。**」可知模運算可用於 RSA 加密法，並且在加密法中佔有重要的一席之地。

二、演算法設計說明

(一) 演算法流程圖



(二) 取質數方法

本節將探討質數建表使用之演算法

1. sieve of Eratosthenes

sieve of Eratosthenes 中文稱埃拉托斯特尼篩法，是目前所知取質數較為快速之演算法，所使用的原理是從 2 開始，將每個質數的各個倍數，標記成合數。

(三) 次方運算

1. 基礎算法

一般時候我們在計算冪次方時，一般是採用冪次方之基礎算法，即將欲運算之底數連乘欲運算之冪次數 k，而這在冪次極大時可能極慢。複雜度 O(k)。

2. 快速冪算法

根據培養與鍛鍊程式設計的邏輯腦一書在第 127 頁所述

由於當$n = 2^k$時可將之表示為$x^n = (x^2)^2$ …… 如此便能以連乘 k 次的平方來輕鬆求出冪次方。將這種想法放在心上，並將 n 表示為 2 的冪次方的和就行了。$n = 2^{k_1} + 2^{k_2} + 2^{k_3}$ ……這樣一來就會變成$x^n = x^{2^{k_1}} + x^{2^{k_2}} + x^{2^{k_3}}$ ……只要依序求取x^{2^i}並同時進行計算就行了，這樣結果就會變成是以$O(\log_2 n)$的複雜度來求取冪次方了。

可知，此法可以有效降低時間複雜度。

三、程式碼簡介及分析

本段將會對程式碼本身進行詳細介紹，並且針對其時間複雜度以及空間複雜度進行分析。

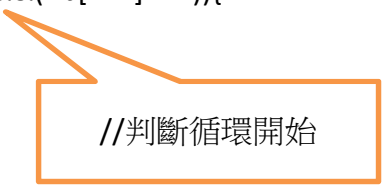
(一) 程式碼

```
1.  #include<bits/stdc++.h>
2.  #define inf 10005                //Fibonacci series 大小
3.  #define maxn 1000000             //尋找 maxn 以內的質數
4.  using namespace std;
5.  bool np[maxn];
6.  int prime[maxn],cnt=0;
7.  void p(){
8.      for(int i=0;i<maxn;++i) np[i]=0;
9.      np[0]=np[1]=true;
10.     for(int i=2;i<maxn;++i){
11.         if(!np[i]){
12.             prime[cnt]=i;cnt++;
13.             for(int j=i+i;j<maxn;j+=i){
14.                 np[j]=true;
15.             }
16.         }
17.     }
18. }
19.
20. int fast_pow(int n,int m){
21. int ans=1;
22.     while(m>0) {
23.         if(m % 2 == 1)
24.             ans = (ans * n);
25.         m = m/2;
26.         n = (n * n) ;
27.     }
28.     return ans;
29. }
30. int fib[inf];            //  Fibonacci series 陣列
31. int dp[505][15];         //循環個數陣列
32. int main(){
33.     p();
34.     int num,power;
35.     for(int i=0;i<500;i++){                    //取前 500 個質數
```

質數建表

快速冪

```
36.             num=prime[i];
37.             for(int j=1;j<10;j++){                    //計算 1-9 次方
38.                 power=j;
39.                 int mod=fast_pow(num,power); //模數以快速冪計算
40.                 fib[0]=0;
41.                 fib[1]=1;
42.                 for(int k=2;k<inf;k++){
43.                     fib[k]=(fib[k-1]+fib[k-2])%mod;
44.                 }
45.                 for(int k=3;k<inf;k++){          //建循環個數表
46.                     if((fib[k]==0)&&(fib[k+1]==1)&&(fib[k+2]==1)){
47.                         dp[i][j]=k;
48.                         break;
49.                     }
50.                 }
51.             }
52.         }
53.         for(int i=0;i<500;i++){                   //輸出
54.             cout<<prime[i]<<" ";
55.             for(int j=1;j<10;j++){
56.                 cout<<dp[i][j]<<" ";
57.             }
58.             cout<<endl;
59.         }
60.         return 0;
61.  }
```



(二) 程式碼各項分析的定義

1. Big O

Fundamental Of Data Structure In C 對 Big O 有以下介紹

Each statement is counted once, so step 9 has 2 statements and is executed once for a total of 2. Clearly,the actual time taken by each statement will vary. The for statement is really a combination of several statements, but we will count it as one.

The total count then is 5n + 5. We will often write this as O(n),ignoring the two constants 5. This notation means that the order of magnitude is proportional to n.

We write O(1) to mean a computing time which is a constant. O(n) is called linear, O(n^2) is called quadratic, O(n^3) is called cubic, and

$O(2^n)$ is called exponential. If an algorithm takes time O(log n) it is faster, for sufficiently large n, than if it had taken O(n). Similarly, O(n log n) is better than $O(n^2)$ but not as good as O(n). These seven computing times, O(1), O(log n), O(n), O(n log n), $O(n^2)$, $O(n^3)$, and $O(2^n)$ are the ones we will see most often throughout the book.

2. 時間複雜度

Fundamental Of Data Structure In Pascal 對時間複雜度做出了以下定義：「**Definition: The time complexity of a program is the amount of computer time it needs to run to completion.**」

3. 空間複雜度

Fundamental Of Data Structure In Pascal 對空間複雜度做出了以下定義：「**Definition: The space complexity of a program is the amount of memory it needs to run to completion.**」

(三)程式碼分析

1. 時間複雜度

本程式使用快速冪來計算次方，所以計算次方部分之複雜度大概是 $O(\log_2 k)$，而之後再在建表時，跑了一個三重巢狀迴圈，長度大概為 500(質數數量)、次方數 1-9 以及 Fibonacci series 大小 inf，所以此部分之時間複雜度為 O(500*9*inf)，故總時間複雜度大約為 $O(500*9*(inf+\log_2 k))$。

2. 空間複雜度

而根據以上定義，本程式在全域中使用了一個長度為 inf 的陣列用以儲存 Fibonacci series，以及兩個長度為 maxn 的陣列用以儲存質數，故本程式之空間複雜度為 O(inf+maxn*2)。

四、研究結果

本段將陳列研究之結果

(一) 各質數循環個數之間之關係

1. 2 之循環個數

圖一、2 之循環狀況

圖一為使用程式所計算出 2 之循環狀況，由圖一我們可以輕易地看出循環個數是 3，這也驗證了 Fibonacci series 之排列為偶數、奇數、

奇數之規則。

2. 3 之循環個數

圖二、3 之循環狀況

圖二為使用程式所計算出 3 之循環狀況，由圖三我們可以得到一個結論--循環個數為 8，並且我們也得到另一性質，即確定 Fibonacci series 對質數取餘數會有循環關係。

3. 其餘質數之循環個數

表一、各質數之循環結果

質數 P	2	3	5	7	11	13	17	19	23
循環個數	3	8	20	16	10	28	36	18	48
質數 P	29	31	37	41	43	47	53	59	61
循環個數	14	30	76	40	88	32	108	58	60
質數 P	67	71	73	79	83	89	97	***101***	103
循環個數	136	70	148	78	168	44	196	***50***	208
質數 P	107	109	***113***	127	131	137	139	149	151
循環個數	72	108	***76***	256	130	276	46	148	150

上表一為將前 36 個質數本身輸入至本文第一段所描述之程式所得之結果整理而得之表格，而由上表一可以推論出以下的規律。令 M=循環個數、P=質數，當 P=2,M=3，當 P=5,M=20

當尾數是 1 時，除 101+60X 外皆遵守 M=P-1。(如表一紅色字)
當尾數是 3 時，除 113+90X 外皆遵守 M=2*P+2。(如表一黃色字)
當尾數是 7 時，除 47+60X 外皆遵守 M=2*P+2。
當尾數是 9 時，除 89+50X 外皆遵守 M=P-1。

(二) 單一質數各次方間之關係

1. 2 的各次方

表二、2 的各次方之循環個數

次方	1	2	3	4	5	6	7	8	9
個數	3	6	12	24	48	96	192	384	768

上表二為將輸入之質數定 2，而次方為 1 至 9 之正整數，經由程式計算輸出之結果整理而得之表格，而我們可以發現這是一個公比為 2 之等比數列，由此可推論質數各次方循環是有一定之關係的。

2. 3 的各次方

表三、3 的各次方之循環個數

次方	1	2	3	4	5	6	7	8	9
個數	8	24	72	216	648	1944	5832	17496	52488

上表三為將輸入之質數定為 3，而次方為 1 至 9 之正整數，經由程式計算輸出之結果整理而得之表格，而我們可以發現這是一個公比為 3 之等比數列，符合上述推論。

3. 5 的各次方

表四、5 的各次方之循環個數

次方	1	2	3	4	5	6	7	8	9
個數	20	100	500	2500	12500	62500	312500	1562500	7812500

上表四為將輸入之質數定為 5，而次方為 1 至 9 之正整數，經由程式計算輸出之結果整理而得之表格，而我們可以發現這是一個公比為 5 之等比數列，符合上述推論。

4. 通式

由窮舉法可做出以下推論，如令質數為 P、次方為 K、M 為 P 的 1 次方循環個數，則循環個數之通式為$\mathrm{M} \times P^{K-1}$。

(三) 綜合討論

由以上結果可畫出如下表五，表五之橫軸為質數 P，縱軸為次方 K，由此表可見，結果符合通式$\mathrm{M} \times P^{K-1}$。

表五、各質數及各次方

質數 P / 次方 K	2	3	5	7	11	13	17
1 (M)	3	8	20	16	10	28	36

2 (M × P^1)	6 (3*2^1)	24 (8*3^1)	100 (20*5^1)	112 (16*7^1)	110 (10*11^1)	364 (28*13^1)	612 (36*17^1)
3 (M × P^2)	12 (3*2^2)	72 (8*3^2)	500 (20*5^2)	784 (16*7^2)	1210 (10*11^2)	4732 (28*13^2)	10404 (36*17^2)
4 (M × P^3)	24 (3*2^3)	216 (8*3^3)	2500 (20*5^3)	5488 (16*7^3)	13310 (10*11^3)	61516 (28*13^3)	176868 (36*17^3)
5 (M × P^4)	48 (3*2^4)	648 (8*3^4)	12500 (20*5^4)	38416 (16*7^4)	146410 (10*11^4)	799708 (28*13^4)	3006756 (36*17^4)
6 (M × P^5)	96 (3*2^5)	1944 (8*3^5)	62500 (20*5^5)	268912 (16*7^5)	1610510 (10*11^5)	10396204 (28*13^5)	51114852 (36*17^5)
7 (M × P^6)	192 (3*2^6)	5832 (8*3^6)	312500 (20*5^6)	1882384 (16*7^6)	17715610 (10*11^6)	135150652 (28*13^6)	868952484 (36*17^6)

參●結論

一、結論說明

(一) 質數間的關係

在質數間的關係，同樣尾數的質數之間通常會遵守一定的關係式，而且尾數相加等於 10 之質數通常也會遵守一樣的規律。

(二) 單一質數各次方間之關係

在質數各次方間之關係，其循環個數成等比數列，且公比為質數本身。

二、研究過程中所提出之問題

(一) 關於一般式

既然質數之間遵守一定的規律，那有沒有辦法推導出其一般式來表示所有的結果?

(二) 關於證明

那如果有一般式的話，能不能用窮舉法以外之數學方法去證明其一般式之正確性，以及更加推廣。

三、未來值得研究的方向

今天我們發現了質數之間之規律，那可以繼續往合數，正整數，甚至是整數的領域進行推廣。

四、創見

因為其規律以及其循環,以後這可能在密碼學上產生一種加密法，從而成為資訊安全領域中重要的一環。

肆●引註資料

一、書籍

(一) Sartaj Sahni, Ellis Horowitz(1990)。Fundamental Of Data Structure In C。Computer Science Press。

(二) 吳振奎(2000)。斐波那契數列。九章出版社。

(三) 秋葉拓哉，岩田陽一，北川宜稔(2013)。培養與鍛鍊程式設計的邏輯腦。博碩出版社，2013 年 11 月。

二、期刊

(一) 鄭振牟(2011)。密碼學與模算術。臺大電機系科普系列，2011 年 7 月，取自 https://www.ee.ntu.edu.tw/hischool/doc/2011.07.pdf。

三、網路

(一) 維基百科(無日期)。斐波那契數列。2017 年 2 月 24 日，取自 https://zh.wikipedia.org/wiki/斐波那契數列

(二) 維基百科(無日期)。黃金分割率。2017 年 2 月 24 日，取自 https://zh.wikipedia.org/wiki/黃金分割率