投稿類別:資訊類

篇名:

Fibonacci series 對質數 K 次方取餘數循環關係

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 項對二的 k 次方取餘數之值($F(N) \equiv F(N+R) \pmod{2^k}$)，而此要求之解法必須假定費波納契數列對二的 M 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以及各次方間之關係，以期能夠解得其一般式。

那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

先了解 Fibonacci Series 之性質，參考 Fibonacci Series 之定義及運用 C++程式語言編寫一條程式，輸入兩變數，一為要探討之質數 P，另一變數為要探討之次方 k，使其輸出 Fibonacci series 各項對 P 的 k 次方取餘數之值，並在出現循環時，輸出一星號，最後將結果輸出至一文字檔中以方便觀察。

貳●正文

以下將會詳述我各個部分的研究過程以及研究結果

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 F(0)=0，F(1)=1，F(N)=F(N-2)+F(N-1)($2 \leq N$) 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) 快速冪原理

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，此發可以有效降低時間複雜度

(三) 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。讓 a mod m 變成 $0 \le a \bmod m \le m-1$。由於執行環境的問題，a 為負時 a%m 也會是負的，此時要改成 a%m+m 就能讓它包含在 0~m-1 的範圍。透過簡單的計算，只要$a \equiv c(\mathrm{mod}\ m)$、$b \equiv d(\mathrm{mod}\ m)$，就可以說 $a+b \equiv c+d(\mathrm{mod}\ m)$ $a-b \equiv c-d(\mathrm{mod}\ m)$ $a \times b \equiv c \times d(\mathrm{mod}\ m)$是成立的。」

2. 模運算應用

(1) 判別奇偶數

奇偶數的判別是模運算最基本的應用，也非常簡單。已知一個整數 n 對 2 取模，如果餘數為 0，則表示 n 為偶數，否則 n 為奇數。

(2) 判別質數

一個數，如果只有 1 和它本身兩個因數，這樣的數叫做質數。例如 2，3，5，7 是質數，而 4，6，8，9 則不是，後者稱為合成數或合數。

判斷某個自然數是否是素數最常用的方法就是試除法：用比該自然數的平方根小的正整數去除這個自然數，若該自然數能被整除，則說明其非質數。

二、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) 程式碼分析

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.

The notation f(n) = O(g(n)) (read as f of n equals big-oh of g of n) has a precise mathematical definition.

Definition: f(n) = O(g(n)) iff there exist two constants c and no such that |f(n)| c|g(n)| for all n no.

f(n) will normally represent the computing time of some algorithm. When we say that the computing time of an algorithm is O(g(n)) we mean that its execution takes no more than a constant times g(n). n is a parameter which characterizes the inputs and/or outputs. For example n might be the number of inputs or the number of outputs or their sum or the magnitude of one of them. For the Fibonacci program n represents the magnitude of the input and the time for this program is written as T(FIBONACCI) = O(n).

We write O(1) to mean a computing time which is a constant. O(n) is called linear, O(n2) is called quadratic, O(n3) is called cubic, and O(2n) 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(n2) but not as good as O(n). These seven computing times, O(1), O(log n), O(n), O(n log n), O(n2), O(n3), and O(2n) are the ones we will see most often throughout the book.」

2. 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.」**
而根據以上定義，本程式在主函式中使用了一變數紀錄質數，一變數紀錄次方，一變數紀錄質數的某次方，而在快速冪函式中使用了一變數紀錄算出之答案，一變數紀錄質數，一變數紀

錄次方，在全域中使用了一長度為 inf 之陣列，故本城市之空間複雜度為 O(inf+1+1+1+1+1+1)。

3. 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.**」

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($\log_2 k$)，而之後再建 Fibonacci series 表時，跑了一個長度為 inf 的迴圈，所以此部分之時間複雜度為 O(inf)，故總時間複

雜度大約為 O(inf+$\log_2 k$)

(二) 程式碼簡介

1. 程式碼介紹

本程式首先引入 stdc++標頭檔，再使用 std 命名空間以便於使用 cin 及 cout 指令，之後使用一名為 inf 之常數以表示 Fibonacci series 表之長度，接著編寫一函式，用於計算快速冪，而使用快速冪的目的為將原本 O(k)的時間複雜度降低為 O($\log_2 k$)，進而在程式執行時間上進行壓縮。

接下來使用一全域陣列名為 fib,其大小為 inf 其作用為儲存費 Fibonacci series 對質數的指定次方取餘數之結果，在 main 函式當中，首先使用兩變數用以紀錄質數(n)以及次方(m)，接著輸入此二變數，使用上述之快速冪函式計算出 n^m 之值，將算出來的值儲存於一名為 mod 之變數方便之後計算各項數值時引用，之後定義 fib[0]=0,fib[1]=1，即定義 Fibonacci series 前兩項為零以及一，之後執行一長度為 inf 之迴圈，使用動態規劃法之概念將各項數值求出，之後將各項輸出。

2. 原理介紹

根據模運算的理論，A 加 B 對某數取餘數會等於 A 對某數取餘數加 B 對某數取餘數，所以利用這個概念寫出主程式，將各項之值求出，且保證輸出之結果一定正確。

3. 原始程式碼

(1) //使用 C++語言

(2) #include<bits/stdc++.h>

(3) #define inf 10005

```
(4)  using namespace std;
(5)  int fast_pow(int n,int m){
(6)      int ans=1;
(7)      while(m>0) {
(8)          if(m % 2 == 1)
(9)              ans = (ans * n);
(10)         m = m/2;
(11)         n = (n * n) ;
(12)     }
(13)     return ans;
(14) }
(15) int fib[inf];
(16) int main(){
(17)     int number,power;
(18)     cin>>number>>power;
(19)     fib[0]=0;
(20)     fib[1]=1;
(21)     int mod=fast_pow(number,power);
(22)     for(int i=2;i<inf;i++){
(23)         fib[i]=(fib[i-1]+fib[i-2])%mod;
(24)     }
(25)     for(int i=0;i<inf;i++){
(26)         cout<<i<<" "<<fib[i];
(27)         if((fib[i]==0)&&(fib[i+1]==1)&&(fib[i+2]==1))
     cout<<" *";
(28)         cout<<"\n";
(29)     }
(30)     return 0;
(31) }
```

三、各質數循環個數之間之關係

(一)2 之循環個數

Fibonacci series 對質數 K 次方取餘數循環關係

圖一、二之循環狀況

圖一為使用正文第一段所介紹之程式所計算出二之循環狀況，由圖一我們可以輕易地看出在二的情況下循環個數是三個一循環，這也驗證了 Fibonacci series 之排列為偶數、奇數、奇數之規則。

(二)3 之循環個數

圖二、三之循環狀況

圖三為使用正文第一段所介紹之程式所計算出三之循環狀況，由圖三我們可以得到一個結論，那就是三的情況下的循環個數為八個一循環，並且我們也得到另一性質，即確定除了二以外 Fibonacci series 對其他質數取餘數會有循環關係。

(三) 其餘質數之循環個數

表一、各質數之循環結果

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

上表一為將前十八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 外其餘皆遵守 M=P-1。當尾數是 3 時，除 113+90X 外其餘皆遵守 M=2*P+2。當尾數是 7 時，除 47+60X 外其餘皆遵守 M=2*P+2。當尾數是 9 時，除 89+50X 外其餘皆遵守 M=P-1。

三、單一質數各次方間之關係

(一)2 的各次方

表二、二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為二，而次方為一至九之正整數，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二之等比數列，由此可推論質數各次方循環是有一定之關係的。

(二)3 的各次方

表三、三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三，而次方為一至九之正整數，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三之等比數列，符合上述推論。

(三)5 的各次方

表四、五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五，而次方為一至九之正整數，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五之等比數列，符合上述推論。

(四)公式

由窮舉法可做出以下推論，如令 M=循環個數,P=質數,K=次方，則循環個數之公式為$M \times P^{K-1}$，但此公式僅止於推論，尚未經嚴格證明導正，故仍可能有誤。

四、綜合討論

由以上結果可畫出下表五，下表五之橫軸為質數，縱軸為次方，由此表可見，各數之各次方間目前仍找不出一定之關聯，但單一質數各次方為言個遞增之等比數列。

表五、各質數及各次方

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

參●結論

一、結論說明

1. 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

2. 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

1. 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現在之狀態?

2. 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、Fundamental Of Data Structure In C。Sartaj Sahni,Ellis Horowitz 著。Computer Science Press。1990。

二、斐波那契數列。吳振奎著。九章出版社。 2000。

三、https://zh.wikipedia.org/wiki/%E6%96%90%E6%B3%A2%E9%82%A3%E5%A5%91%E6%95%B0%E5%88%97

四、https://zh.wikipedia.org/wiki/%E9%BB%84%E9%87%91%E5%88%86%E5%89%B2%E7%8E%87

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