國立臺南大學資訊工程學系

資工三「演算法」課程

第1次作業

**題目: Primality Testing**

|  |  |  |
| --- | --- | --- |
| 班級 | ： | 資工三 |
| 姓名 | ： | 陳博琳 |
| 學號 | ： | S10859007 |

老師：陳宗禧

中華民國 110年9月23日

# 目錄

1. **簡介及問題描述……….……………..…………………………………………1**
   1. **簡介…………….…………………………………………………………………………2**
   2. **問題………………………….……………………………………………………………4**
2. **理論分析….………………………..………………………………………8**
3. **演算法則….…………………………..……………………………………10**
4. **程式設計環境架構.………………………..…………………………………12**
5. **程式.…………………………………………..………………………………14**
6. **執行結果、討論與心得.………………………..……………………………18**

參考文獻………………………………………………………….…………………22

**(一) 簡介及問題描述**

設計與實作四種質數測試 (Primality test)演算法，理論驗證與實驗分析該問題!

1. 簡介

求Prime(*n*)? 首先定義 一整數*n*，判斷n是質數or合成數? …

質數定義:大於1，且只含有1和自己本身兩個因數的數。2為最小的因數

因此只要判斷是否有1與自己本身以外的因數則可判斷是否為質數。

2. 問題

Prime(*n*)問題定義：確認n數否為質數

求Prime(*n*)，實作與分析

* + 1. Basic Prime Testing
    2. Sieve of Eratosthenes
    3. Fermat Primality Testing
    4. Miller-Rabin Primality Testing

**(二) 理論分析**

首先第一個Basic Prime Test…

1.Basic Prime Testing理論

從2開始到小於n的最大正整數，若有任何數可被n整除則n不是質數。

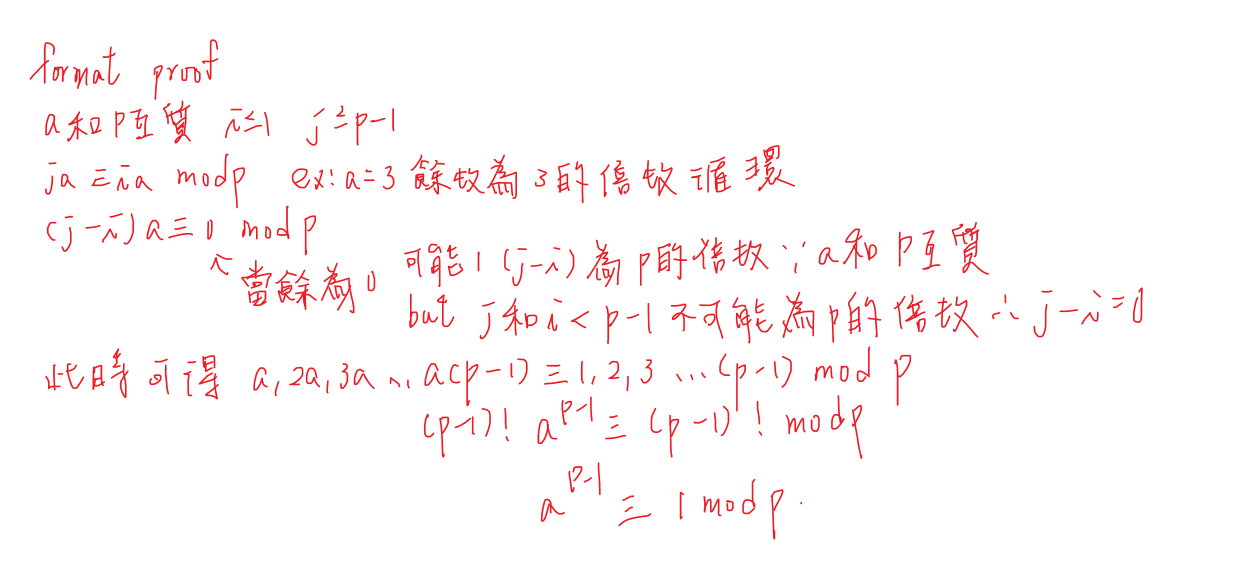
2. Sieve of Eratosthenes理論

若n可以整除從2開始到根號n的質數則不是質數，假如一個數N是合數，一個因數a,a x b=N，其中一個數必大於等於根號n，一個小於等於根號n 。因此只要小於等於根號n的質數不能整除N則N一定是質數。

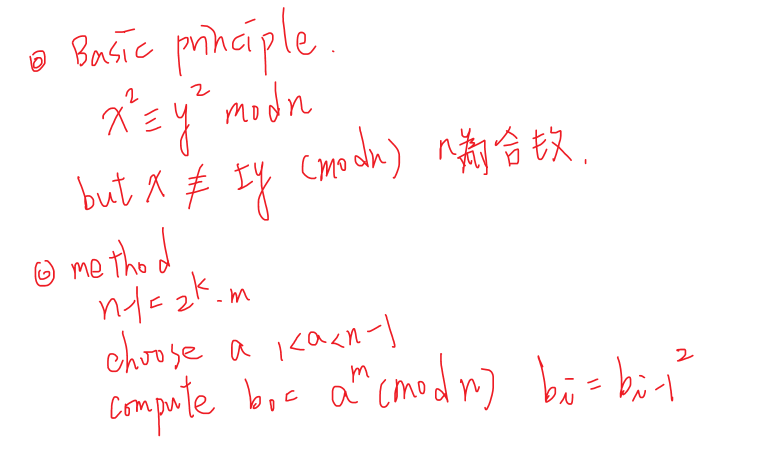
3. Format Primality Testing理論

A為正整數1<A<n-1，A^n-1%n=1

定理證明:



4. Miller-Rabin Primality Testing理論



**(三) 演算法則**

Prime Test? 演算法以及流程圖…

1. 第一個演算法(Algorithm)

Algotithm: basic prime testing

Input:N(正整數)

Output:bool(T or F)

Step 1:for(i=2,i<n-1,i++)

Step2:if N%i==0 return 0

else return 1

1. 演算法時間複雜度(time complexity)

O(n)

迴圈將i跑過2到N-1的每個數，因此時間複雜度為n

1. 演算法空間複雜度(space complexity)

O(1)

都是單一變數空間儲存，因此空間複雜度為1

2. 第二個演算法(Algorithm)

Algotithm: Sieve of Eratosthenes

Input:N(正整數)

Output:bool(T or F)

Step1:for(p=2,p\*p<N,p++)

Step2:if prime[p]is true

Step3:for(m=p\*p,m<=N,m+=p) prime[i] to false

Step4:if prime[N] is true return 1

Else return 0

1. 演算法時間複雜度(time complexity)

O()

雙層for迴圈外層部分p從2到最靠近N的p平方取n，內層迴圈m從p平方開始可以得到N(-1)但是會發散趨近無窮大數值25時，時間負雜度大約為n在此使用n但是其實不會收斂所以數字越大成長速度會快很多

1. 演算法空間複雜度(space complexity)

O(n),

使用陣列的方式將所有數字分別用true false識別辨認完之後為何者

3. 第三個演算法(Algorithm)

Algotithm: Format Primality Testing

Input:N(正整數)

Output:bool(T or F)

Step1:if n和2互質 2^n-1%n 直到餘數為+-1,else break

Step2: if n==1 return 1,else return 0Step3: 看N是否與3互質若是重複1-2步驟將2改為3

1. 演算法時間複雜度(time complexity)

O(logn),

此處的時間複雜度為取餘數時所產生的，為了能運算大數，使用的方法為循環餘數取法將餘數可相乘的原理套用此種作法時間複雜度為logn

1. 演算法空間複雜度(space complexity)

O(1),都是單一變數空間儲存，因此空間複雜度為1

4. 第四個演算法(Algorithm)

Algotithm: Miller-Rabin Primality Testing

Input:N(正整數)

Output:bool(T or F)

Step1:把N-1不斷/2直到除不進可質因數分解成2^k\*m

Step2:用2^m跟N取餘數

Step3:判斷餘數是否為N+1/-1

Step4:if 餘數等於num-1 return 1

Else return 0

1. 演算法時間複雜度(time complexity)

O(log n)

此處的時間複雜度為取餘數時所產生的，為了能運算大數，使用的方法為循環餘數取法將餘數可相乘的原理套用此種作法時間複雜度為logn

1. 演算法空間複雜度(space complexity)

O(1) 都是單一變數空間儲存，因此空間複雜度為1

(四) 程式設計環境架構

程式設計語言、工具、環境與電腦硬體等規格說明…

1. 程式語言

C in MS Windows

2. 程式開發工具

Visual C++ 2019

3. 電腦硬體

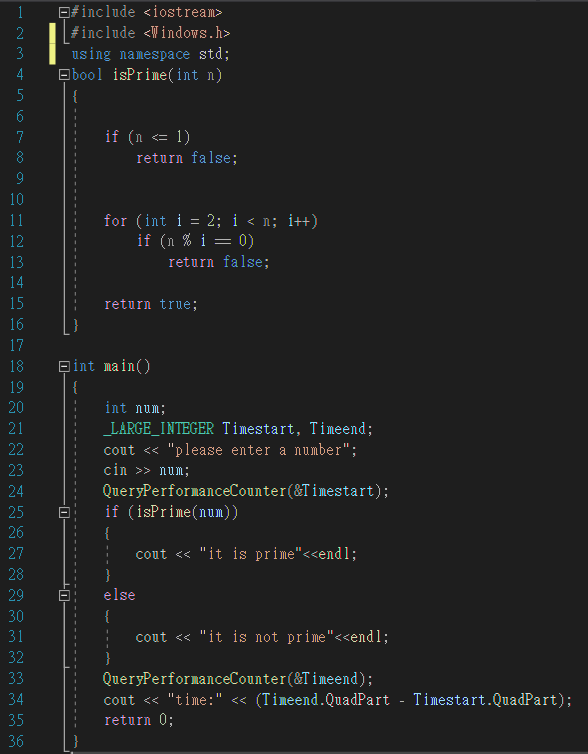
**(五) 程式 (含source code, input code, and output code)**

程式含source code, input code, and output code等…

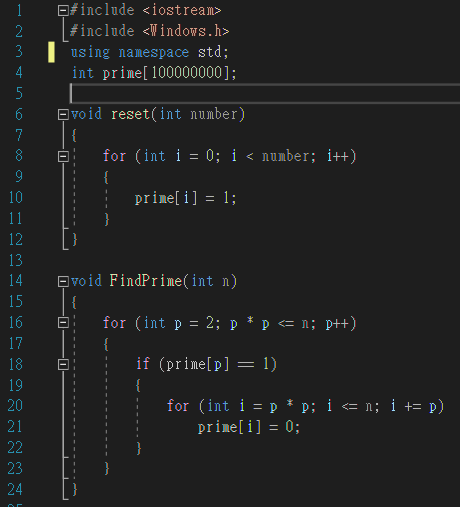
1. 主程式

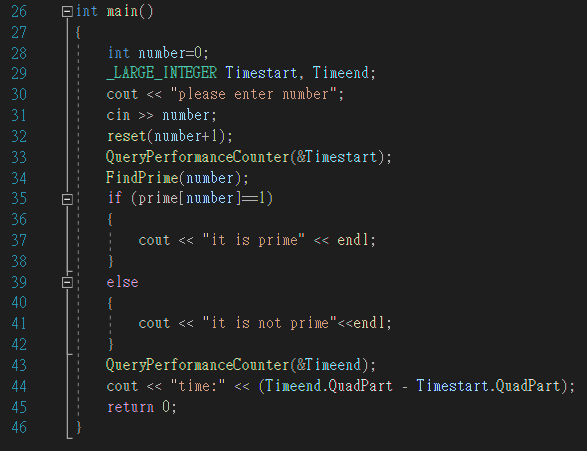
C++ program

1. Basic Prime Testing

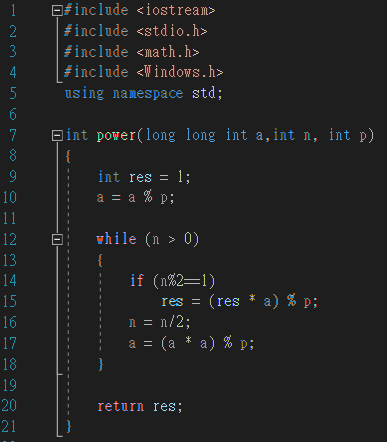


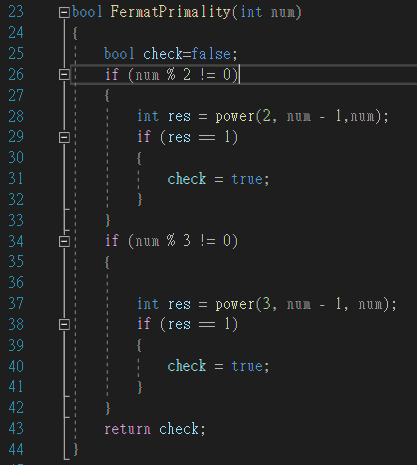
1. Sieve of Eratosthenes

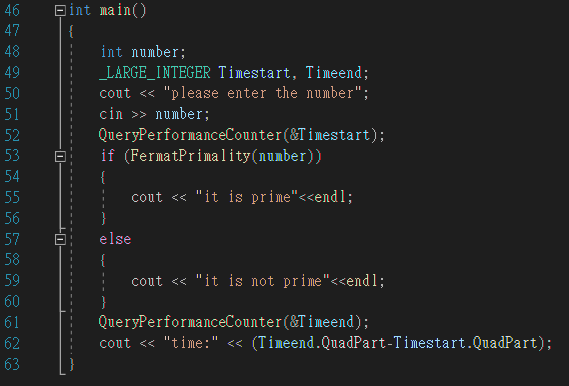




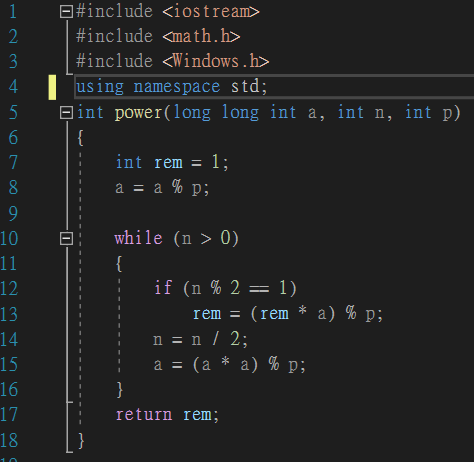
1. Fermat Primality Testing

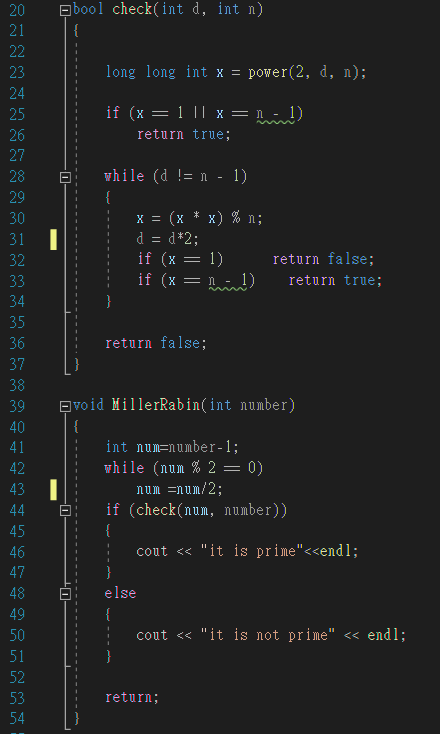


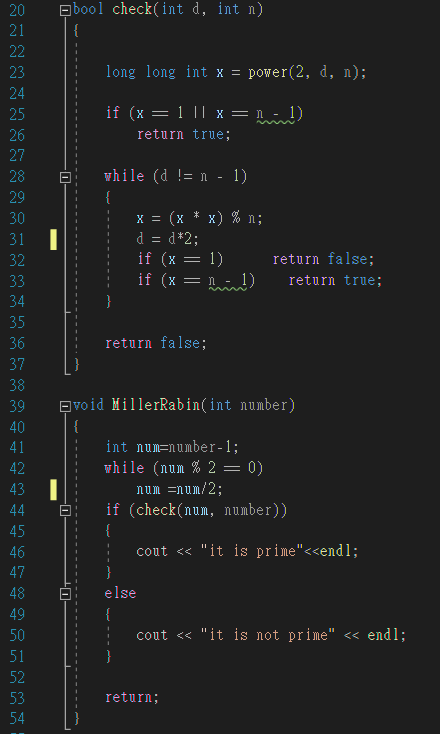


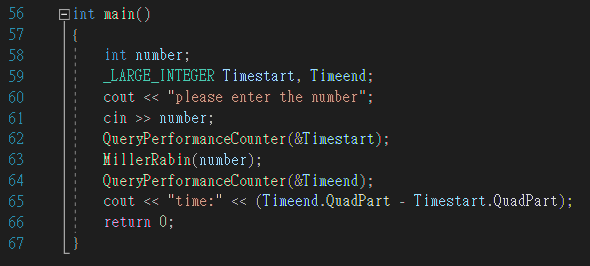


1. Miller-Rabin Primality Testing









2. Input Code Format

Three of examples for input use are in below….

(1)100

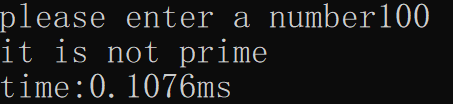
(2)101

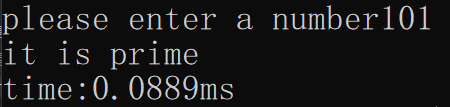
(3) 43112609

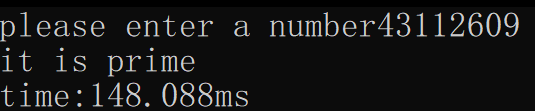
3. Output Code Format

Three of examples for output use are in below….

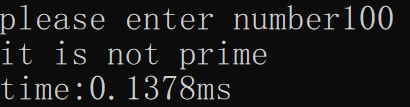
1. Basic Prime Testing

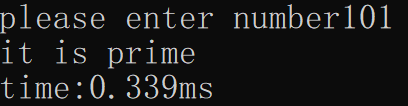
(1) 

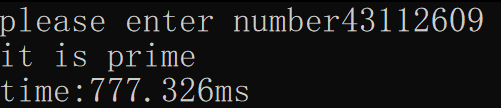
(2) 

(3) 

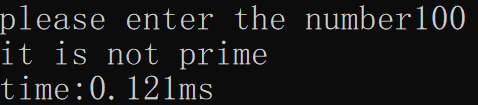
1. Sieve of Eratosthenes

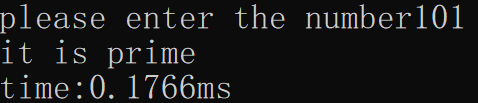
(1) 

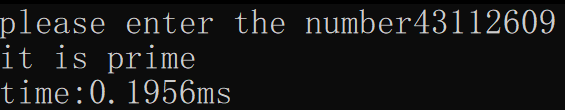
(2) 

(3) 

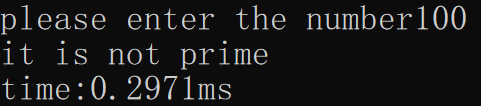
1. Format Primality Testing

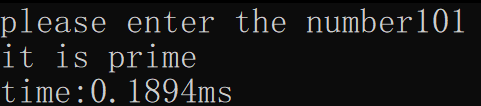
(1) 

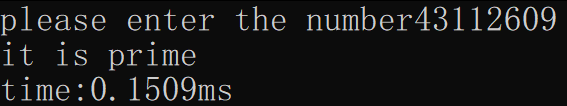
(2) 

(3) 

1. Miller-Rabin Primality Testing

(1) 

(2) 

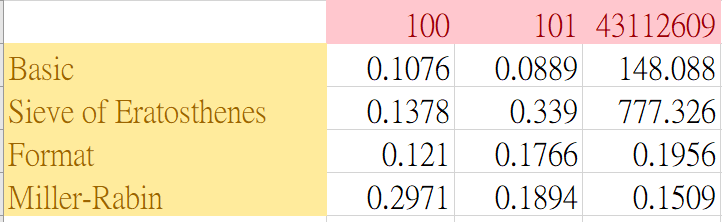
(3) 

**(五) 執行結果、討論與心得**

執行結果與討論 (執行時間、problem *n*的大小等問題討論)等…

1. 執行結果

Output of program:



由較小的三位數測資來看其實時間差異都不太大，但是當數字來到8為數時可以發現format跟miller的花費時間依然沒有大幅度的改變特別是miller的時間表現一直處於較低的位置，但是format跟miller因為演算法需要用到指數的緣故需要注意數字是否overflow

Format跟miller的k值主要希望能測試在不同底數的情況下運算解果，能避免偽質數情況而我是直接將2,3作為底數測試沒出現過錯誤情形，省去使用亂數座重複底數影響運算時間。

2. 討論

執行時間、問題大小等問題討論! 利用 MS Excel 畫出問題大小與執行時間的關係!

1. Running Time

i.

ii.

此處會放兩張圖主要是basic跟sieve在大數運算花費太多時間小比例尺無法呈現，大比例尺會導致小樹時的軸線消失，因此i將max設為0.35可以觀察小數時的折線變化並且知道basic和sieve將花費大量時間，ii可以觀察在大數情況下sieve與basic的增長速度

(2) Problem size *n*

i.三位數時所有演算法的差異都不大，Miller最慢

ii.大數時basic與sieve花費時間爬升快，sieve最多

可以發現sieve是隨著N增長時間動盪最大的

3. 心得

實作與設計Primality Testing問題，使我更加熟悉時間複雜度與真正執行時間的相同處，以及n數越大不同演算法的時間變化，其實確定演算法程式在撰寫的時候還是有些許的差異，例如2^M取餘數原本我想的方法就是次方完再取發現N的大小受到相當大的限制，因為computer最大變數微long long int2^32，很容易就超過了，後來查找資料才發現餘數有循環特性，即便相同的數學式，了解這個特性對程式還是有相當大的幫助。

**參考文獻**

1. <https://cp-algorithms.com/algebra/primality_tests.html>
2. <https://www.geeksforgeeks.org/sieve-of-eratosthenes/>
3. <https://www.geeksforgeeks.org/primality-test-set-2-fermet-method/>
4. <https://www.youtube.com/watch?v=SyK3IXPITco>
5. https://www.youtube.com/watch?v=qfgYfyyBRcY