國立臺南大學資訊工程學系

資工三「演算法」課程

第一次作業

**題目: Counting the *n*-th Fibonacci Number**

|  |  |  |
| --- | --- | --- |
| 班級 | ： | 資工三 |
| 姓名 | ： | 周巧晴 |
| 學號 | ： | S10959042 |

老師：陳宗禧

中華民國 111年10月5日

# 目錄

1. **簡介及問題描述摘要….……………..…………………………………………3**
   1. **簡介…………….…………………………………………………………………………3**
   2. **問題………………………….……………………………………………………………3**
2. **理論分析….………………………..………………………………………4**
3. **演算法則….…………………………..……………………………………7**
4. **程式設計環境架構.………………………..…………………………………11**
5. **程式.…………………………………………..………………………………12**
6. **執行結果、討論與心得.………………………..……………………………21**

參考文獻…………………………………………………………………………31

**(一) 簡介及問題描述摘要**

設計與實作五個演算法，計算Fibonacci Number?….

1. 簡介

求Fibonacci Number(*Fn*)! 首先定義 F(n)為F(n-2)+F(n-1)，n>=2，且F(n)=n，0<=n<=1，已知有許多不同演算法能計算出此結果，請實作與分析下列演算法in Subsection 2.5：

2. 問題

實作與分析下列演算法in Subsection 2.5：

1. 求Fibonacci Number(*Fn*)，實作與分析下列演算法in Subsection 2.5：
   * 1. Recursive algorithm
     2. Non-recursive definition-based algorithm
     3. Explicit formula algorithm
     4. Sequential algorithm based on the following formula:



* + 1. Logarithmic algorithm based on the following formula:



(b) 回答第2.5節的習題第9題(答案寫在完整報告中)。此外，寫出五個演算法的時間複雜度。

**(二) 理論分析**

首先第一個遞迴演算法如下…

1. Recursive Algorithm

Recursive 是利用function 定義內call function自己本身的方法，需要明訂初始值與非初始值之所有n的遞迴定義，並return答案。

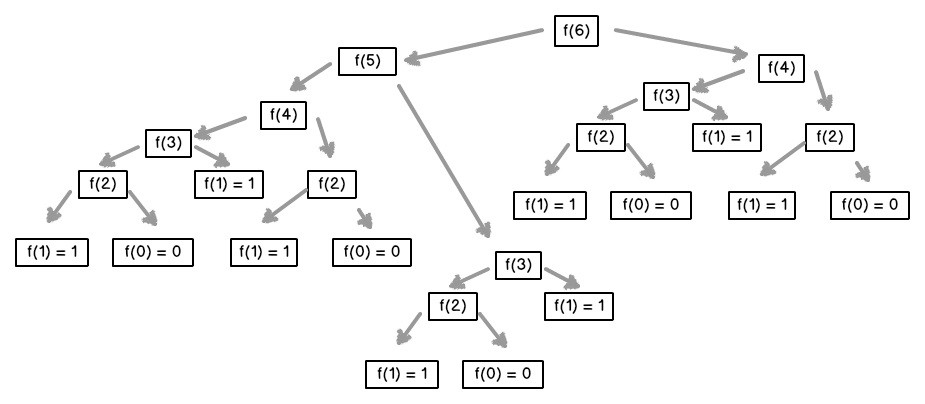
初始值：F(n)=n，0<=n<=1，n為整數

遞迴定義：F(n)為F(n-2)+F(n-1)，n>=2，n為整數

n>=2的Return ：F(n-2)+F(n-1)

Time Complexity?

由於每次獲得F(n)值都須先取得F(n-1)和F(n-2)的值，而獲得F(n-1)值也須先取得F(n-2)和F(n-3)的值，以此類推，所以時間複雜度是2n也就是O(2n)。



2. Non-Recursive Algorithm

Non-Recursive 利用For迴圈依次取得F(0)、F(1)、F(2)、…、F(n)。

如下先訂好初始值，並用迴圈和前兩次的值取得後續的值。

F(0)=0

F(1)=1

F(2)=F(0)+F(1)

F(3)=F(1)+F(2)

…

F(n)=F(n-2)+F(n-1)

Time Complexity?

由於每次運算都能得到下個F的值，所以只需時間複雜度n，也就是O(n)。

F(6)

F(5)

F(4)

F(3)

F(2)

F(0)

F(1)

3. Explicit formula

計算顯式(用特徵方程式)

F(n)-F(n-1)-F(n-2)=0

r2-r-1=0

r1,2=

F(n)=

Time Complexity?

4. Sequential algorithm

Exercise 9 in Subsection 2.5



乘以 n-1次

Time Complexity?

2X2矩陣每次相乘都須用到8次乘法，所以需相乘8(n-1)次，也就是。

5. Logarithmic algorithm

Exercise 9 in Subsection 2.5

  
 會拆分成

其中,

Time Complexity?

m會是，2X2矩陣每次相乘需用到8次乘法。

所以會乘constant)次，也就是。

**(三) 演算法則**

求解 *Fn*…

1. 第一個演算法(Algorithm)---**Recursive Algorithm 遞迴演算法**

input：A nonnegative integer n

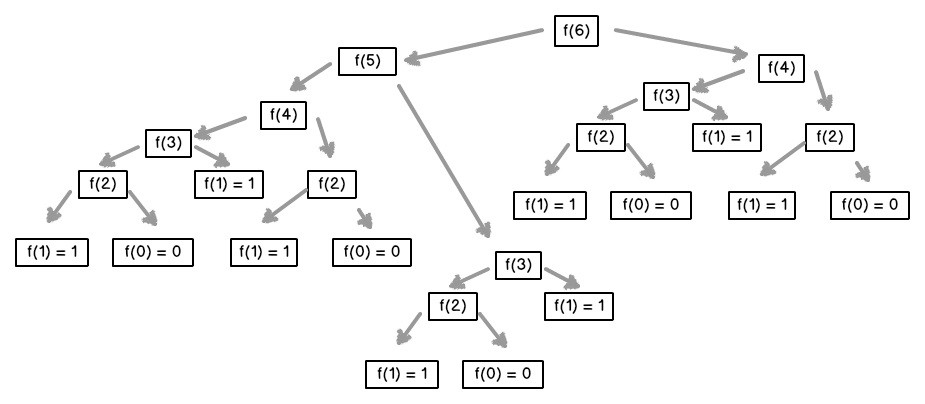
Output：The nth Fibonacci number

if n<=1 return n

else return F(n-1)+F(n-2)

1. 演算法時間複雜度(time complexity)

O(), 由於每次獲得F(n)值都須先取得F(n-1)和F(n-2)的值，而獲得F(n-1)值也須先取得F(n-2)和F(n-3)的值，以此類推，所以時間複雜度是2n也就是O(2n)。



1. 演算法空間複雜度(space complexity)

O(1), 沒有用到任何n以外的空間

2. 第二個演算法(Algorithm)---**Non-Recursive Algorithm 非遞迴演算法**

Input：A nonnegative integer n

Output：The nth Fibonacci number

F[0] <- 0; F(1) <- 1

for i <- 2 to n do

F[i] <- F[i-1] + F[i-2]

Return F[n]

1. 演算法時間複雜度(time complexity)

O(*n*), 由於每次運算都能得到下個F的值，所以只需時間複雜度n，也就是O(n)。

F(6)

F(5)

F(4)

F(3)

F(2)

F(0)

F(1)

1. 演算法空間複雜度(space complexity)

O(n), 額外用到F[n]的空間

3. 第三個演算法(Algorithm)---**Explicit formula 顯式演算法**

Input：A nonnegative integer n

Output：The nth Fibonacci number

return pow((1 + sqrt(5)) / 2, n) - pow((1 - sqrt(5)) / 2, n)) / sqrt(5)

1. 演算法時間複雜度(time complexity)

,

1. 演算法空間複雜度(space complexity)

O(1), 沒有用到任何n以外空間

4. 第四個演算法(Algorithm)--- **Sequential algorithm**

Input：A nonnegative integer n

Output：The nth Fibonacci number

mtx <- {{0,1},{1,1}}

arr <- {{0,1},{1,1}}

for i <- 2 to n do

arr <- arr\*mtx

return arr[0][1]

1. 演算法時間複雜度(time complexity)

O(*n*)，2X2矩陣每次相乘都須用到8次乘法，所以需相乘8(n-1)次。

1. 演算法空間複雜度(space complexity)

O(1), 只額外用到兩個2x2矩陣(被乘數、乘數、相乘結果)。

5. 第五個演算法(Algorithm)---**Logarithmic algorithm**

Input：A nonnegative integer n

Output：The nth Fibonacci number

arr <- {{0,1},{1,1}}

arr2 <- {{0,1},{1,1}}

for (i = n ; !(i % 2) && i != 0; i /= 2)

arr2 = arr2\*arr2

arr = arr2

i /= 2

for (j=I; j; j/=2)

arr2=arr2\*arr2

if(j%2)

arr=arr\*arr2

return arr[0][1]

1. 演算法時間複雜度(time complexity)

O(), 會乘constant)次。

1. 演算法空間複雜度(space complexity)

O(1), 只額外用到兩個2x2矩陣(被乘數、乘數)。

**(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 Windows 10

2. 程式開發工具

Visual Studio 2022

3. 電腦硬體

處理器 Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz

已安裝記憶體(RAM) 8.00 GB (7.81 GB 可用)

系統類型 64 位元作業系統，x64 型處理器

手寫筆與觸控 手寫筆支援

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

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

**(一)Recursive Algorithm 遞迴演算法**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

double Fibonacci(unsigned int);

int main(void) {

int n;

double START, END;

printf("Please input n for Fibonacii number F(n) for recursive algorithm>>");

scanf("%n", &n);

START = clock();

printf("Answer:%.0f\n", Fibonacci(n));

END = clock();

printf("Time:%.0fms", (END - START));

system("pause");

return 0;

}

double Fibonacci(unsigned int n) {

if (n <= 1) {

return n;

}

else {

return Fibonacci(n - 1) + Fibonacci(n - 2);

}

}

2. Input Code Format

Three of examples for input use are in below….

(1)40

(2)45

(3)50

3. Output Code Format

Three of examples for output use are in below….

(1)

Please input n for Fibonacii number F(n) for recursive algorithm>>40

Answer:102334155

Time:519ms

(2)

Please input n for Fibonacii number F(n) for recursive algorithm>>45

Answer:1134903170

Time:5703ms

(3)

Please input n for Fibonacii number F(n) for recursive algorithm>>50

Answer:12586269025

Time:65610ms

**(二)Non-Recursive Algorithm 非遞迴演算法**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

int main(void) {

int n;

double\* arr;

double START, END;

printf("Please input n for Fibonacii number F(n) for non-recursive definition-based algorithm>>");

scanf("%d", &n);

START = clock();

arr= new double[n+1];

arr[0] = 0;

arr[1] = 1;

for (int i = 2; i <= n; i++) {

arr[i] = arr[i-1] + arr[i-2];

}

END = clock();

printf("Answer:%.0f\n", arr[n]);

printf("Time:%.0fms", (END - START));

system("pause");

delete[] arr;

return 0;

}

2. Input Code Format

Three of examples for input use are in below….

(1)100

(2)200

(3)1000

3. Output Code Format

Three of examples for output use are in below….

(1)

Please input n for Fibonacii number F(n) for non-recursive definition-based algorithm>>100

Answer:354224848179261997056

Time:0ms

(2)

Please input n for Fibonacii number F(n) for non-recursive definition-based algorithm>>200

Answer:280571172992510158789853658433108939112448

Time:0ms

(3)

Please input n for Fibonacii number F(n) for non-recursive definition-based algorithm>>1000

Answer:43466557686937427506996155551113736989014930606753135940421893445574924103291704556590724031629457130224242562009803138605795985275055883724744645670017852944230821885448147900618234216285807506383791353495552

Time:0ms

**(三)Explicit formula 顯式演算法**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "math.h"

#include "time.h"

int main(void) {

double START, END;

int n;

printf("Please input n for Fibonacii number F(n) for explicit formula algorithm>>");

scanf("%d", &n);

START = clock();

printf("Answer:%.0f\n", (pow((1 + sqrt(5)) / 2, n) - pow((1 - sqrt(5)) / 2, n)) / sqrt(5));

END = clock();

printf("Time:%.0fms", (END - START));

system("pause");

return 0;

}

2. Input Code Format

Three of examples for input use are in below….

(1)30

(2)1001

(3)1200

3. Output Code Format

Three of examples for output use are in below….

(1)

Please input n for Fibonacii number F(n) for explicit formula algorithm>>30

Answer:832040

Time:1ms

(2)

Please input n for Fibonacii number F(n) for explicit formula algorithm>>1001

Answer:70330367711425174290694829277217085237890517319611544836492248291277937018392669473182883449499700079838702138514693954865215540324463972269041336236408103803463044485157310035600578886573282113950100689518592

Time:0ms

(3)

Please input n for Fibonacii number F(n) for explicit formula algorithm>>1200

Answer:27269884455407365847192026241192332547250796698164704715463580385597197075537548253492087865024098245321784944722311472705029391914337116136306691783618953277982977916818418369582710642008814191391090984759478348473276714347732043884108324642586886144

Time:1ms

**(四)Sequential algorithm**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

int main(void) {

const int mtx[2][2] = { {0,1},{1,1} };

double arr[2][2] = { {0,1},{1,1} }, temp[2][2] = { {0,1},{1,1} };

int n;

double START, END;

printf("Please input n for Fibonacii number F(n) for sequential algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>");

scanf("%d", &n);

START = clock();

if (n == 0) {

END = clock();

printf("Answer:%.0f\n", arr[0][0]);

printf("Time:%.0fms", (END - START));

}

else {

for (int i = 1; i < n; i++) {

temp[0][0] = arr[0][0] \* mtx[0][0] + arr[0][1] \* mtx[1][0];

temp[0][1] = arr[0][0] \* mtx[0][1] + arr[0][1] \* mtx[1][1];

temp[1][0] = arr[1][0] \* mtx[0][0] + arr[1][1] \* mtx[1][0];

temp[1][1] = arr[1][0] \* mtx[0][1] + arr[1][1] \* mtx[1][1];

arr[0][0] = temp[0][0]; arr[0][1] = temp[0][1]; arr[1][0] = temp[1][0]; arr[1][1] = temp[1][1];

}

END = clock();

printf("Answer:%.0f\n", arr[0][1]);

printf("Time:%.0fms", (END - START));

}

system("pause");

return 0;

}

2. Input Code Format

Three of examples for input use are in below….

(1)5

(2)60

(3)500

3. Output Code Format

Three of examples for output use are in below….

(1)

Please input n for Fibonacii number F(n) for sequential algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>5

Answer:5

Time:0ms

(2)

Please input n for Fibonacii number F(n) for sequential algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>60

Answer:1548008755920

Time:0ms

(3)

Please input n for Fibonacii number F(n) for sequential algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>500

Answer:139423224561697698330489613862193018947914545343780323868775030943672596190605867771863846635039486902272

Time:0ms

**(五) Logarithmic algorithm**

1. 主程式

C program

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "stdlib.h"

#include "time.h"

int main(void) {

double arr[2][2] = { {1,0},{0,1} }, arr2[2][2] = { {0,1},{1,1} }, temp[2][2] = { {0,1},{1,1} };

int n, i;

double START, END;

printf("Please input n for Fibonacii number F(n) for logarithmic algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>");

scanf("%d", &n);

START = clock();

if (n == 0) {

END = clock();

printf("Answer:0\nTime:%.0fms", END - START);

}

else {

for (i = n; !(i % 2) && i != 0; i /= 2,

temp[0][0] = arr2[0][0] \* arr2[0][0] + arr2[0][1] \* arr2[1][0],

temp[0][1] = arr2[0][0] \* arr2[0][1] + arr2[0][1] \* arr2[1][1],

temp[1][0] = arr2[1][0] \* arr2[0][0] + arr2[1][1] \* arr2[1][0],

temp[1][1] = arr2[1][0] \* arr2[0][1] + arr2[1][1] \* arr2[1][1],

arr2[0][0] = temp[0][0], arr2[0][1] = temp[0][1], arr2[1][0] = temp[1][0], arr2[1][1] = temp[1][1]) {

}

arr[0][0] = arr2[0][0]; arr[0][1] = arr2[0][1]; arr[1][0] = arr2[1][0]; arr[1][1] = arr2[1][1];

i /= 2;

for (int j = i; j; j /= 2) {

temp[0][0] = arr2[0][0] \* arr2[0][0] + arr2[0][1] \* arr2[1][0];

temp[0][1] = arr2[0][0] \* arr2[0][1] + arr2[0][1] \* arr2[1][1];

temp[1][0] = arr2[1][0] \* arr2[0][0] + arr2[1][1] \* arr2[1][0];

temp[1][1] = arr2[1][0] \* arr2[0][1] + arr2[1][1] \* arr2[1][1];

arr2[0][0] = temp[0][0]; arr2[0][1] = temp[0][1]; arr2[1][0] = temp[1][0]; arr2[1][1] = temp[1][1];

if (j % 2) {

temp[0][0] = arr[0][0] \* arr2[0][0] + arr[0][1] \* arr2[1][0];

temp[0][1] = arr[0][0] \* arr2[0][1] + arr[0][1] \* arr2[1][1];

temp[1][0] = arr[1][0] \* arr2[0][0] + arr[1][1] \* arr2[1][0];

temp[1][1] = arr[1][0] \* arr2[0][1] + arr[1][1] \* arr2[1][1];

arr[0][0] = temp[0][0]; arr[0][1] = temp[0][1]; arr[1][0] = temp[1][0]; arr[1][1] = temp[1][1];

}

}

END = clock();

printf("Answer:%.0f\n", arr[0][1]);

printf("Time:%.0fms", (END - START));

}

system("pause");

return 0;

}

2. Input Code Format

Three of examples for input use are in below….

(1)300

(2)600

(3)900

3. Output Code Format

Three of examples for output use are in below….

(1)

Please input n for Fibonacii number F(n) for logarithmic algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>300

Answer:222232244629420392510089794533240483298694966682232223447384064

Time:0ms

(2)

Please input n for Fibonacii number F(n) for logarithmic algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>600

Answer:110433070572952188993157030676824190043274346968340085820550479891225298564502208305490264786994951789852253601676432697720832

Time:0ms

(3)

Please input n for Fibonacii number F(n) for logarithmic algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>900

Answer:54877108839479967174347228393035880541630461690686101300835264930796266557250522567675091845368393446441342328235615385087999131801982432981424657026181706641029828773559969069223265697792

Time:0ms

**(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…

下列圖表縱軸為執行時間(ms)，橫軸為n的值

**(一)Recursive Algorithm 遞迴演算法**

1. 執行結果

Output of program:

i.

Please input n for Fibonacii number F(n) for recursive algorithm>>10

Answer:55

Time:1ms

ii.

Please input n for Fibonacii number F(n) for recursive algorithm>>30

Answer:832040

Time:8ms

iii.

Please input n for Fibonacii number F(n) for recursive algorithm>>50

Answer:12586269025

Time:63136ms

2. 討論

執行時間、問題大小等問題討論

1. Running Time
   1. 1ms
   2. 8ms
   3. 63136ms

因為當n>50時執行時間會過長(n=55約10分鐘)，所以只採到n<=50的樣本。

圖表呈指數圖形，符合O(結果。

執行時間效率為5種演算法最差。

(2) Problem size *n*

1. i. input size=1，n=10
2. input size=1，n=30
3. input size=1，n=50

input size不變，n的值根據使用者輸入而定。

**(二)Non-Recursive Algorithm 非遞迴演算法**

1. 執行結果

Output of program:

i.

Please input n for Fibonacii number F(n) for non-recursive definition-based algorithm>>100

Answer:354224848179261997056

Time:0ms

ii.

Please input n for Fibonacii number F(n) for non-recursive definition-based algorithm>>1000000

Answer:inf

Time:4ms

iii.

Please input n for Fibonacii number F(n) for non-recursive definition-based algorithm>>100000000

Answer:inf

Time:748ms

經過查表和比對實驗結果發現當n=79時，查表答案為14,472,334,024,676,221 ，而程式執行結果為14,472,334,024,676,220，所以當n>=79時，答案會產生誤差。

2. 討論

執行時間、問題大小等問題討論

1. Running Time
2. 0ms
3. 4ms
4. 748ms

圖表呈線性，符合O(n)結果。

執行時間效率為5種演算法中第四名。

(2) Problem size *n*

1. input size=1，n=100
2. input size=1，n=1000000
3. input size=1，n=100000000

input size不變，n的值根據使用者輸入而定。

**(三)Explicit formula 顯式演算法**

1. 執行結果

Output of program:

i.

Please input n for Fibonacii number F(n) for explicit formula algorithm>>100

Answer:354224848179263111168

Time:0ms

ii.

Please input n for Fibonacii number F(n) for explicit formula algorithm>>1000000

Answer:inf

Time:0ms

iii.

Please input n for Fibonacii number F(n) for explicit formula algorithm>>100000000

Answer:inf

Time:0ms

經過查表和比對實驗結果發現當n=71時，查表答案為308,061,521,170,129 ，而程式執行結果為308,061,521,170,130，所以當n>=71時，答案會產生誤差。

2. 討論

執行時間、問題大小等問題討論

1. Running Time
2. 0ms
3. 0ms
4. 0ms

因時間單位為ms，此演算法的執行時間均不到1ms，成長速度過慢，所以無法具體畫出圖形。

但理論的明顯不符合實際結果。

執行時間效率為5種演算法中第一。

(2) Problem size *n*

1. input size=1，n=100
2. input size=1，n=1000000
3. input size=1，n=100000000

input size不變，n的值根據使用者輸入而定。

**(四)Sequential algorithm**

1. 執行結果

Output of program:

i.

Please input n for Fibonacii number F(n) for sequential algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>100

Answer:354224848179261997056

Time:0ms

ii.

Please input n for Fibonacii number F(n) for sequential algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>1000000

Answer:-nan(ind)

Time:3ms

iii.

Please input n for Fibonacii number F(n) for sequential algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>100000000

Answer:-nan(ind)

Time:252ms

經過查表和比對實驗結果發現當n=79時，查表答案為14,472,334,024,676,221 ，而程式執行結果為14,472,334,024,676,220，所以當n>=79時，答案會產生誤差。

2. 討論

執行時間、問題大小等問題討論

1. Running Time
2. 0ms
3. 3ms
4. 252ms

圖表呈非常漂亮的線性，符合O(n)的結果。

執行時間效率為5種演算法中第三。

(2) Problem size *n*

1. input size=1，n=100
2. input size=1，n=1000000
3. input size=1，n=100000000

input size不變，n的值根據使用者輸入而定。

**(五) Logarithmic algorithm**

1. 執行結果

Output of program:

i.

Please input n for Fibonacii number F(n) for logarithmic algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>100

Answer:354224848179261931520

Time:0ms

ii.

Please input n for Fibonacii number F(n) for logarithmic algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>1000000

Answer:inf

Time:0ms

iii.

Please input n for Fibonacii number F(n) for logarithmic algorithm based on [(F(n-1),F(n)),(F(n(,F(n+1)]=[(0,1),(1,1)]^n,n>=1>>100000000

Answer:inf

Time:0ms

經過查表和比對實驗結果發現當n=79時，查表答案為14,472,334,024,676,221 ，而程式執行結果為14,472,334,024,676,220，所以當n>=79時，答案會產生誤差。

2. 討論

執行時間、問題大小等問題討論

1. Running Time
2. 0ms
3. 0ms
4. 0ms

因時間單位為ms，此演算法的執行時間均不到1ms，成長速度過慢，所以無法具體畫出圖形。

不知是否符合理論的O()。

執行時間效率為5種演算法中第一，與第三種並列。

(2) Problem size *n*

1. input size=1，n=100
2. input size=1，n=1000000
3. input size=1，n=100000000

input size不變，n的值根據使用者輸入而定。

Exercise 9 in Subsection 2.5

已知，n>=2

=，n>=1  
根據數學歸納法

得證。

3. 心得

設計求*Fn*程式，使我更加熟悉時間複雜度與真正執行時間的相同處，以及各種演算法的實踐與理論的不同處，親身體驗了演算法時間複雜度的影響，O()和O(n)的時間所需差距甚大，O()在input n =50時就需要跑一分多鐘了，而O(n)即使input n達到100000000仍只需不到1ms的時間，以及時間複雜度相同時，時間仍有不小差距(第二題、第四題)。n太大時都會over flow導致算不出正確解答。在寫作業時也重溫了一遍C的基本語法，重拾半年沒用到的C語言能力，順便還認識了clock function，第一次使用函數取得程式執行時間。

順道一提，我認為n不該是小數，因為如果是小數就完全違反了定義中的「非負整數」，既然如此，就會改變規則，那麼就不是Fibonacci number了！

**參考文獻**

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, "Introduction to Algorithms," Third Edition, The MIT Press, 2009.
2. R.C.T. Lee, S.S. Tseng, R.C. Chang, and Y.T.Tsai, "Introduction to the Design and Analysis of Algorithms," McGraw-Hill, 2005.
3. Anany V. Levitin, "Introduction to the Design and Analysis of Algorithms," 2nd Edition, Addison Wesley, 2007.
4. Richard Neapolitan and Kumarss Naimipour, "Foundations of Algorithms," Fourth Edition, Jones and Bartlett Publishers, 2010.
5. …