資料結構報告範例

姓名 惲子威

July 26, 2024

CONTENTS

CHAPTER 2

解題說明

遞迴:依照題目規則去實作。
非遞迴: 根據函數表，歸納出 m 和 n 的關係式

Ackermann's function $A(m,n)$ is defined as follows:

$$A(m,n) = \begin{cases} n+1 & \text{, if } m = 0 \\ A(m-1,\ 1) & \text{, if } n = 0 \\ A(m-1,\ A(m,n-1)) & \text{, otherwise} \end{cases}$$

This function is studied because it grows very fast for small values of m and n. Write a recursive function for computing this function. Then write a nonrecursive algorithm for computing Ackermann's function.

Figure 1.1: sum.cpp

CHAPTER 3

演算法設計與實作

遞迴

```
int A_r(int m,int n)//遞迴
{
    if(m==0){return n+1;}
    else if(n==0){return A_r(m-1,1);}
    else{return A_r(m-1,A_r(m,n-1));}

}
```

非遞迴

```
int A_nr(int m,int n)//非遞迴
{
    stack<int> s;          //name s stack
    s.push(m);             //push m
    while (!s.empty()) //不是empty繼續
    {
        m=s.top();
        s.pop();

        if(m==0){n++;          //n+1
        }
        else if(n==0)          //A(m-1,1)
        {
            s.push(m-1);
            n=1;
        }
        else                   //A(m-1,A(m,n-1))
        {
            s.push(m-1);
            s.push(m);
            n--;
        }

    }
    return n;
}
```

CHAPTER 4

```
//41043257 徐子晟 Ackermann Function
#include <cstdio>
#include <iostream>
#include <stack>
using namespace std;
int A_r(int m,int n)//遞迴
{
    if(m==0){return n+1;}
    else if(n==0){return A_r(m-1,1);}
    else{return A_r(m-1,A_r(m,n-1));}

}

int A_nr(int m,int n)//非遞迴
{
    stack<int> s;       //name s stack
    s.push(m);          //push m
    while (!s.empty()) //不是empty繼續
    {
        m=s.top();
        s.pop();

        if(m==0){n++;       //n+1
        }
        else if(n==0)       //A(m-1,1)
        {
            s.push(m-1);
            n=1;
        }
        else                //A(m-1,A(m,n-1))
        {
            s.push(m-1);
            s.push(m);
            n--;
        }

    }
    return n;
}


int main()
{
    int a,b,m,n,status;     //a=Recursive answer, b= Nonrecursive answer, m=m, n=n
    while(cin>>m>>n)//輸入兩個數
    {
    a = A_r(m,n);
    b = A_nr(m,n);
    cout<<"recursive "<<a<<endl; //output answer
    cout<<"non recursive "<<b<<endl;
    }
    return 0;
 }
```

Figure 2.1: main.cpp

CHAPTER 5

效能分析

遞迴

時間複雜度

A(0,n)=n+1
A(m,0)=A(m−1,1)當
m>0m > 0m>0
A(m,n)=A(m−1,A(m,n
−1)) 當 m>0m > 0m>0
且 n>0n > 0n>0

空間複雜度

O(A(m,n))

非遞迴

時間複雜度

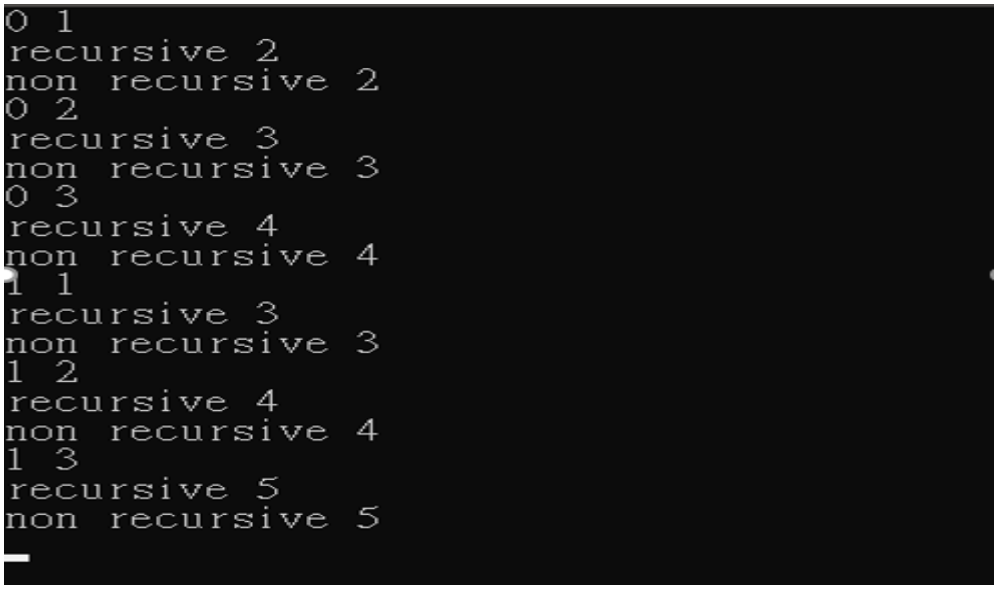
A(1,n)=n+2（線性增長）
A(2,n)=2n+3 線性增長）
A(3,n) 增長極快（指數增長）
A(4,n)和更高的值在計算上是不可行的，即使 nnn 值相對較小（超指數增長）

空間複雜度

O(A(m,n))

CHAPTER 6

測試與過程



討論:
如果使用遞迴程式，輸入n越大，程式跑的時間越久。
但是用非遞迴解的話，程式的速度幾乎不會受到n的數值影響。