Ackermann function

羅冠穎

2024/10/21

解題說明_遞迴

如果m=0,就n=n+=1
如果m!=0且n=0就把當前m值減1,把n值設為1在執行一次
如果非以上兩種情況則將m值-1,把A(m,n-1)的結果設為n值再執行一次

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

實作參見檔案HW01_01,以下為遞迴函式:

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

解題說明_非遞迴

非遞迴的方式則是用陣列來當作遞迴的深度
Top則作為當前深度的指標

實作參見檔案HW01_01,以下為非遞迴函式:

```
int Acmf_nonrecursive(int m,int n){
    int stack[1000];
    int top=0;
    stack[top]=m;

    while(top>=0){
        m=stack[top--];

        if(m==0){
            n=n+1;
        }
            else if(n==0){
                n=1;
                stack[++top]=m-1;
            }
        else{
            stack[++top]=m-1;
            stack[++top]=m;
            n=n-1;
        }
    }
    return n;
}
```

設計&實作01

```
int main(){
    unsigned int m,n;
    cout<<"輸入m值:";
    cin>>m;
    cout<<"輸入n值:";
    cin>>n;
    if(m<4&&n<4){//判斷m或n是否大於3
        int ctrl;
        cout<<"使用遞迴還是非遞迴?(1:遞迴/2:非遞迴):";
        cin>>ctrl;
        switch(ctrl){
            case 1:
                cout<<"Acmf("<<m<<","<<n<<")="<<Acmf_recursive(m,n)<<"\n";//執行陣列版本
                break;
            case 2:
                cout<<"Acmf("<<m<<","<<n<<")="<<Acmf_nonrecursive(m,n)<<"\n";//執行非陣列版本
                break;
            default:
                cout<<"沒這個選項";
        }
    }
    else{
        cout<<"m或n的數值超過3了,無法執行";
    }
}
```

設計&實作02_遞迴

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

設計&實作03_非遞迴

```
//非遞迴函式
int Acmf_nonrecursive(int m,int n){
    int stack[1000];//用陣列來看成是遞迴的深度
    int top=0;
    stack[top]=m;

    while(top>=0){//非初始層判斷
        m=stack[top--];//取出之前的值,並且調整top

        if(m==0){
            n=n+1;
        }
            else if(n==0){
                n=1;
                stack[++top]=m-1;//將m-1的值存入下一層
            }
        else{
            stack[++top]=m-1;//將m-1的值存入下一層
            stack[++top]=m;//再將m存入下一層
            n=n-1;
            //達到A(m-1,A(m,n-1))的效果;
        }
    }
    return n;
}
```

效能分析_遞迴

時間複雜度:

T(P)=基於 m 和 n 的指數成長

空間複雜度:

S(P)=O(m+n);

效能分析_非遞迴

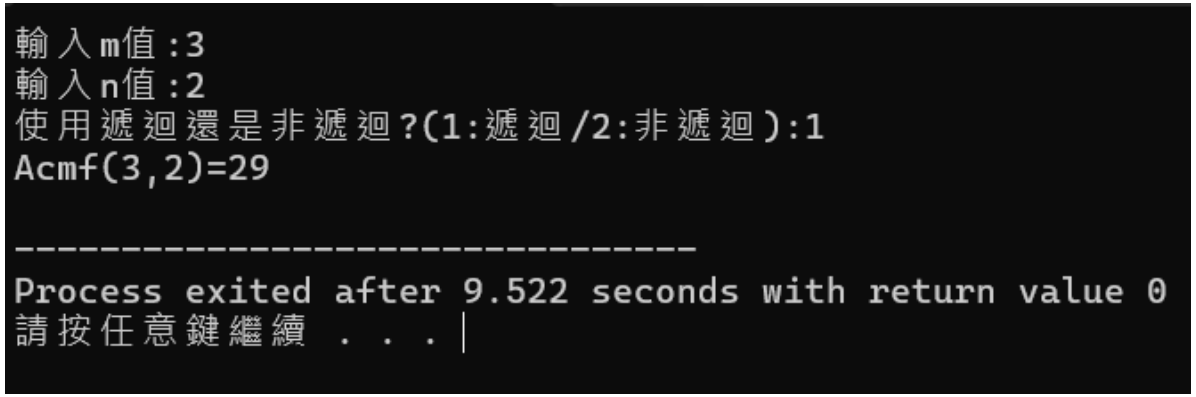
時間複雜度:

T(P)=基於m和n的指數成長

空間複雜度:

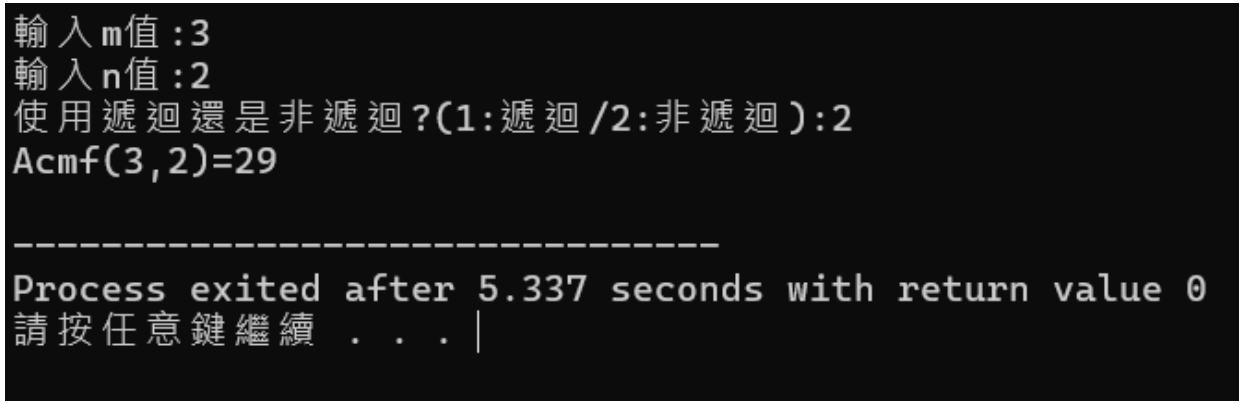
S(P)=O(1)
靜態數組大小固定

測試&過程_遞迴


輸入m值:3
輸入n值:2
使用遞迴還是非遞迴?(1:遞迴/2:非遞迴):1
Acmf(3,2)=29

Process exited after 9.522 seconds with return value 0
請按任意鍵繼續 . . .

測試&過程_非遞迴


輸入m值:3
輸入n值:2
使用遞迴還是非遞迴?(1:遞迴/2:非遞迴):2
Acmf(3,2)=29

Process exited after 5.337 seconds with return value 0
請按任意鍵繼續 . . .

心得&申論

心得:

雖然是我的問題,但在半夜兩點寫完程式才發現不能使用STL容器有多幹嗎?五天!我他媽光是程式就搞了五天!真的，我當下超想哭的...

申論:

阿卡曼函數展示了遞迴的極限和計算的複雜性。透過遞迴與非遞迴實現，可以深入理解其特性，對於學生的演算法設計具有非~常~好啟發，probably