

資料結構報告

紀宇駿

July 30, 2024

CONTENTS

1	解題說明	2
2	演算法設計與實作	3
3	效能分析	5
4	測試與過程	6-7

CHAPTER 1

解題說明

將ackermann函數表示為ackermann (m, n)，其中m和n是非負整數。

函數定義如下：

1. 如果m等於0，則回傳n + 1。
2. 如果m大於0且n等於0，則回傳ackermann (m-1, 1)。
3. 如果m大於0且n大於0，則回傳ackermann (m-1, ackermann (m, n-1))。

將m和n的值減小，直到滿足其中一個情況。

實作參見檔案 hw1.cpp，其遞迴函式：

```
int ackermann(int m, int n) {  
    if (m == 0) {  
        return n + 1;  
    } // m為0, 回傳n+1  
    else if (n == 0) {  
        return ackermann(m - 1, 1);  
    } // n為0, 回傳m-1, n=1  
    else {  
        return ackermann(m - 1, ackermann(m, n - 1));  
    } // m, n不為0, 回傳m-1, n=ackermann(m, n-1)  
}
```

1
2
3
4
5

Figure 1.1: hw1.cpp

CHAPTER 2

演算法設計與實作

```
int main() {  
    int m, n;  
    cin >> m;  
    cin >> n;  
    cout << "Ackermann(" << m << ", " << n << ") = " << ackermann(m, n) << endl;  
    return 0;  
}
```

Figure 2.1: hw1.cpp

堆疊作法

實作參見檔案 hw1-2.cpp, 其堆疊函式:

```
int ackermann(int m, int n) {
    int array[100];
    int a = -1; //初始化堆疊
    array[++a] = m;
    while (a >= 0) {
        //若堆疊不為0,則繼續執行
        m = array[a--];
        if (m == 0) {
            n = n + 1;
        } //當m為0,回傳n+1
        else if (n == 0) {
            array[++a] = m - 1;
            n = 1;
        } //當n為0,m=m-1,n=1
        else {
            array[++a] = m - 1;
            array[++a] = m;
            n = n - 1;
        } //當m,n不等於0,m=m-1, n=ackermann(m, n - 1)
    }
    return n;
}
```

```
int main() {
    int m, n;
    cin >> m;
    cin >> n;
    cout << "ackermann(" << m << ', ' << n << ") = " << ackermann(m, n);
    return 0;
}
```

CHAPTER 3

效能分析

時間複雜度

當 $m = 0$ 時，函數的時間複雜度是 $O(1)$

當 $n = 0$ 且 $m > 0$ 時，函數的時間複雜度是 $O(n)$

當 $m > 0$ 且 $n > 0$ 時，函數的時間複雜度非常高

空間複雜度

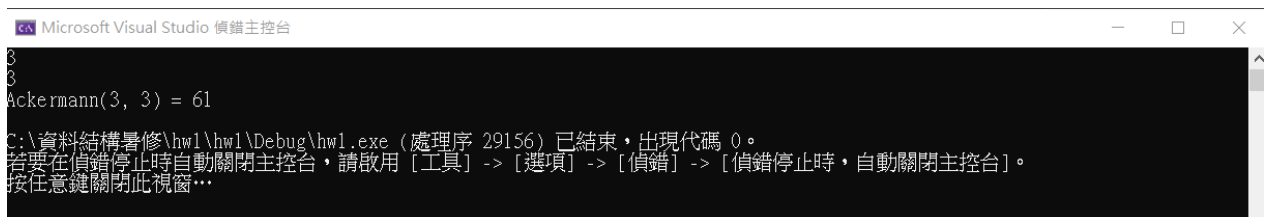
當 $m=0$ 時，空間複雜度是 $O(1)$

當 $m>0$ 且 $n=0$ 時，空間複雜度是 $O(m)$

當 $m>0$ 且 $n>0$ 時，空間複雜度是超多項式

CHAPTER 4

測試與過程



```
Microsoft Visual Studio 偵錯主控台
3
3
Ackermann(3, 3) = 61
C:\資料結構暑修\hw1\hw1\Debug\hw1.exe (處理序 29156) 已結束，出現代碼 0。
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [偵錯] -> [偵錯停止時，自動關閉主控台]。
按任意鍵關閉此視窗...
```

Figure 4.1: shell command

驗證

$$\begin{aligned} A(3, 3) &= A(2, A(3, 2)) \\ A(3, 2) &= A(2, A(3, 1)) \\ A(3, 1) &= A(2, A(3, 0)) \\ A(3, 0) &= A(2, 1) \\ A(2, 1) &= A(1, A(2, 0)) \\ A(2, 0) &= A(1, 1) \\ A(1, 1) &= A(0, A(1, 0)) \\ A(1, 0) &= A(0, 1) \\ A(0, 1) &= 2 \\ A(1, 0) &= 2 \\ A(1, 1) &= A(0, 2) = 3 \\ A(2, 0) &= 3 \\ A(2, 1) &= A(1, 3) = A(0, 4) = 5 \\ A(3, 0) &= 5 \\ A(3, 1) &= A(2, 5) = A(1, A(2, 4)) = 13 \\ A(3, 2) &= A(2, 13) = A(1, A(2, 12)) = 29 \end{aligned}$$

$$A(3, 3) = A(2, 29) = A(1, A(2, 28)) = 61$$