

資料結構報告

資工二甲 41243101 伍翊瑄

日期: 2024/10/23

目錄

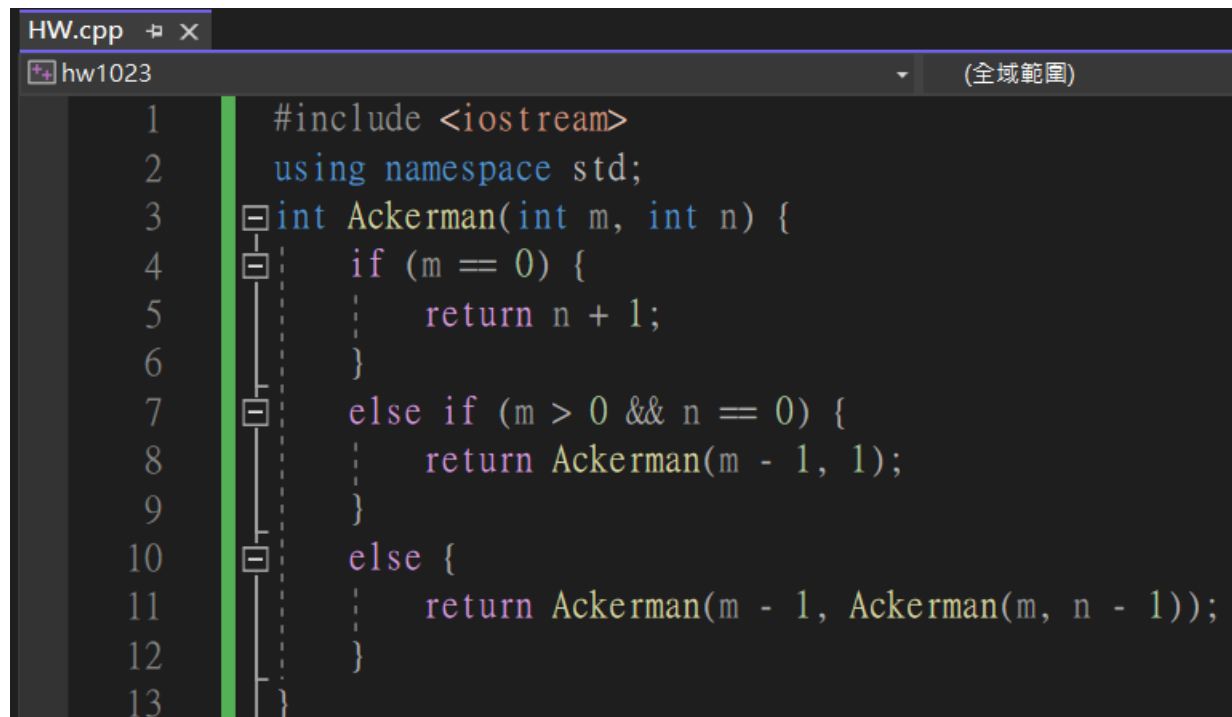
1 .	解題說明	3
2 .	演算法設計與實作	5
3 .	效能分析	6
4 .	測試與過程	7

一. 解題說明

1. 以遞迴實作計算阿克曼函數 $A(m,n)$ ，已知定義如下：

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

實作參見檔案 HW.cpp，其遞迴函式：



```
HW.cpp  [icon] [x]
hw1023 (全域範圍)
1  #include <iostream>
2  using namespace std;
3  int Ackerman(int m, int n) {
4      if (m == 0) {
5          return n + 1;
6      }
7      else if (m > 0 && n == 0) {
8          return Ackerman(m - 1, 1);
9      }
10     else {
11         return Ackerman(m - 1, Ackerman(m, n - 1));
12     }
13 }
```

2. 以非遞迴(堆疊)實作計算阿克曼函數 $A(m,n)$

實作參見檔案 HW1_2.cpp，其函式：

```
HW1_2.cpp
hw1023_1_2
(全域範圍)
Ackerman(int m, int n)

1  #include <iostream>
2  #include <stack>
3  using namespace std;
4
5  int Ackerman(int m, int n) {
6      stack<pair<int, int>> stk; // 使用堆疊來模擬遞迴
7      stk.push({ m, n }); // 將輸入的 m 和 n 推入堆疊
8      int result = 0; // 儲存和追蹤當前阿克曼函數的結果
9
10     while (!stk.empty()) {
11         auto cur = stk.top(); // 取得目前的(m, n)
12         stk.pop(); // 移除頂端元素
13         m = cur.first; // 提取 m
14         n = cur.second; // 提取 n
15
16         if (m == 0) {
17             result = n + 1; // Ackerman(0, n) = n + 1
18
19         else if (m == 1) {
20             result = n + 2; // Ackerman(1, n) = n + 2
21         }
22         else if (m == 2) {
23             result = 2 * n + 3; // Ackerman(2, n) = 2n + 3
24         }
25         else if (m == 3) {
26             result = 5; // Ackerman(3, 0) = 5
27             for (int i = 1; i <= n; i++) {
28                 result = 2 * result + 3;
29             }
30         }
31         else {
32             stk.push({ m - 1, result }); // 將需要處理的 (m-1, result) 推入堆疊，根據阿克曼函數的定義，計算 Ackerman(m, n) 會依賴於 Ackerman(m-1, A(m, n-1))
33             stk.push({ m, n - 1 }); // 計算 Ackerman(m, n-1)，根據阿克曼函數的定義，這是第一步必須進行的計算
34             continue; // 繼續處理堆疊
35         }
36     }
37     return result;
38 }
```

二. 演算法設計與實作

1. HW.cpp 的 main.cpp

```
45  int main() {  
46      int m, n;  
47      cout << "請輸入兩個正整數 m 和 n: ";  
48      cin >> m >> n;  
49      cout << "Ackerman(" << m << ", " << n << ") = " << Ackerman(m, n) << endl;  
50      return 0;  
51  }
```

2. HW1_2.cpp 的 main.cpp

```
39  int main() {  
40      int m, n;  
41      cout << "請輸入兩個正整數 m 和 n: ";  
42      cin >> m >> n;  
43      cout << "Ackerman(" << m << ", " << n << ") = " << Ackerman(m, n) << endl;  
44      return 0;  
45  }
```

三. 效能分析

時間複雜度

$T(P) = O(A(m,n))$ ，阿克曼函數本身的增長速度

當 m 增加時，阿克曼函數的輸出增長非常快。

因此，阿克曼函數的時間複雜度不是一個多項式函數，而是一個指數增長的函數。

1. 遞迴

空間複雜度

$S(P) = O(m+n)$ ，與遞迴的深度相關

在最壞情況下，遞迴的最大深度為 $m+n$ ，這是因為在每次遞迴中，當 m 和 n 都大於零時，遞迴會再次呼叫自己兩次。

2. 非遞迴

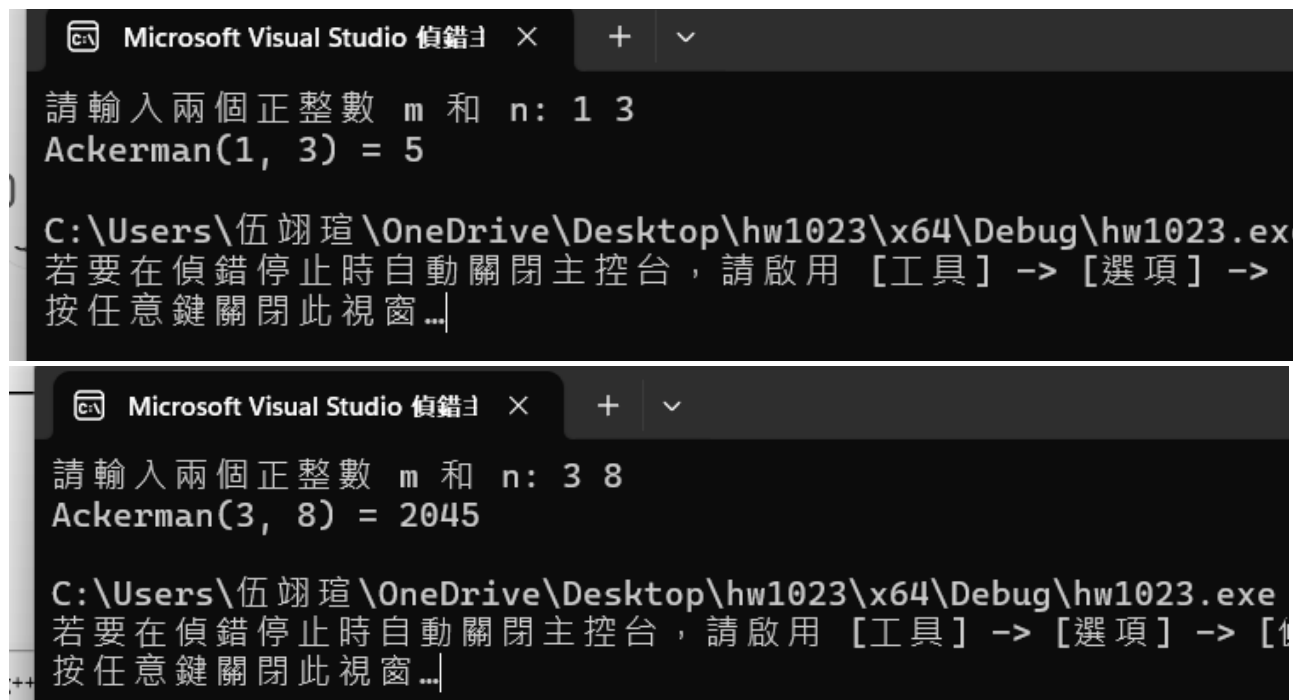
空間複雜度

$S(P) = O(m+n)$ ，與堆疊的深度相關

堆疊中的每個條目都需要儲存 m 和 n ，並且最壞情況下的深度與 m 和 n 的總和有關。

四. 測試與過程

1. 遞迴



The image shows two screenshots of a Microsoft Visual Studio console window. The top screenshot shows the input '1 3' and the output 'Ackerman(1, 3) = 5'. The bottom screenshot shows the input '3 8' and the output 'Ackerman(3, 8) = 2045'. Both screenshots include the file path 'C:\Users\伍翊瑄\OneDrive\Desktop\hw1023\x64\Debug\hw1023.exe' and a message about closing the console window.

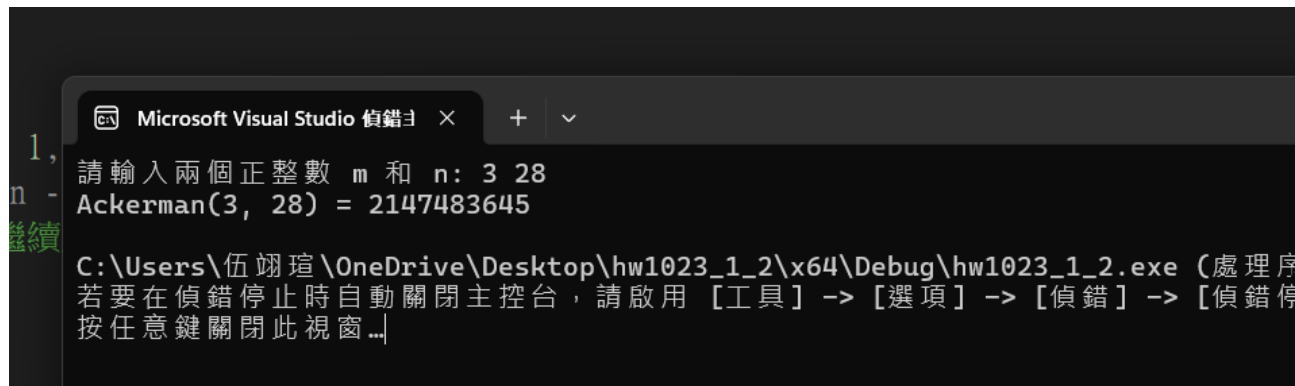
```
Microsoft Visual Studio 偵錯控制台 × + v
請輸入兩個正整數 m 和 n: 1 3
Ackerman(1, 3) = 5

C:\Users\伍翊瑄\OneDrive\Desktop\hw1023\x64\Debug\hw1023.exe
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [
按任意鍵關閉此視窗...
```

```
Microsoft Visual Studio 偵錯控制台 × + v
請輸入兩個正整數 m 和 n: 3 8
Ackerman(3, 8) = 2045

C:\Users\伍翊瑄\OneDrive\Desktop\hw1023\x64\Debug\hw1023.exe
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [
按任意鍵關閉此視窗...
```

2. 非遞迴



The screenshot shows a Visual Studio debugger window titled "Microsoft Visual Studio 偵錯器". The main text area displays the following content:

```
請輸入兩個正整數 m 和 n: 3 28
Ackerman(3, 28) = 2147483645

C:\Users\伍翊瑄\OneDrive\Desktop\hw1023_1_2\x64\Debug\hw1023_1_2.exe (處理序
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
按任意鍵關閉此視窗...]
```

驗證

進入函式，第一層 $m=1, n=3$ ， m 和 n 皆不為 0，回傳 $\text{Ackerman}(m-1, \text{Ackerman}(m, n-1))$ ，即 $A(0, A(1, 2))$ ，接著第二層， m 和 n 皆不為 0，回傳 $\text{Ackerman}(m-1, \text{Ackerman}(m, n-1))$ ，即 $A(0, A(0, A(1, 1)))$ ，接著第三層， m 和 n 皆不為 0，回傳 $\text{Ackerman}(m-1, \text{Ackerman}(m, n-1))$ ，即 $A(0, A(0, A(0, A(0, 1))))$ ，接著第四層， $m=0$ ，回傳 $n+1$ ，即 $A(0, A(0, A(0, 1+1)))$ ，接著第五層， $m=0$ ，回傳 $n+1$ ，即 $A(0, A(0, 2+1))$ ，接著第六層， $m=0$ ，回傳 $n+1$ ，即 $A(0, 3+1)$ ，接著第七層， $m=0$ ，回傳 $n+1$ ，即 $4+1$ 。

$$\begin{aligned} A(1, 3) &= A(0, A(1, 2)) \\ &= A(0, A(0, A(1, 1))) \\ &= A(0, A(0, A(0, A(0, 1)))) \\ &= A(0, A(0, A(0, 1+1))) \\ &= A(0, A(0, 2+1)) \end{aligned}$$

$$=A(0,3+1)$$

$$=4+1$$

$$=5$$