# HW1 第一題

## 解題說明

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.

如上方之題目要求，實作阿克曼函數的遞迴寫法跟非遞迴的寫法

1. 遞迴寫法

```
int AckermannRecursive(int m, int n)
{
    //如果m=0 ,回傳n+1
    if (m == 0)
        return n + 1;
    //如果n=0 進入AckermannRecursive(m - 1, 1)
    else if (n == 0)
        return AckermannRecursive(m - 1, 1);
    //不滿足上述條件，進入AckermannRecursive(m - 1, 1)
    else
        return AckermannRecursive(m - 1, AckermannRecursive(m, n - 1));

}
```

如上圖，題目所給的條件已經包含遞迴所需的過程與結束條件，直接套用即可

2. 非遞迴寫法

```
int AckermannNonrecursive(int m,int n)
{
    stack<int> s;// 用堆疊來模擬遞迴的進行
    s.push(m);  //  將 m 推入堆疊
    while (!s.empty())
    {
        m = s.top();//將m放入堆疊的頂端
        s.pop();// 將堆疊頂端元素彈出
        // 如果 m 等於 0，結果為 n + 1
        if (m == 0)
            n = n + 1;
        // 如果 m > 0 且 n 等於 0，則將 m-1 推入堆疊，並將 n 設為 1
        else if (n == 0)
        {
            n = 1;
            s.push(m - 1);
        }
        // 如果 m > 0 且 n > 0，先將 m-1 推入堆疊，再次將 m 推入堆疊，並將 n 減 1
        else
        {
            s.push(m - 1);
            s.push(m);
            n = n - 1;
        }
    }
    return n;
}
```

如上圖，我使用堆疊的方式來完成，每當需要遞迴時，就將參數 m 加入堆疊。當 m == 0 時，直接計算結果並更新 n，當 n == 0 且 m > 0 時，更新 n = 1 並加入 m-1，當 m > 0 且 n > 0 時，依照 Ackermann 函數的定義，執行兩次堆疊壓入操作來模擬兩次遞迴。

## 效能分析

1. 當 m=0 時
   時間複雜度為 O(1)
2. 當 m=1 時
   時間複雜度為 O(n)
3. 當 m=3 或更高時
   時間複雜度為 O(2^2^2^n)或者更高

## 測試與驗證

下圖為阿克曼函數之執行結果表，圖片取自維基百科

| *m*\n | 0 | 1 | 2 | 3 | 4 | n |
|---|---|---|---|---|---|---|
| **0** | 1 | 2 | 3 | 4 | 5 | $n+1$ |
| **1** | 2 | 3 | 4 | 5 | 6 | $n+2$ |
| **2** | 3 | 5 | 7 | 9 | 11 | $2\cdot(n+3)-3$ |
| **3** | 5 | 13 | 29 | 61 | 125 | $2^{(n+3)}-3$ |
| **4** | 13 | 65533 | $2^{65536}-3$ | $A(3, 2^{65536}-3)$ | $A(3, A(4, 3))$ | $\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}-3$（n+3個數字2） |
| **5** | 65533 | $A(4, 65533)$ | $A(4, A(5, 1))$ | $A(4, A(5, 2))$ | $A(4, A(5, 3))$ | |
| **6** | $A(5, 1)$ | $A(5, A(5, 1))$ | $A(5, A(6, 1))$ | $A(5, A(6, 2))$ | $A(5, A(6, 3))$ | |

如下為我的程式測試結果

```
輸入 m 和 n: 2 2
Ackermann's Rescursive(2, 2) = 7
Ackermann's NonRescursive(2, 2) = 7
輸入 m 和 n: 2 3
Ackermann's Rescursive(2, 3) = 9
Ackermann's NonRescursive(2, 3) = 9
輸入 m 和 n: 3 3
Ackermann's Rescursive(3, 3) = 61
Ackermann's NonRescursive(3, 3) = 61
輸入 m 和 n: 2 5
Ackermann's Rescursive(2, 5) = 13
Ackermann's NonRescursive(2, 5) = 13
輸入 m 和 n: 2 7
Ackermann's Rescursive(2, 7) = 17
Ackermann's NonRescursive(2, 7) = 17
```

## 心得與申論

這次的作業是做阿克曼函數的遞迴跟非遞迴版本，遞迴的部分很簡單，題目上都已經有所有的條件了，照著打就能完成，主要是難在非遞迴的部分，本來打算用迴圈去實現，但實際做了之後發現似乎迴圈的最後也會用到遞迴，只是表達的方式不一樣，因此，最後採用堆疊的方式，收益良多。