1.解題說明

使用一個 2 維陣列 temp 來儲存已經計算過的結果，以避免重複遞迴計算。當呼叫 ackermann(m, n) 時，程式會先檢查是否已經計算過該結果，如果有，則直接從陣列中取值。如果沒有，則根據 Ackermann 函數的定義進行遞迴計算。遞迴過程中會記錄總共執行了多少次遞迴。

2.效能分析

一般遞迴寫法:

優點: 可以處理較大的數字組合，例如 A(3,14)，極限取決於系統的記憶體和遞迴深度。如果主記憶體足夠大，沒有明確的計算極限，主要是時間的問題。

缺點: 遞迴次數比優化版本多得多，因此每次計算需要花費大量時間，且重複計算之前已經算過的結果。

優化寫法 (使用陣列儲存結果):

優點: 遞迴次數大幅減少，因為會記錄已經計算過的結果，因此計算速度較快。

缺點: 儘管使用一個額外的陣列來儲存結果，但在處理大範圍的 m 和 n 時，這個陣列的大小會受到 C++ 陣列大小的限制，可能無法計算超過 A(3,13) 的結果。

3.測試與驗證

```
請輸入 m 和 n 的值：3 13
一般遞迴：
A(3, 13) = 65533
總共遞迴:2862983902次
優化遞迴：
A(3, 13) = 65533
總共遞迴:196622次
```

4.申論及心得

Ackermann's function 是一個著名的時間複雜度問題，它以非常恐怖的速度增長。即使在當今科技盛行的社會，計算 Ackermann 函數對電腦來說仍然是一個極大的負載。由於其遞迴定義的特性，這個函數的運算時間和空間需求會隨著輸入值的增加而迅速上升，導致在處理較大的參數時，程式可能面臨堆疊溢出、性能下降等問題。因此，理解和應對這一挑戰對於計算理論及程式設計都是非常重要的。