資料結構報告範例

王語晨

OCT 31, 2024

CONTENTS

CHAPTER 1

解題說明

以阿克曼函數實作，已知阿克曼函數計算公式如下：

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

實作參見檔案 acker.cpp，其acker函式：

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

Figure 1.1: acker.cpp

CHAPTER 2

演算法設計與實作

```
int main() {
	cout << acker(1, 1);
	return 0;
}
```

Figure 2.1: main.cpp

CHAPTER 3

效能分析

$$f(n) = O(n)$$

時間複雜度

$$T(m,n) = O(A(m,n))$$

超指數級（無法用簡單函數描述，取決於m和n的值）。

空間複雜度

$$S(m,n) = O(\text{遞迴深度})$$

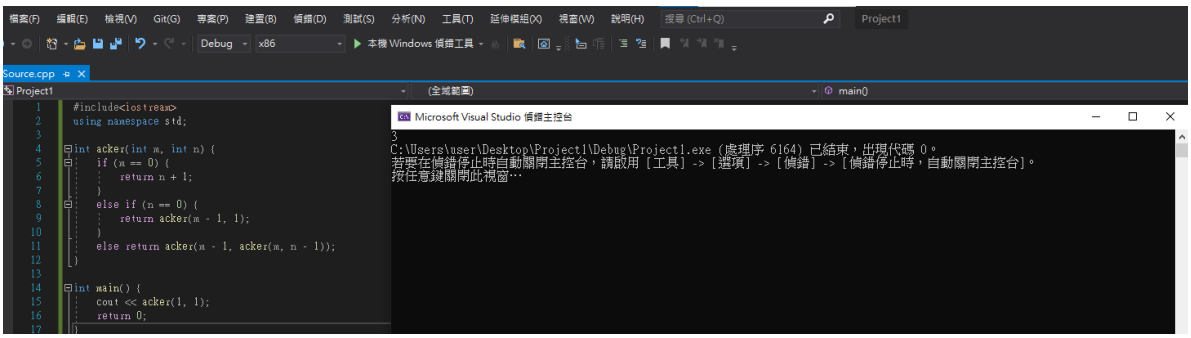
超指數級（由遞回深
度局定，隨著m和n的
值增長急劇增加）。

CHAPTER 4

測試與過程

```
$ g++ main.cpp -o main.exe && ./main.exe
3
```

Figure 4.1: shell command



驗證

在主函式main（）中，程式執行cout<<acker(1,1)。根據Ackermann函數的定義,當m不等於0，n也不等於0時，應該回傳acker(m-1,acker(n-1))。因此，acker(1,1)應該輸出1+2=3。

此首先第一層，m不等於0所以進入第二層，但n也不等於0所以進入第三層，進行遞迴呼叫acker(0,acker(1,0))，然後需要計算acker(1,0)，它會進入第二個條件，進行遞迴呼叫acker(0,1)會回傳1+1=2，然後回到acker(1,1)，相當於acker(0,2)，會回傳2+1=3。

CHAPTER 5

申論與心得

申論

這題要求實現 Ackermann 函數，它是一種經典的遞迴函數，具有非常高的增長速率。Ackermann 函數在計算複雜性中通常被用來測試遞迴能力和資源限制的標準。

程式透過遞迴函數 acker來實現 Ackermann 函數。函數使用三層條件判斷結構：

當（ m = 0 ）時，返回（ n + 1 ）。

當（ n = 0 ）且（ m > 0 ）時，返回 acker(m - 1, 1)。

當（ m > 0 ）且（ n > 0 ）時，返回 acker(m - 1, acker(m, n - 1))。

透過這樣的遞迴結構，Ackermann 函數可以計算出不同 m 和 n 所產生的結果。

由於 Ackermann 函數的遞迴深度隨著 m 和 n 的值快速增加，導致此函數的時間複雜度和空間複雜度呈超指數增長。

程式的計算過程如下： 1.acker(1, 1) → acker(0, acker(1, 0)) 2.acker(1, 0) → acker(0, 1) = 2 3.acker(1, 1) → acker(0, 2) = 3，最終結果為 3。

心得

透過這份作業更加了解Ackermann 函數，也感受到它操作上的複雜，它不但展示了遞迴處理的威力及極限，也說明了計算資源和效率在面對快速增長的函數時的重要性。