資料結構 課堂作業
HW1 Problem 1 & Problem 2
姓名: 廖章竹
日期: **10/22**

CHAPTER 1: 解題說明

Problem 1: Ackermann Function

此問題要求實作 Ackermann 函數，分為遞迴 (Recursive) 及非遞迴 (Non-Recursive) 版本。
Ackermann 函數是一個典型的遞迴函數，其定義如下：

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

Problem 2: Powerset

此問題要求生成集合的冪集，利用遞迴來實作所有子集的產生過程。冪集是所有可能的子集組合，包括空集與全集。

CHAPTER 2: 演算法設計與實作

Problem 1: Ackermann Function

Recursive

```
 6   int ack_r(int m,int n){
 7       if(m==0){
 8           return n+1;
 9       }
10       else if(m>0 && n==0){
11           return ack_r(m-1,1);
12       }
13       else{
14           return ack_r(m-1,ack_r(m,n-1));
15       }
16   }
```

Non-recursive

```
18   int ack_nr(int m, int n) {
19       while (m != 0) {
20           if (m > 0 && n == 0) {
21               // 當m > 0且n == 0時
22               m = m - 1;
23               n = 1;
24           } else if (m > 0 && n > 0) {
25               // 當m > 0且n > 0時
26               int temp_n = n;
27               n = n - 1;
28               m = m - 1;
29               //(m-1, ackermann(m, n-1))
30               n = ack_nr(m + 1, temp_n - 1);  //手動展開遞歸
31           }
32       }
```

Problem 2: Powerset

```
// 遞迴函數生成冪集
void powerset(int set[], int subset[], int set_size, int subset_size, int idx) {
    // 如果索引超出集合大小，列印當前子集
    if (idx == set_size) {
        cout << "{ ";
        for (int i = 0; i < subset_size; i++) {
            if (i != 0) cout << ", ";
            cout << subset[i];
        }
        cout << " }" << endl;
        return;
    }

    // 情況1：不包含當前元素
    powerset(set, subset, set_size, subset_size, idx + 1);

    // 情況2：包含當前元素
    subset[subset_size] = set[idx];  // 將當前元素加入子集
    powerset(set, subset, set_size, subset_size + 1, idx + 1);
}
```

CHAPTER 3: 效能分析

Problem 1: Ackermann Function

時間複雜度:

- 由於遞迴深度可能非常深，Ackermann 函數的時間複雜度極高，尤其在 m 增大時，時間複雜度近似於指數增長。

空間複雜度:

- 遞迴版本：需要依賴系統堆疊，空間複雜度為 O(m)O(m)O(m)。
- 非遞迴版本：雖然避免了系統堆疊溢出問題，但運算過程中仍需額外儲存部分狀態。

時間複雜度:

- 生成冪集的時間複雜度為 O(2n) O(2^n) O(2n)，因為每個元素有兩種狀態（包含或不包含），需要遍歷所有可能的子集。

空間複雜度:

- 空間複雜度為 O(n) O(n) O(n)，用於存儲當前子集。

CHAPTER 4: 測試與過程

Problem 1 測試

```
PS D:\資料結構\homework1> .\ack.exe
input m n:3 2
Ack_r(3,2) = 29
Ack_nr(3,2) = 29
PS D:\資料結構\homework1> .\ack.exe
input m n:1 2
Ack_r(1,2) = 4
Ack_nr(1,2) = 4
```

Problem 2 測試

```
PS D:\資料結構\homework1> .\problem2.exe
Powerset:
{  }
{ 3 }
{ 2 }
{ 2, 3 }
{ 1 }
{ 1, 3 }
{ 1, 2 }
{ 1, 2, 3 }
```

測試結果驗證：

- **Ackermann** 函數的兩種實作結果相同，驗證了非遞迴版本的正確性。
- **Powerset** 程式正確地生成了集合 **{1, 2, 3}** 的所有子集。

Ackermann 函數的實作，通過遞迴與非遞迴兩種方式進行了實現。在測試過程中，兩種實作方式的輸出結果一致，說明非遞迴版本成功模擬了遞迴運算，並有效解決了遞迴深度可能過深的問題。

Powerset 的生成部分，遞迴方法在處理這類問題上較直觀與便利。