# 資料結構 第一周 作業

姓名:林國善

7/30/2024

# 阿克曼函數

## 1 解題說明

阿克曼函數是非原始遞迴函數；它需要兩個自然數作為輸入值，輸出一個自然數。因為上機實習時有講道是一個遞迴的關係我用上課時交的東西配合這個算式。

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

## 2 演算法設計與實作

```cpp
#include <iostream>
#include <string>

using namespace std;

// 阿克曼函數函数
int a(int m,int n) {
    if (m == 0) {
        return n + 1;
    }
    else if (m > 0 && n == 0) {
        return a(m - 1, 1);
    }
    else if (m > 0 && n > 0) {
        return a(m - 1, a(m, n - 1));
    }
    return 0;
}

int main() {
    int m, n;
    cout << "輸入m和n的值: ";
    cin >> m >> n;

    cout << "阿克曼函數(" << m << ", " << n << ") = " << a(m, n) << endl;

    return 0;
}
```

3 效能分析

時間複雜度: Ackermann 函數的時間複雜度極高，由於函數的遞迴深度隨著 m 和 n 值的增加而指數級增長，因此其時間複雜度遠超指數級，通常表示為非常高的複雜度。

空間複雜度: 空間複雜度主要由遞迴調用的深度決定，遞迴深度為 O(A(m, n))，即 Ackermann 函數的結果大小。

4 測試與過程

輸入

0 0

輸出

```
輸入m和n的值: 0 0
阿克曼函數(0, 0) = 1
```

輸入

3 4

輸出

```
輸入m和n的值: 3 4
阿克曼函數(3, 4) = 125
```

# Problem 2: Powerset

## 1 解題說明

集合的所有子集的集合。計算一個集合可以使用遞迴方法，但可以通過遞迴和回溯來解決這個問題。

## 2 演算法設計與實作

使用遞迴來生成子集。對於每個元素，都有兩種選擇：包括在子集中或不包括在子集中。

```cpp
#include <iostream>
#include <vector>
using namespace std;
void gen(vector<int>& s, int i, vector<int> c, vector<vector<int>>& r) {
    if (i == s.size()) {
        r.push_back(c);
        return;
    }
    // 不包含當前元素
    gen(s, i + 1, c, r);
    // 包含當前元素
    c.push_back(s[i]);
    gen(s, i + 1, c, r);
}
vector<vector<int>> ps(vector<int>& s) {
    vector<vector<int>> r;
    vector<int> c;
    gen(s, 0, c, r);
    return r;
}
int main() {
    vector<int> s = { 1, 2, 3 };
    vector<vector<int>> r = ps(s);

    cout << "Powerset: " << endl;
    for (const auto& subset : r) {
        cout << "{ ";
        for (int elem : subset) {
            cout << elem << " ";
        }
        cout << "}" << endl;
    }
    return 0;
}
```

## 3 效能分析

時間複雜度: 產生冪集的時間複雜度是 O(2n)O(2^n)O(2n)，因為每個元素都有兩種選擇。

空間複雜度: 空間複雜度也是 O(2n)O(2^n)O(2n)，因為中包含 2n2^n2n 個子集，每個子集的大小最多為 nnn。

4 測試與過程

輸入

{ 1, 2, 3 }

輸出