資料結構報告

張哲維

July 30, 2024

CONTENTS

4 測試與過程

CHAPTER 1

解題說明

遞迴實作計算阿克曼函式，條件是如下

Problem 1:

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.

1

其遞迴函是如下 參考 ack.cpp

```cpp
#include <iostream>

using namespace std;

int ackermann(int m,int n) {
    if (m == 0) {
        return n + 1;   //假如m=0,n+1
    } else if (n == 0) {
        return ackermann(m - 1, 1);//假如n=0,m-1
    } else {
        return ackermann(m - 1, ackermann(m, n - 1));//都不是的話就跳這行
    }
}
```

2.非遞迴實作計算阿克曼函式

參考 ack-nonrecursive.cpp

```cpp
#include <iostream>
#include <stack>

using namespace std;

int Ackerman(int m, int n) {
    stack<pair<int,int>> s;
    s.push({m, n});              //將m跟n放入堆疊中

    while (!s.empty()) {         //當s堆疊內不為空值則運行while
        pair<int,  int> top = s.top();         //
        m = top.first;
        n = top.second;
        s.pop();

        if (m == 0) {
            if (!s.empty()) {
                s.top().second = n + 1;
            } else {
                return n + 1;                  //s堆疊內為空值, (m,n) = (0,0), 則輸出為1
            }
        } else if (n == 0) {
            s.push({m - 1, 1});                //m!=0, n=0 判斷這行
        } else {                               //m=0, n!=0,跳這行
            s.push({m - 1, 0});                //將(m-1,(m,n-1))拆為(m-1,0)
            s.push({m, n - 1});                //將(m-1,(m,n-1))拆為(m,n-1)
        }
    }

    return 0;
}
```

二.

以遞迴計算集合內組合 計算公式如下

Problem 2:

If S is a set of n elements, the *powerset* of S is the set of all possible subsets of S. For example, if $S = (a,b,c)$, then *powerset* $(S) = \{(), (a), (b), (c), (a,b), (a,c), (b,c), (a,b,c)\}$. Write a recursive function to compute *powerset* (S).

生成一個字串集合的所有子集（冪集）。

n 個元素的組合共有：2n 種組合

參考 powerset.cpp

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

using namespace std;

template <typename T>
void generate_powerset(const vector<T>& s, vector<vector<T>>& result, vector<T>& current, int index) {
    if (index == s.size()) {
        result.push_back(current);
        return;
    }
    generate_powerset(s, result, current, index + 1);
    current.push_back(s[index]);
    generate_powerset(s, result, current, index + 1);
    current.pop_back();
}

template <typename T>
vector<vector<T>> powerset(const vector<T>& s) {
    vector<vector<T>> result;
    vector<T> current;
    generate_powerset(s, result, current, 0);
    return result;
}

int main() {
    vector<string> s = { "a", "b", "c" };
    vector<vector<string>> result = powerset(s);

    cout << "Powerset:" << endl;
    for (const auto& subset : result) {
        cout << "{ ";
        for (const auto& element : subset) {
            cout << element << " ";
        }
        cout << "}" << endl;
    }

    return 0;
}
```

CHAPTER3

效能分析

時間複雜度

$$T(P) = n \times C$$

每層迴圈所需 C 時間、n 次遞迴。

空間複雜度

S(P) = 2 × n

2 個變數、n 次遞迴

CHAPTER 4

測試與過程

1-1

```
請輸入 m 和 n 的值: 2 3
Ackermann(2, 3) = 9
PS C:\Users\Administrator\Desktop\C++> ^C
PS C:\Users\Administrator\Desktop\C++>
PS C:\Users\Administrator\Desktop\C++>  & 'c:\Users'
' '--stdout=Microsoft-MIEngine-Out-q3xo5ak3.z5x' '--
請輸入 m 和 n 的值: 3 2
Ackermann(3, 2) = 29
```

1-2

```
' '--stdout=Microsoft-MIEngine-Out-c0y3ik1r.
請輸入 m 和 n 的值: 2 3
ackermann_iterative(2, 3) = 9
PS C:\Users\Administrator\Desktop\C++> ^C
PS C:\Users\Administrator\Desktop\C++>
PS C:\Users\Administrator\Desktop\C++>  & 'c
' '--stdout=Microsoft-MIEngine-Out-zsdlclis.
請輸入 m 和 n 的值: 3 2
ackermann_iterative(3, 2) = 29
PS C:\Users\Administrator\Desktop\C++>
```

```
Powerset:
{ }
{ c }
{ b }
{ b c }
{ a }
{ a c }
{ a b }
{ a b c }
```

2-1

驗證

1. **初始狀態**: 當計算 A(1, 1) 時，由於 m > 0 且 n > 0，根據函數定義，計算 A(m - 1, A(m, n - 1))，即 A(0, A(1, 0))。

2. **遞迴展開**: 先計算 A(1, 0)

對於 A(1, 0)，因為 m > 0 且 n = 0，根據函數定義，返回 A(m - 1, 1)，即 A(0, 1)。

A(0, 1) 的計算。根據函數定義，當 m = 0 時，返回 n + 1。因此，A(0, 1) = 2。

3. 計算 A(0, A(1, 0)) 的步驟，已經知道 A(1, 0) = 2，所以計算 A(0, 2)。

A(0, 2)，因為 m = 0，根據定義返回 n + 1，即 2 + 1 = 3。

因此，A(1, 1) = 3。