資料結構 homework1

41243125 柯建佑

1.解題說明:

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.

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).

Ackermann function (有遞迴)

```cpp
#include<iostream>

using namespace std;

int ack(int m, int n) {
    if (m == 0)
        return n + 1;        // Ackermann 函數的第一個基本情況 if m=1 函數值 n+1
    else if ((m>0)&&(n == 0))
        return ack(m - 1, 1);          // Ackermann 函數的第二個基本情況 if m>0 and n=0, 返回ack(m - 1, 1)
    else if ((m > 0) && (n > 0))
        return ack(m - 1, ack(m, n - 1)); //Ackermann 函數的遞歸情況：如果 m&&n>0 則返回ack(m - 1, ack(m, n - 1))
}
int main() {
    int n, m,a;
    cin >> m >> n;   //輸入m,n值
    a = ack(m, n);    //讓a = ackermann function 的答案
    cout << a << endl;
    return 0;
}
```

Ackermann function (無遞迴)

```cpp
#include<iostream>
#include<stack>
 using namespace std;

int ackermann(int m, int n) {
    stack<int> s;
    s.push(m);              //將初始的m值推入堆疊

    while (!s.empty()) {
        m = s.top();
        s.pop();           //將堆疊頂部的值彈出並把值給 m, 模擬了每次從遞歸中返回上一層的情況,

        if (m == 0) {
            n = n + 1;
        }
        else if (n == 0) {
            s.push(m - 1);
            n = 1;        //如果達成條件就會把 m-1 推入堆疊 and 把n設為1, 這模擬了遞歸的情況 ack(m - 1, 1)
        }
        else {
            s.push(m - 1);
            s.push(m);
            n = n - 1;  //先將 m - 1 推入堆疊，然後將當前的 m 也推入堆疊，並減少 n。這對應於 ack(m - 1, ack(m, n - 1)) 的情況。
        }
    }

    return n;
}

int main() {
    int m, n, a;
    cin >> m >> n;
    a = ackermann(m, n);
    cout << "Ackermann(" << m << ", " << n << ") = " << a << endl;
    return 0;
}
```

Problem2:

```cpp
#include <iostream>
using namespace std;

void func(char word[], int count[], int n, int index) {
    // 當處理完所有元素時，輸出結果
    if (index == n) {
        cout << "{ ";
        for (int i = 0; i < n; i++) {
            if (count[i] == 1) {
                cout << word[i] << " ";
            }
        }
        cout << "}" << endl;
        return;
    }

    // 不選擇當前索引位置的字元
    count[index] = 0;
    func(word, count, n, index + 1);

    // 選擇當前索引位置的字元
    count[index] = 1;
    func(word, count, n, index + 1);
}

int main() {
    // 定義字元集合
    char word[3] = { 'a', 'b', 'c' };
    // 用來標記每個字元是否被選中
    int count[3] = { 0, 0, 0 };

    // 調用函數生成子集
    func(word, count, 3, 0);

    return 0;
}
```

心得:

這次的作業讓我學到了使用堆疊來代替遞迴

讓我的程式之路又多了一個工具來使用