Homework1

姓名

41243155　鄭師墉

10. 23, 2024

==========

CONTENTS

==========

=CHAPTER 1==========

解題說明

==========

Pronblem 1_a

該程式實現了阿克曼函數（Ackermann function）遞迴的計算。已知公式如下：

$$A(m,n) = \begin{cases} n+1 \\ A(m-1,1) \\ A(m-1,A(m,n-1)) \end{cases}$$

實作請參考附件pronblem1_a.cpp。

Pronblem 1_b

該題為Pronblem1_a的衍生題，內部計算使用非遞迴來做計算。
詳情請見附件pronblem1_b.cpp。

Pronblem 2

這個問題要求我們寫一個遞歸函數來計算一個集合的冪集。冪集是給定集合的所有子集，包括空集和集合本身。

S = (a, b, c)

power (S) = {(), (a), (b), (c), (a,b), (a,c), (b,c), (a,b,c)}

=CHAPTER 2==========

演算法設計與實作

==========

```cpp
long long ack[10][100000] = {0}; //設定 m,n 計算過程的上限

long long akm (int m, int n) {
    if(ack[m][n] != 0) {
        return ack[m][n];    //如果(m, n)計算過的話回傳結果
    }

    if(m == 0) ack[m][n]=n+1;    //阿克曼函數的計算
    else if(n == 0) ack[m][n]=akm(m-1, 1);
    else ack[m][n]=akm(m-1,akm(m, n-1));

    return ack[m][n];
}
```

Figure 2.1：pronblem1_a.cpp

```cpp
const int MAX_STACK_SIZE = 100000;  //設定stack最大容量

int akm(int m, int n) {
    int stackM[MAX_STACK_SIZE];  //分別存取m, n 的stack
    int stackN[MAX_STACK_SIZE];

    int top = -1;  //堆疊指標初始化 -1

    top++;  //將 m and n 存入
    stackM[top] = m;
    stackN[top] = n;

    while (top >= 0) {  //阿克曼函數的計算
        m = stackM[top];
        n = stackN[top];
        top--;  //彈出頂端的 m and n

        if (m == 0) {
            n = n + 1;
        } else if (n == 0) {
            top++;
            stackM[top] = m - 1;
            stackN[top] = 1;
        } else {
            top++;
            stackM[top] = m - 1;
            stackN[top] = -1;

            top++;
            stackM[top] = m;
            stackN[top] = n - 1;
        }

        if (top >= 0 && stackN[top] == -1) {  //指標 == 0 and n ==-1，結束
            stackN[top] = n;  //用 n 更新結果
        }
    }
    return n;
}
```

Figure 2.2：pronblem1_b.cpp

=CHAPTER 2==========

演算法設計與實作

==========

```cpp
//遞歸函數生成冪集
void powerset(char set[], string current, int index, int setSize) {
    if (index == setSize) {  //處理完所有元素，列印當前子集
        cout << "{" << current << "}" << endl;
        return;
    }
    //結果1. 不把元素放入子集，處理下一個
    powerset(set, current, index + 1, setSize);

    //結果2. 把當前元素放入子集，處理下一個
    if (!current.empty()) {
        current += ", ";
    }
    current += set[index];  //加入當前元素
    powerset(set, current, index + 1, setSize);
}
```

Figure 2.3：pronblem2.cpp

=CHAPTER 3==========

效能分析

==========

Pronblem 1_a

- **時間複雜度**：

$$T(P)=O(m \cdot n)$$

  m **是阿克曼函數中的參數範圍（在此極限為** 10）
  n **是阿克曼函數中的參數範圍**

- **空間複雜度**：

$$S(P)=O(m \cdot n)+O(1)$$

  m **是阿克曼函數中的陣列大小（在此為** 10）
  n **是阿克曼函數中的陣列大小（在此為** 100000）
  O(1) **是一些固定大小的變量（如** m、n **和返回**值）

Pronblem 1_b

- **時間複雜度**：

$$T(P)=O(2^n)$$

  n **是函數的輸入規模**

- **空間複雜度**：

$$S(P)=O(1) \text{ or } O(100000)$$

  陣列最大值為100000

=CHAPTER 3==========

效能分析

==========

Pronblem 2

- **時間複雜度：**

$$T(P)=O(n\cdot 2^n)$$

`2^n` **是子集的數量**
每個子集的生成最多需要 `O(n)` **的時間來構建字符串**

- **空間複雜度：**

$$S(P)=O(n)$$

`n` **是堆疊需要的空間**

=CHAPTER 4==========

測試與過程

==========

Pronblem 1_a

```
1_a.cpp -o pronblem1_a } ; if ($?) { .\pronblem1_a }
1 2
4
PS C:\Users\justi\Desktop\Add_new_folder\work\project\data_structure\homework1_>
```

Figure 4.1：pronblem1_a.cpp

驗證

終止條件為n=0&n=0。進入程式後 m=1 和 n=2 皆不等於0，回傳akm(0, akm(1, 1))給記憶陣列。akm(1, 1)皆不等於0，回傳akm(0, akm(1, 0))。akm(1, 0)因為n=0，回傳akm(0, 1)。akm(0, 1)回傳ack[0][1]=1+1=2 ... 以此類推

akm(1, 2)=2+1+1=4

Pronblem 1_b

```
1_b.cpp -o pronblem1_b } ; if ($?) { .\pronblem1_b }
3 4
125
PS C:\Users\justi\Desktop\Add_new_folder\work\project\data_structure\homework1_>
```

Figure 4.2：pronblem1_b.cpp

驗證

終止條件為top>=0。進入程式後將 m and n 存入個別陣列。進入while後 m=3 和 n=4 皆不等於0，top++, stackM[0] = 2, stackN[0] = -1, top++, stackM[1] = 3, stackN[1] = 3。m = stackM[1] = 3, n = stackN[1] = 3 ... 以此類推 最終top = -1, n=125。

akm(3, 4)=125

=CHAPTER 4==========

測試與過程

==========

Pronblem 2

```
\project\data_structure\homework1_\" ; if ($?) { g++ pronblem2.cpp -o pronblem2 } ; if ($?) { .\pronb
lem2 }
input S element total: 2
a b
powerset:
{}
{b}
{a}
{a, b}
PS C:\Users\justi\Desktop\Add_new_folder\work\project\data_structure\homework1_>
```

Figure 4.3：pronblem2.cpp

驗證

終止條件為index == setSize。進入程式後index = 0，不包含 a，進入下一個層次處理 index = 1。不包含 b，到達 index = 2，打印 {}。包含 b，到達 index = 2，打印 {b}。包含 a，進入下一個層次處理 index = 1，此時 current = "a"：
不包含 b，到達 index = 2，打印 {a}。包含 b，到達 index = 2，打印 {a, b}。

powerset: S={{},{b},{a},{a, b}}