資料結構報告

李柏勳

July 30, 2024

CONTENTS

CHAPTER 1 解題說明

(一)

1. 以遞迴實作計算阿克曼函式，已知條件式如下：

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

實作參見檔案 ack.cpp，其遞迴函式：

```
5   int ack(int m,int n){        //宣告ack函數並設定條件式
6       while(true){
7           if(m == 0){
8           return n+1;          //m=0, 回傳n+1
9       }
10      else if(m>0 && n==0) {
11          return ack(m-1,1);   //m>0且n=0, 回傳(m-1,1)
12      }
13      else if(m>0 && n>0){
14          return ack(m-1,ack(m,n-1));     //m>0且n>0, 回傳(m-1,ack(m,n-1)
15      }
16  }
17  }
```

2.以非遞迴實作計算阿克曼函式，已知條件式如下：

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

實作參見檔案 ack-nonrecursive.cpp，其遞迴函式：

```cpp
unsigned int Ackerman(unsigned int m, unsigned int n) {
    stack<pair<unsigned int, unsigned 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<unsigned int, unsigned int> top = s.top();         //
        m = top.first;
        n = top.second;
        s.pop();                 //彈出m

        if (m == 0) {
            if (!s.empty()) {
                s.top().second = n + 1;          //m=0,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, 則(m-1,1)
        } else {                                 //m!=0, n!=0,
            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)

以遞迴實作計算集合內的組合，已知計算公式如下：

n 個元素的組合共有：2^n 種組合

實作參見檔案 powerset.cpp，其遞迴函式：

```
7   int powerset(int num){      //宣告powerset函數並設定條件式
8       if (num == 0){
9           return 1;           //元素為0時只有空集合一個，因此回傳值為1
10      }
11      else return 2 * powerset(num-1);    //當n!=0，回傳(2 * (2 * num - 1))
12  }
```

CHAPTER 2 演算法設計與實作

(一)

1.

```
19  int main() {
20
21      int m,n;                    //宣告變數以利後續作輸入使用
22
23
24      while (true) {
25          cin>>m>>n;              //輸入數值
26          cout << ack(m,n) << "\n";     //呼叫ack函數計算並輸出
27      }
28      return 0;
29
30  }
```

2.

```
34  int main() {
35      int m, n;                                   //宣告變數以利後續作輸入使用
36      while(true){
37          cin >> m >> n;                          //輸入數值
38
39          cout << Ackerman(m, n) << endl;         //呼叫Ackerman函數計算並輸出
40      }
41
42      return 0;
43  }
```

(二)

```
14  int main() {
15      int num;
16      while(true){
17          cin>>num;           //輸入有幾個元素
18          cout<<powerset(num)<<"\n";      //呼叫powerset函數計算並輸出
19      }
20      return 0;
21  }
```

CHAPTER 3 效能分析

(一)

1.

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

$$S(P) = 2 \times n$$

2 個變數、n 次遞迴。

CHAPTER 4 測試與過程

(一)

1.

```
PS C:\Users\user\Desktop\school\c++> .\ack.exe
1 1
3
2 1
5
3 2
29
```

2.

```
PS C:\Users\user\Desktop\school\c++> .\ack-nonrecursive.exe
1 1
3
2 1
5
3 2
29
```

(二)

```
PS C:\Users\user\Desktop\school\c++> .\powerset.exe
0 1 2 3
1
2
4
8
4 5 6
16
32
64
```

驗證

(一)

1. 此函式遞迴若欲求得(1,1)，則呼叫 ack 函式，進入函式後，首先 m > 0 且 n > 0 所以回傳(m-1, (m, n-1))，即(0, (1, 0))，接著計算(1, 0)，m >0 且 n = 0，所以回傳(m-1, 1)，即(0, 1)，再來 m = 0, 回傳 n + 1，則(0, 1) = 2，回到(0, (1, 0))，已知(1, 0) = 2，(0, 2) = 3。ack (1, 1) = 3。

2. 此函式非遞迴若欲求得(1,1)，則呼叫 ack 函式，進入函式後，先創造一個堆疊並將 m 跟 n 放入其中，接著寫迴圈判斷如果堆疊內不為空值則運行，將 m 設置為第一個元素並彈出，先判斷 m 是否為 0，不為 0 則判斷 n 是否為 0，兩者皆不為 0 則將(m-1,(m,n-1))拆為(m-1,0)及(m,n-1)，個別做運算，前者(0, 0)輸出為 1，後者(1, 0)則再帶入到 m > 0, n = 0，則(m-1,1) = (0, 1)，接著 m = 0, n > 0 則輸出 n + 1，因此(1, 0) = 2，兩者輸出 1 + 2 = 3。ack-nonrecursive (1, 1) = 3。

(二)

此函式遞迴若欲求得 3 個元素的集合共有幾種組合，則呼叫 powerset(3)，進入函式後，首先 2 * powerset(num-1) = 2 * powerset(2)，powerset(2) = 2 * powerset(1)，powerset(1) = 2 * powerset(0)，powerset(0)回傳為 1，因此 powerset(1) = 2，powerset(2) = 4，powerset(3) = 8。