

資料結構報告

資工二甲 41243101 伍翊瑄

日期: 2024/10/23

目錄

1 .	解題說明	3
2 .	演算法設計與實作	4
3 .	效能分析	5
4 .	測試與過程	6

一. 解題說明

使用遞迴配合字串

自訂一 `countpowerset` 遞迴函數，計算幂集。

需要參數 `S`、`size`、`index` 與 `currentSubset` 和 `currentSubsetSize`，字串 `currentSubset[]` 用來紀錄集合 `S` 可能的子集組合。

- 字串 `S` 表示 `n` 個元素的集合
- `size` 為字串 `S` 的大小，表示輸入給字串 `S` 的元素個數 `n`
- `Index` 是一整數索引值，紀錄目前處理到的元素位置
- `currentSubset`，儲存集合 `S` 可能的子集組合
- `currentSubsetSize`，表示集合 `S` 可能的子集組合的元素個數

實作參見檔案 `HW2.cpp`，其遞迴函式：

```
17 // 計算幂集的遞迴函數
18 void countpowerset(string S[], int size, int index, string currentSubset[], int currentSubsetSize) //index是目前處理到的元素位置
19 {
20     // 當 startindex 到達集合 S 的結尾，輸出當前子集
21     if (index == size)
22     {
23         printSubset(currentSubset, currentSubsetSize);
24         return;
25     }
26     // 「不包含」當前元素，直接遞迴進到下一層，處理下一個元素
27     countpowerset(S, size, index + 1, currentSubset, currentSubsetSize);
28     // 「包含」當前元素，生成包含當前元素的子集並處理下一個元素
29     currentSubset[currentSubsetSize] = S[index];
30     countpowerset(S, size, index + 1, currentSubset, currentSubsetSize + 1);
31 }
32
33
34
35
```

二. 演算法設計與實作

HW2.cpp 的 main.cpp

```
36 int main() {
37     int n;
38     cout << "請輸入集合的元素個數 (最多" << MAX_SIZE <<"個): ";
39     cin >> n;
40
41     if (n < 0 || n > MAX_SIZE)
42     {
43         cout << "元素個數必須在 0 到 " << MAX_SIZE << " 之間!" << endl; //包含0
44         return 0;
45     }
46
47     string S[MAX_SIZE]; // 儲存輸入元素之集合
48     string currentSubset[MAX_SIZE]; // 儲存子集
49     cout << "請輸入集合的元素: ";
50     for (int i = 0; i < n; i++)
51     {
52         cin >> S[i];
53     }
54
55     cout << "冪集powerset(S)=" << endl;
56     countpowerset(S, n, 0, currentSubset, 0);
57     return 0;
58 }
```

三. 效能分析

時間複雜度

$T(P) = O(2^n)$ ，冪集的組合數量指數增長

在計算冪集的過程中，每個元素都會被考慮兩次（一次包含，一次不包含），所以對於 n 個元素，將會有 2^n 種可能的子集。

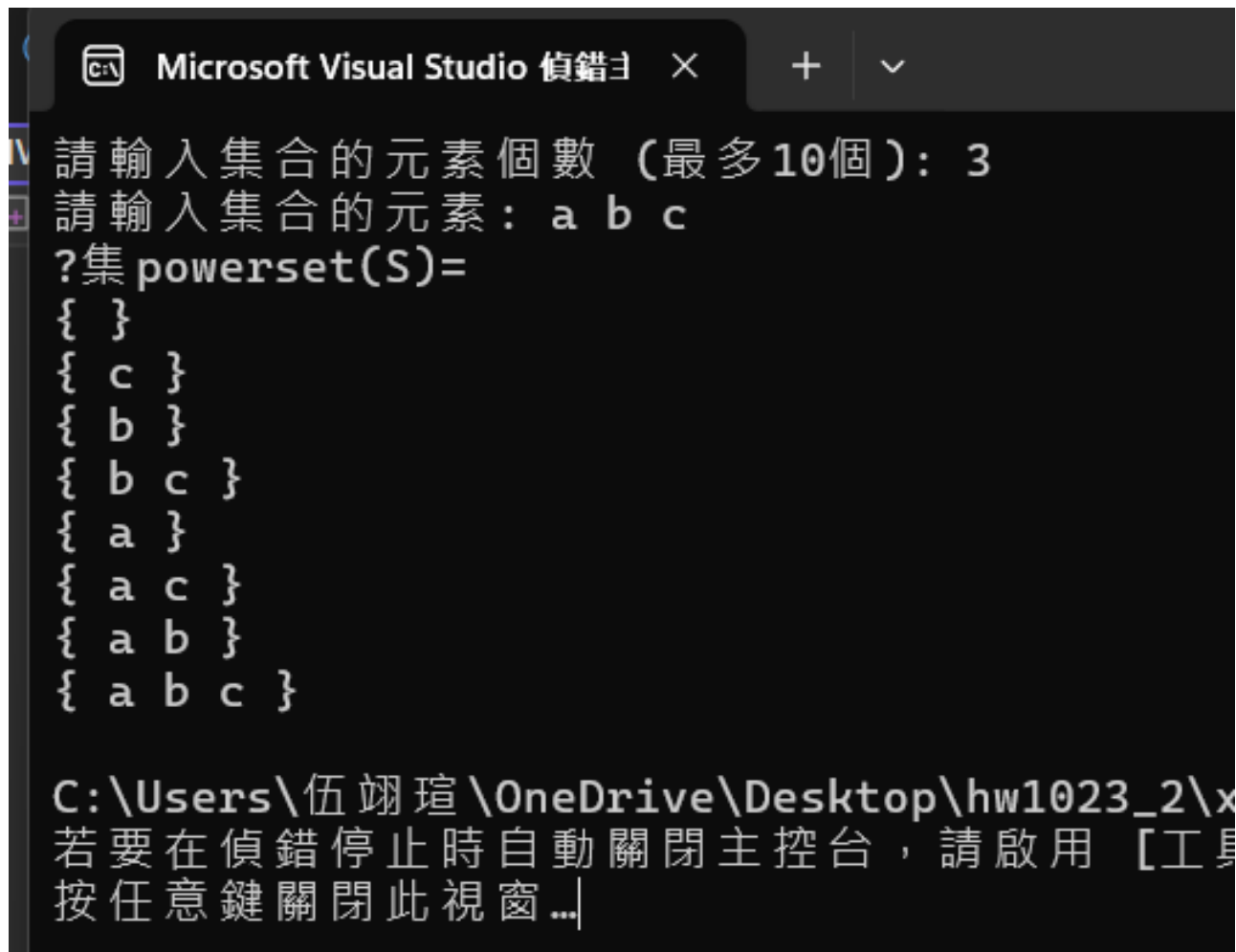
最終的時間複雜度是 $O(2^n)$ ，而不會是 $O(2^n * 2^n)$ 。這是因為每個遞迴調用都不是獨立的，而是建立在前一層的基礎上。每個元素的包含與否決定了樹狀結構的不同分支，但最終所有分支都只會展開為 2^n 個子集。

空間複雜度

$S(P) = O(n)$ ，與遞迴的深度相關

在最壞情況下，遞迴的最大深度是 n ，因為當我們遞迴進到最深層時，我們處理的元素個數是 n 。

四. 測試與過程



```
Microsoft Visual Studio 偵錯器 × + ∨  
請輸入集合的元素個數 (最多10個): 3  
請輸入集合的元素: a b c  
?集 powerSet(S)=  
{ }  
{ c }  
{ b }  
{ b c }  
{ a }  
{ a c }  
{ a b }  
{ a b c }  
  
C:\Users\伍翊瑄\OneDrive\Desktop\hw1023_2\x  
若要在偵錯停止時自動關閉主控台，請啟用 [工具  
按任意鍵關閉此視窗...]
```

驗證

初始呼叫：countpowerset (S, 3, 0, {}, 0)

當前元素 a, index = 0

選擇遞迴不包含 a：

呼叫 countpowerset (S, 3, 1, {}, 0)

當前元素 b, index = 1

選擇不包含 b：

呼叫 computePowerset(S, 3, 2, {}, 0)

當前元素 c, index = 2

選擇不包含 c：

呼叫 computePowerset(S, 3, 3, {}, 0) -> 輸出：{ }

選擇包含 c：

呼叫 computePowerset(S, 3, 3, {c}, 1) -> 輸出：{ c }

選擇包含 b：

呼叫 computePowerset(S, 3, 2, {b}, 1)

當前元素 c, index = 2

選擇不包含 c :

呼叫 computePowerset(S, 3, 3, {b}, 1) -> 輸出 : { b }

選擇包含 c :

呼叫 computePowerset(S, 3, 3, {b, c}, 2) -> 輸出 : { b
c }

選擇包含 a :

呼叫 computePowerset(S, 3, 1, {a}, 1)

當前元素 b, index = 1

選擇不包含 b :

呼叫 computePowerset(S, 3, 2, {a}, 1)

當前元素 c, index = 2

選擇不包含 c :

呼叫 computePowerset(S, 3, 3, {a}, 1) -> 輸出 : { a }

選擇包含 c :

呼叫 computePowerset(S, 3, 3, {a, c}, 2) -> 輸出 : { a
c }

選擇包含 b :

呼叫 computePowerset(S, 3, 2, {a, b}, 2)

當前元素 c, index = 2

選擇不包含 c :

呼叫 `computePowerset(S, 3, 3, {a, b}, 2)` -> 輸出 : { a
b }

選擇包含 c :

呼叫 `computePowerset(S, 3, 3, {a, b, c}, 3)` -> 輸出 :
{ a b c }

這樣的流程會遍歷所有可能的子集，並在最底層的遞迴中將子集輸出。