資料結構報告範例

王語晨

OCT 31, 2024

CONTENTS

1 解題說明 **2**

2 演算法設計與實作 **3**

3 效能分析 **4**

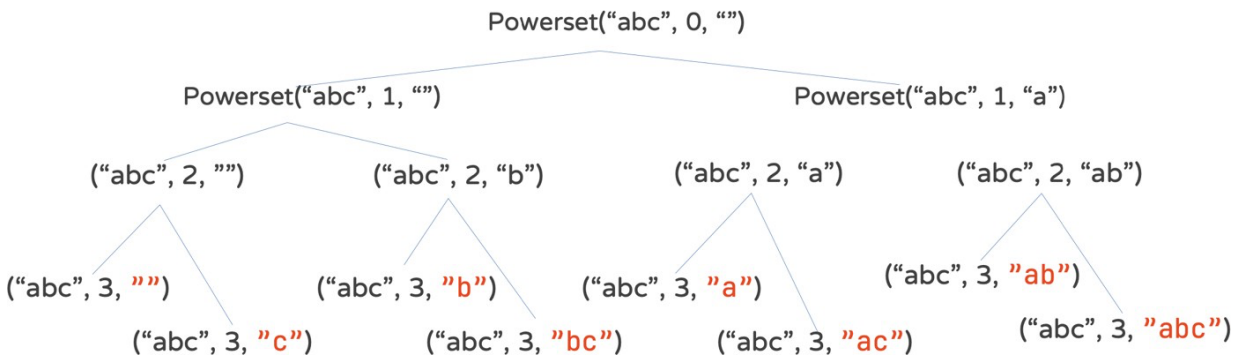
4 測試與過程 **5**

5 申論與心得 **6**

CHAPTER 1

解題說明

以powerset函數實作，已知powerset函數計算公式如下：



實作參見檔案 powerset.cpp，其powerset函式：

```
int powerset( string a, int b, string c)
      { if (b == a.size()) {
            cout << c << " ";
            return 0;
      }
      powerset(a, b + 1, c);
      powerset(a, b + 1, c+a[ b ]);
}
```

Figure 1.1: powerset.cpp

CHAPTER 2

演算法設計與實作

```
int main() {
	string a, c;
	a = "abc";
	c = "";
```

Figure 2.1: main.cpp

CHAPTER 3

效能分析

$$f(n) = O(n)$$

時間複雜度

$$O(2^n)$$

Powerset函式使用遞迴，並且在每一次呼叫時會進行兩次遞迴；一次是跳過當前的字元(b+1)，一次是將當前字元添加到結果集合中（b+1,c+a[b]）。這會導致每個字元有兩種型態：取或不取。對於一個長度為n的字串，遞迴的數量將會是2^n，因此時間複雜度為O(2^n)。

空間複雜度

$$S(P) = O(n*2^n)$$

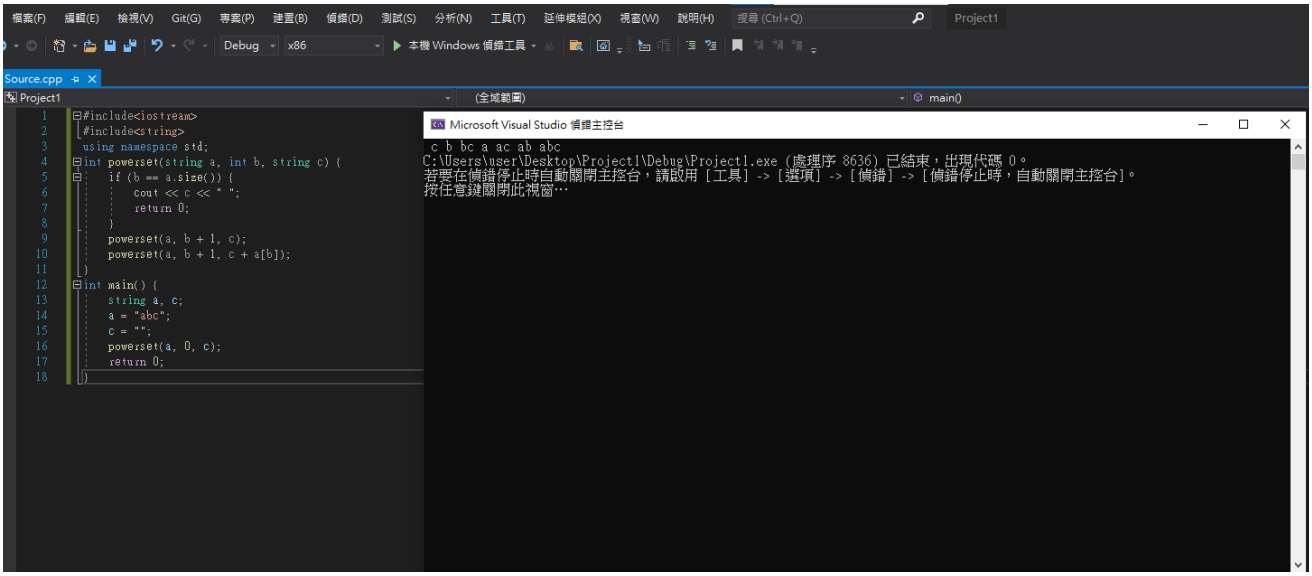
遞迴的深度最多會是n，而每次呼叫會創建一個新的結果字串（即c+a[b]）。由於最多會有2^n個結果集合，因此額外的空間複雜度與這些結果集合成正比。結果集合長度最多為n，因此空間複雜度為O(n*2^n)。

CHAPTER 4

測試與過程

```
$ g++ main.cpp -o main.exe && ./main.exe"
 " " "c" "b" "bc" " a" " ac" " ab"
" abc"
```

Figure 4.1: shell command



驗證

powerset 函數接受三個參數：
a：輸入的字串（“abc”）、b：索引（跟蹤字串中當前處理的位置）、c：結果變量（用於累積當前子集）。然而該函數在每一個位置上有兩個選擇：
1. 跳過當前字元，直接遞迴到下一個索引位置、2.選擇當前字元並將它加入當前子集（即 c + a[b]），然後遞迴到下一個索引位置。而終止條件為if (b == a.size())，這是遞迴的終止條件。當 b 等於字串長度時，表示已經處理完所有的字元，這時將當前的子集 c 輸出並返回。

CHAPTER 5

申論與心得

申論

這題要求撰寫一個遞迴函數來計算集合powerset。

Powerset 是一個集合的所有子集的集合，程式使用遞迴來達成這個目標，並透過掃描每個字元的所有可能組合來生成子集。

函式powerset他接受三個參數：

字串a:代表要生成子級的來源字串

整數b:代表當前字元的索引

字串c:代表當前已經生成的子集

當b遞增到等於a.size( )時，代表字串的每個字符都已經被考慮過了，將當前的子集輸出。

而遞迴分為兩種情況：

不包含當前字符:powerset(a, b+1, c)

包含當前字符:powerset(a, b+1, c+a[b])

心得

透過這份作業，更加了解powerset的操作，展示了遞迴的應用，在保持程式碼簡潔的前提下，考慮「包含」與「不包含」的情況，而這樣也避免了多層次的迴圈或複雜的判斷條件。