

## 一、實作演算法介紹

### KMP：

#### 1. 讀檔：

先讀第一個檔，使用getline將第一個檔裡面所有的pattern存到陣列s1裡面，並呼叫FailureFunction函數把每個pattern的failure function算出來。

#### 2. 字串比對：

分別使用while, for迴圈去比對每個字串，外圈是text，內圈是pattern，因此程式執行的順序是先將一行text裡面所有包含pattern的字串找出來，再找下一行text裡面所有的pattern。

#### 3. FailureFunction的實作：

FailureFunction是用來計算出pattern的failure function的函數，利用failure function可以找出pattern裡面相同的子字串，例如， $f(k)$  表示由  $i$  往前  $f(k)+1$  個字和字串開頭的  $f(k)+1$  個字相同。

首先計算出pattern的長度，然後設定failure function的初始值為-1，在for (int  $j = 1$ ;  $j < \text{lengthP}$ ;  $j++$ )的條件下，如果pattern的第 $j$ 個字元跟第 $i + 1$ 個字元( $i$ 為第 $j-1$ 個字元的失敗函數)不同且 $i \geq 0$ ，那 $i$ 等於 $i$ 的失敗函數。如果pattern的第 $j$ 個字元跟第 $i + 1$ 個字元相同，那第 $j$ 個字元的失敗函數就是前一個字元的失敗函數再+1，反之如果不同，第 $j$ 個字元的失敗函數就是-1。

#### 4. KMP的實作：

KMP在進行字串比對的途中，每次要移動 pattern 時，就先取出 pattern 中有比對到相同的前綴部分，然後利用該pattern的failure function求其「次長的相同前後綴」，然後以此大幅移動pattern，而不必一次移動一個位置。

posP是pattern正在比較的字元的位置，posT是text正在比較的字元的位置。首先使用while迴圈，在posP不超過pattern的長度、posT不超過text的長度的情況下，如果pattern的第posP個字元跟text的第posT個字元相同，那posP和posT都+1，繼續和下個字元比較。

如果在 $\text{posP} == 0$ 的情況下比對的字元不同，那 $\text{posT}+1$ ，繼續跟 $\text{text}$ 的下個位置比較。如果在 $\text{posP} != 0$ 的情況下比對的字元不同，那 $\text{posP}$ 移動到 $\text{posP}$ 前一個字元的失敗函數的下個位置繼續比較。

跳出while迴圈有兩種可能，一種可能是因為 $\text{posP} > \text{lengthP}$ ，表示找到相同字串，因此回傳 $\text{posT} - \text{lengthP}$ ，也就是 $\text{text}$ 與 $\text{pattern}$ 相同字串的起始位置；另一種可能是因為 $\text{posT} > \text{lengthT}$ ，表示沒找到相同字串，因此回傳-1表示沒有相同字串。

## 5. KMP的應用：

先呼叫一次KMP函數並把回傳值存到 $\text{lastResult}$ ，如果 $\text{lastResult}$ 不是-1，表示找到相同字串，就輸出結果。為了知道後面是否能找到相同字串，再呼叫一次KMP函數，並把KMP函數的第二個參數改成 $\text{s2} + \text{lastResult} + 1$ ，如此才能從一開始找到字串的後面一個位置繼續找。如此一直重複呼叫KMP函數直到KMP函數回傳-1，也就是找不到為止。

## 6. 全部的pattern對每行的text都做KMP，以找出測資裡所有比對相符的字串。

## 7. 利用 $\text{clock}()$ 函數印出程式執行時間。

## 二、測試資料與結果

pat - 記事本	text - 記事本
<pre>1 6 "mile" "you smile, I smile and everyone smiles" 1 15 "mile" "you smile, I smile and everyone smiles" 1 34 "mile" "you smile, I smile and everyone smiles" 2 1 "Xx" "xxxxx" 2 2 "Xx" "xxxxx" 2 3 "Xx" "xxxxx" 2 4 "Xx" "xxxxx" 3 3 "Xx" "ooxx" 3 1 "oo" "ooxx" 4 2 "123a b" "3123a bpp 12 3ab"  15 ms  ----- Process exited after 0.02021 seconds with return value 0 請按任意鍵繼續 . . .</pre>	<pre>you smile, I smile and everyone smiles xxxxx ooxx 3123a bpp 12 3ab Qq13fB</pre>

第四行測資是範例三的測資

[illegible][illegible]

我自行編造的測資，為了測試KMP和Brute-Force的速度差而編造的長測資。

三、比較KMP和Brute-Force的執行時間

<div>C:\Users\呂翊愷\Desktop\資料結構\project1\0416025_KMP.exe</div> <div>1 6 "mile" "you smile, I smile and everyone smiles" 1 15 "mile" "you smile, I smile and everyone smiles" 1 34 "mile" "you smile, I smile and everyone smiles" 2 1 "Xx" "xxxxx" 2 2 "Xx" "xxxxx" 2 3 "Xx" "xxxxx" 2 4 "Xx" "xxxxx" 3 3 "Xx" "ooxx" 3 1 "oo" "ooxx" 4 2 "123a b" "3123a bpp 12 3ab"  15 ms  ----- Process exited after 0.02021 seconds with return value 0 請按任意鍵繼續 . . .  微軟注音 半 :</div>	<div>C:\Users\呂翊愷\Desktop\資料結構\project1\0416025_BF.exe</div> <div>1 6 "mile" "you smile, I smile and everyone smiles" 1 15 "mile" "you smile, I smile and everyone smiles" 1 34 "mile" "you smile, I smile and everyone smiles" 2 1 "Xx" "xxxxx" 2 2 "Xx" "xxxxx" 2 3 "Xx" "xxxxx" 2 4 "Xx" "xxxxx" 3 3 "Xx" "ooxx" 3 1 "oo" "ooxx" 4 2 "123a b" "3123a bpp 12 3ab"  31 ms  ----- Process exited after 0.027 seconds with return value 0 請按任意鍵繼續 . . .  微軟注音 半 :</div>
---	--

用較短的測資比較，KMP的執行時間比Brute-Force快了快一半

<div>C:\Users\呂翊愷\Desktop\資料結構\project1\0416025_KMP.exe</div> <div>aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaa aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaa bbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba aaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbb aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba 218 1855 "aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbb aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaa bbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba aaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbb aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaa bbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba aaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbb aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaa bbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba aaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbb aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaa aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba 9080 ms  ----- Process exited after 9.102 seconds with return value 0 請按任意鍵繼續 . . . 微軟注音 半 :</div>	<div>C:\Users\呂翊愷\Desktop\資料結構\project1\0416025_BF.exe</div> <div>aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaa aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaa bbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba aaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbb aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba 218 1855 "aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaa bbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba aaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbb aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaa bbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba aaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbb aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaa aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaa bbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba aaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbb aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaaaaaaa aaaaaaaaabbbbbaaaaaaaaaaaaaaaaaabbbbba 9997 ms  ----- Process exited after 10.01 seconds with return value 0 請按任意鍵繼續 . . . 微軟注音 半 :</div>
---	--

用較長的測資比較，KMP的執行時間僅是Brute-Force的將近九成

Brute-Force的時間複雜度是 $O(\text{lengthP} * \text{lengthT})$ ，它要連續檢查text的每個字元是否是pattern的第一個字元，如果是才能進而比較下一個字元。

KMP的時間複雜度是 $O(\text{lengthP} + \text{lengthT})$ ，利用failure function可以得知相同子字串的位置，因此可以一次大幅移動，而不必像Brute-Force一格一格移動，省去不少時間。