

# KMP Algorithm ON Intel X86

## 研究動機

經過一整個學期組語的薰陶，和四個作業的摧殘.....(誤)，我們多少體驗到了組語的神秘與奧妙。不過我們一學期學了三個語言，不免感到對各個語言有點不熟悉....因此秉著追根究底的研究精神(XD)，我們決定用 intel 語言來 implement 一個演算法，藉此讓我們更加了解 intel 語言。

## KMP 演算法

最簡易的子字串搜尋法，是對於每一個字元開頭  $S_i$ ，判斷字串  $S(i)S(i+1)...S(i+n-1)$ ，是否與欲搜尋之字串  $P(0)P(1)...P(n-1)$  相同，然而，這樣的搜尋法複雜度為  $O(n * m)$ ，當資料量極大時，將耗費相當多的時間

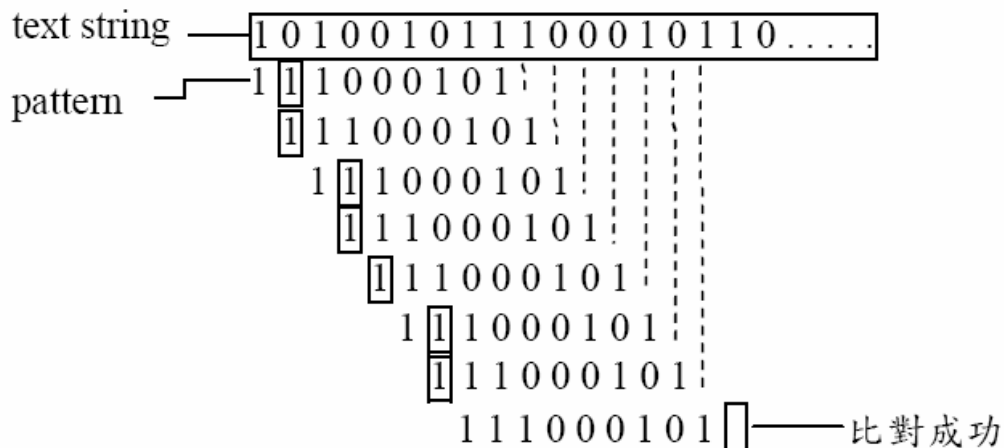


圖 4-2 暴力式演算法

Knuth 演算法如下：

對於每次的比對，若比對成功，則原字串及子字串的 **index** 各加 1，繼續比對。若比對失敗，則視情況決定位移：

(1) 若子字串的 **index** 為 0，代表開頭便不符，因此只需把原字串的 **index** 加 1 繼續比對。

(2) 若在其他位置比對失敗，則根據位移表決定子字串的 **index** 移至何處繼續比對。位移概念如下：

當原字串在位置  $i$  和子字串在位置  $j$  比對失敗，代表  $P(0) \sim P(j-1)$  與  $S(i-j) \sim$

$S(i - 1)$  的比對皆成功。

若字串  $P(j - n) \dots P(j - 1)$  與  $P(0) \dots P(n - 1)$  相同 ( $n$  為可行值之最大值)，再根據上述條件，可知字串  $P(j - n) \dots P(j - 1)$  與  $S(i - n) \dots S(i - 1)$  相同，因此可知  $P(0) \sim P(n - 1)$  與  $S(i - n) \sim S(i - 1)$  的比對皆會成功。

既然如此，只需從  $P(n)$  與  $S(i)$  比對起便可 (因為確保  $P(0) \sim P(n - 1)$  的比對一定成功)。此演算法之時間複雜度為  $O(m+n)$

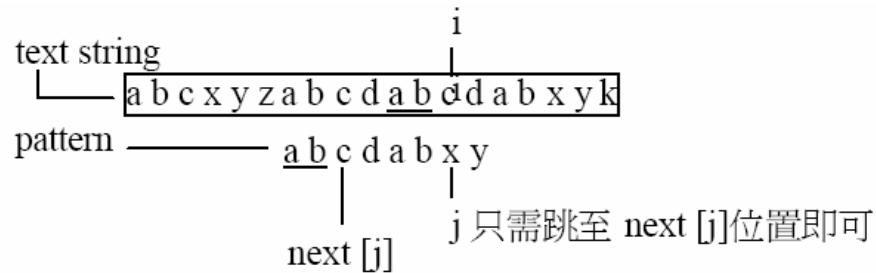


圖 4-3 K. M. P 演算法圖 1

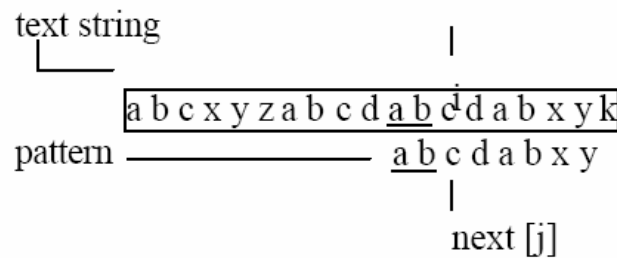


圖 4-4 K. M. P 演算法圖 2

## 開始作業

一開始我們先將 C code 寫出來。在 DP 建表的部份遇到了些許的困難：

```
void prekmp(char*x,int m,int kmpnext[])
{
    int i=0,j=kmpnext[0]=-1;
    while(i<m)
    {
        while(j>-1 && x[i]!=x[j])
            j=kmpnext[j];
        i++;
        j++;
        if(x[i]==x[j])

```

```

        kmpnext[i]=kmpnext[j];
        else
        kmpnext[i]=j;      €
    }
    for(i=0;i<m;i++)printf("%d",kmpnext[i]);
    system("pause");
}
(節錄)

```

在好不容易搞懂它並寫出來後。我們開始正式動工，將它翻成 intel 版本。在 coding 過程中，我們才驚覺 intel 其實沒有我們想像中那麼低階，在作業四裡許多沒有遇到的問題，在這次的 project 中都浮了出來。在不斷的找資料、查書、向人求助下.....我們對 intel 語言有了更多的了解。並初步的寫出了原始版的 code: (節錄)

```

push esi          ;將 esi 存好，把 esi 當 temp 用
movsx esi,al
    add  edi,esi
    mov  ch,[edi]  ;ch=x[i]
    sub  edi,esi
    movsx esi,bl
    add  edi,esi
    mov  dl,[edi]  ;dl=x[j]
    sub  edi,esi
    pop  esi       ;pop esi
    cmp  ch,dl     ;if(x[i]==x[j])

```

接著，我們開始修改原始版的 code。我們赫然發現，以上的這段 code 可以修改成：

```

    mov  ch,[edi+eax]  ;ch=x[i]
    mov  cl,[edi+edx]  ;cl=x[j]
    cmp  ch,cl        ;if(x[i]==x[j])

```

如此的精簡!!我們才理解到原來 intel asm 比我們想像中聰明許多，只要真的花時間去了解它，他其實跟 c 語言的差距沒有我們想像中那麼大。

## 使用方式

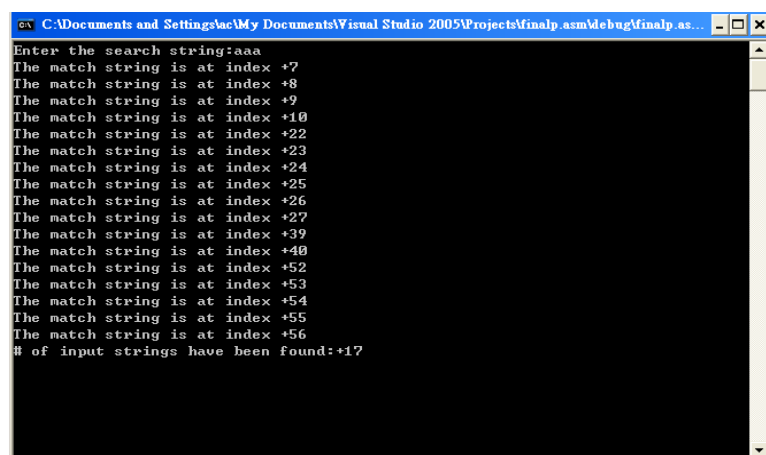
在與執行檔相同的路徑下有一個 myInput.txt，使用者需將要被 search 的字串 (文章) 放在裡面。

接著執行程式：



```
C:\Documents and Settings\ac\My Documents\Visual Studio 2005\Projects\finalp_asm\debug\finalp.as...
Enter the search string:
```

輸入想 search 的字串



```
C:\Documents and Settings\ac\My Documents\Visual Studio 2005\Projects\finalp_asm\debug\finalp.as...
Enter the search string:aaa
The match string is at index +7
The match string is at index +8
The match string is at index +9
The match string is at index +10
The match string is at index +22
The match string is at index +23
The match string is at index +24
The match string is at index +25
The match string is at index +26
The match string is at index +27
The match string is at index +39
The match string is at index +40
The match string is at index +52
The match string is at index +53
The match string is at index +54
The match string is at index +55
The match string is at index +56
# of input strings have been found:+17
```

將跑出一共有幾個 match 的字串，和所有 match 字串在原文字的起始位置。

## 感想

老實說，寫完了作業四之後，我們對 intel 這個語言還是矇矇懂懂...，覺的好像沒有真正了解這個語言。而在這次的 project 中，我們真的遇到了許多老師上課有講，但作業沒寫到的東西。原來，asm 可以開 array、可以寫 for loop、可以作 function call.....，原來，它遠比我們一開始想像的還要高階許多。作完這次的 project，我們學到了很多東西，也順便復習與實鑒了演算法課所學到的 DP，真的可說是獲益良多。

## 參考資料

- 1.逢甲資工硬體防火牆專題報告 ppt
- 2.<http://www-igm.univ-mlv.fr/~lecroq/string/node8.html>
- 3.<http://zh.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt%E7%AE%97%E6%B3%95>
4. <http://www.csie.ntu.edu.tw/~wcchen/algorithm/strMatching/Doc2.html>