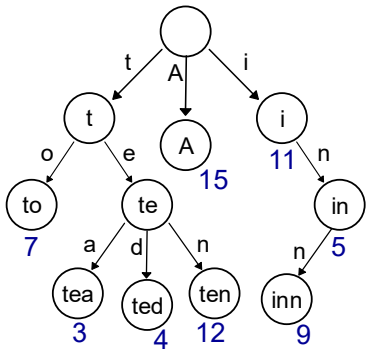
文字過濾器

1. 新增關鍵詞與過濾關鍵詞
   - 新增關鍵詞：將需新增的關鍵詞輸入至檔案中 (file1)
   - 需被過濾之文本：將需被過濾之文本獨立為另外一個檔案 (file2)
   - Usage: filter file1 file2
   - Result

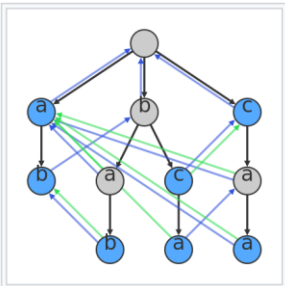
```
$ ./filter.exe test.in inputFile.html
It took 0.124 seconds.
```

   過濾結果以名稱 file2_filtered 放在同一目錄下，並顯示所花費時間。

2. 原理
   - Aho–Corasick algorithm
     (https://en.wikipedia.org/wiki/Aho%E2%80%93Corasick_algorithm)
     1. 建立 Trie



     (https://en.wikipedia.org/wiki/Trie)
     2. 利用 breadth-first search 根據最長後綴建立 suffix link
     (The following figure illustrates how to build the suffix links)



A visualization of the trie for the dictionary on the right. Suffix links are in blue; dictionary suffix links in green. Nodes corresponding to dictionary entries are highlighted in blue.

Dictionary {a, ab, bab, bc, bca, c, caa}

Path	In dictionary	Suffix link	Dict suffix link
()	–		
(a)	+	()	
(ab)	+	(b)	
(b)	–	()	
(ba)	–	(a)	(a)
(bab)	+	(ab)	(ab)
(bc)	+	(c)	(c)
(bca)	+	(ca)	(a)
(c)	+	()	
(ca)	–	(a)	(a)
(caa)	+	(a)	(a)

     3. 比對關鍵字時，若出現不匹配的狀況，則根據 suffix link 移動至適當的節點，再進行比對。
   - 因為避開了一定不匹配的冗餘比較，time complexity ~ O(n)，其中 n 為被過濾的

文本大小。

- 若遇關鍵字，則根據 UTF-8 格式計算詞長，並替換成同樣長度的'x'

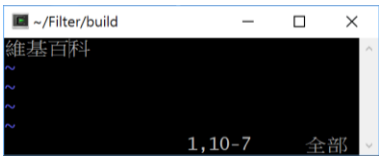
Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

(https://en.wikipedia.org/wiki/UTF-8)

3. 測試功能

- 以關鍵字”維基百科”為例

file1



原始檔案

過濾結果

- **執行時間**

n	m	Elapsed time (s)
195000	1	0.094
195000	100	0.11
195000	1000	0.109
195000	10000	0.171
195000	47000	0.312

4. **可進一步優化方向**
   - **Trie 的結構: double-array trie, 降低存儲 pointer 的 overhead**
     Jun-Ichi Aoe, Katsushi Morimoto, and Takashi Sato, "An efficient implementation of trie structures," *Software - Practice and Experience*, VOL. 22(9), 695–721, John Wiley & Sons, Ltd, 1992.