**How you implement your code**

先利用一張含有 文字 的圖片

自動產生的描述exists()，確認資料夾中的txt檔案有幾個。並把所有檔案中的title存起來，同時把所有內文parse和 split後insert進兩個Trie裡。其中一個Trie正著存，另一個則先把字串reverse後再insert (用來suff-search)。

之後打開query.txt，每次讀入一行並把split的結果建成一個expression tree。經過expression tree把各個node的答案intersection或是union，就可以把得到答案並輸出到output裡。

以下是我trie的建造方法。Who是用來表示是從哪個檔案insert的單字，如果他有經過那個node，就把他insert到那個node的whose中，這樣就能知道這個檔案中有沒有這串字。如果是這個單字的結尾，則額外insert “who”到 flag裡，表示在這個文檔中這個詞已經到結尾了。這樣exact-search才能知道這個詞是不是跟要找的詞完全一樣。

一張含有 文字 的圖片

自動產生的描述一張含有 文字, 螢幕, 螢幕擷取畫面 的圖片

自動產生的描述

Pre-search跟suff-search很像，都是檢測完要的字串後，返回所有那個節點中的whose，就能知道哪些data中有內文經過這個節點。只是suffix-search要用反著存的trie和反著的關鍵字來搜索。

**一張含有 文字 的圖片

自動產生的描述一張含有 文字 的圖片

自動產生的描述**

而**一張含有 文字 的圖片

自動產生的描述**exact-search則是返回最後節點的flag，才能保證這是這個單字的最後一個字母。

最後計算結果的部分，則是用query\_tree和recursion。將左右sub\_tree的值回傳後，選擇將兩邊結果並集或是交集並回傳。

**Challenges you encounter in this project**

1. 時間跑太久。

原本使用的方法是把每個data都建造兩個的trie，並在查完後直接刪掉，沒有把trie存起來，這樣發現跑一千筆資料需要120秒。後來嘗試把跑過的trie存起來而不delete掉，這樣還是需要四到五秒時間。所以最後我嘗試將所有data合起來存成兩個trie，所需時間才大幅下降。

1. Suff-search不會做。

本來不知道suff-search怎麼使用一個trie做出來，但後來上討論區看到很多人使用兩個trie反著insert才會做。

1. Compile問題。

用助教給的command compile一直報錯，最後只好借同學的電腦compile。

**References that give you the idea (github/paper…)**

<https://iter01.com/563709.html> 字典樹詳解