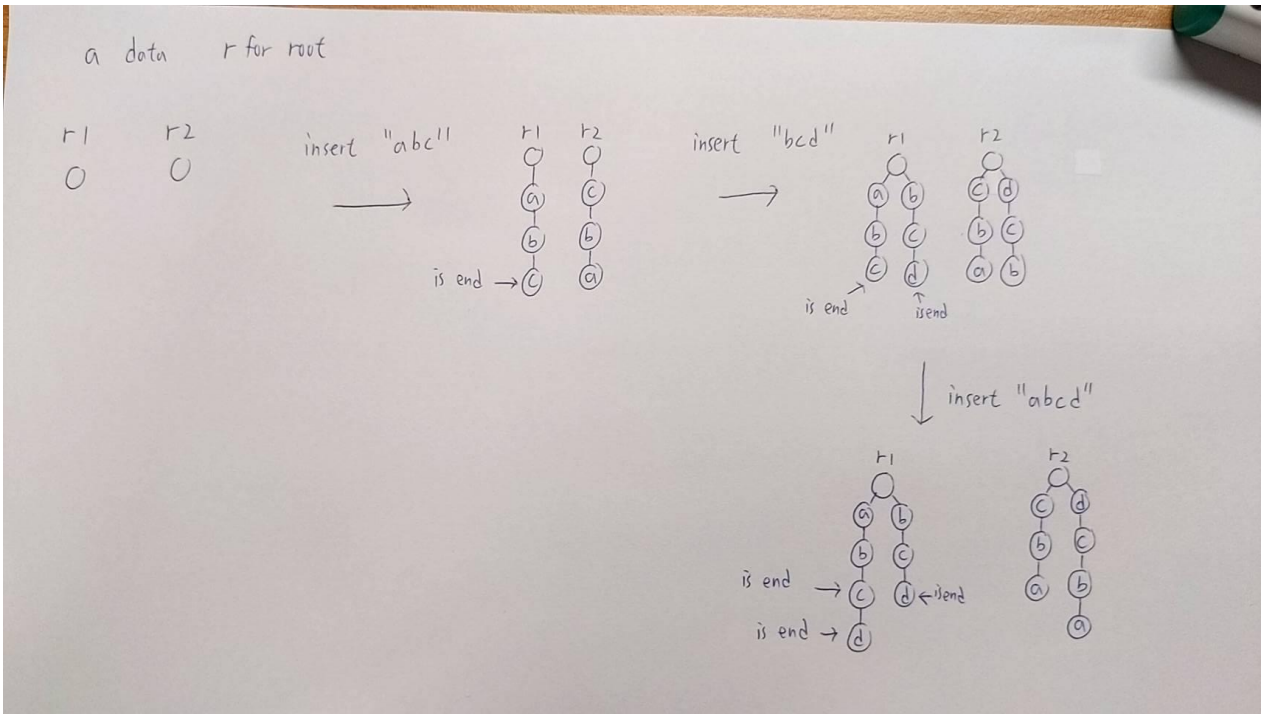
109062204 唐聖翔 Final project report

- How I implement my code:

首先要能夠讀多個檔案，讓 fi 一直開檔案，直到沒有成功打開就 break。再將每一個 data 建成兩個個 Tries。



r1 是存正向 insert 的 Trie，r2 存反向的 insert。

搜尋時分三種情況：

1. 找 exact 時 r1 若存在這條路且最後一個 node 的 isend 是 true 代表 data 內存在個字的 exact。
2. 找 prefix 時只需在 r1 要找到同一條路且最夠一個 node 是否是結尾不重要。
3. 找 suffix 時將要找的 string 反過來然後在 r2 找是否有反過來的路，只要存在就代表有這個 suffix。

每個 node 的 child 用一個大小為 26 的陣列儲存代，表 26 個字母，所以搜尋速度很快。

將同一排 query 分成許多 string，string 可能是字串或是+或/。若是字串則用第一個字元判斷是否為 exact 或是 suffix。每條 query 第基數個 string 一定是字，偶數則是+或/。並且是 left associative 所以先將第一個 string 做搜尋，在一次將這排 query 讀完與第一個字串做完搜尋的值運算更新，只要這顆 trie 符合就將這棵 trie 代表的 data 的 title 寫進 output。需要每個 data 都算完後才能進入下一排 query。若沒有一個 data 符合這排 query 則需要寫入 not found。

➢ Challenges I encounter in this project

1. 使用 **g++ -std=c++17 -o essay-search.exe ./*.cpp -lstdc++fs** 時沒辦法 compile 問同學後試了將 fs 刪掉就成功了。
2. 沒有使用過 fstream 和 ofstream 所以上網查了一下。
3. 一開始使用一顆 trie 來記錄每個 data，每個 data 內另有 26 個 list 將每個是字尾的同樣英文字母存在同一個 list，需要找 suffix 時將要搜尋的字反過來找到對應的 list 再依序往 parent 確認，但是計算後發現這樣雖然省空間到是較費時，所以改為最後的寫法。

➢ References that give you the idea

■ reference: https://www.geeksforgeeks.org/trie-insert-and-search