How I implement my code

這次 final project 要我們實作的是從上千篇的論文標題和大綱(abstract)當中進行關鍵字的 exact search, prefix search, 和 suffix search，搜索出哪幾篇符合 query 裡 expression 的條件，並把該論文標題依照論文 input 的順序 output 到 output file。

首先，因為檔名為從 0 開始的連續數字加上".txt"，所以我們可以簡單用迴圈的方式，讀到所有的檔案，如下圖：

```
//traverse all data files
ll cur_data_num=0;
while(1){
    string str_file_num;
    ll tmp_oper_num=cur_data_num;
    while(1){
        int last_bit=tmp_oper_num%10;
        str_file_num.push_back(last_bit+'0');
        tmp_oper_num/=10;
        if(tmp_oper_num==0) break;
    }
    reverse(str_file_num.begin(),str_file_num.end());
    string open_data_file=data_dir+str_file_num+".txt";
```

接下來，利用 open 與 getline 將檔案中的內容讀出來並建 trie。建 trie 時，使用 vector<vector<int>> trie 與 vector<bool> is_terminal 來記錄。trie 的每個 element 都為一個大小為 26 的 vector(從 a 到 z 共 26 個字母)，且每個 element 預設值為 0。在將字串放入 trie 時，將字元一個一個讀取，並從 index 為 0 的 root 開始拜訪對應(依照字母)的 children。若拜訪的值為 0 代表之前沒有加入過以這個字串為 prefix 的字串，如此，則 emplace_back 一個新的 node，並將其編號(每次加 1)。這樣下次 traverse 同一個字串時就會有先前儲存的紀錄。而到字串最後，若現在 node 的 index 為 x，則將 is_terminal[x]改為 1，代表這個 node 為某個字串的結尾。而之所以要多用一個 is_terminal 來記錄是因為在執行 exact search 時，拜訪 trie 的過程中我們必須要有一個可以判斷該 node 是否為某個字串的結尾的方法。Prefix search 時只要能把要查詢的字串一個字元一個字元地去拜訪完並都有找到(沒有某特定 trie[x][y]值==0 的情況發生)，就算是成功。至於 suffix search 的部分，我的作法是將所有論文標題和大綱內的每個字串取 reverse 並加進另一個 trie，名為 trie_rev，並依照 prefix search 的做法去執行。

```
// PARSE CONTENT
vector<string> content = word_parse(tmp_string);
for(string s:content){
    cur_node_idx=0;
    for(char c:s){
        if(c<'a') c+=32; //轉小寫
        if(trie[cur_node_idx][c-'a']==0) {
            trie[cur_node_idx][c-'a']=++cnt_node;
            trie.emplace_back(vector<int>(26));
            is_terminal.emplace_back(0);
        }
        cur_node_idx=trie[cur_node_idx][c-'a'];
    }
    is_terminal[cur_node_idx]=1;
}
```

在替一個 data file 建完 trie 後，就一次把所有 query 處理完。若 query 中第 i 個 expression 成功，則將 cur_data_num push 進 found[i]裡。在進行 query 時，若將 expression 反向的執行，則在某些情況可以不用把 query 做完就可以得知最後的 expression 是否成立：若 expression 中某個 query string 的左邊為 +(and)，且該項查詢在 trie 中查詢失敗，則代表整個 expression 最後的結果也會為 false；反之，若某個 query string 左邊為/(or)，若該項查詢成立，則代表整個 expression 為 true。這兩種情況都可以直接從 for 迴圈中 break。為了確認查詢是否成立，寫了 check 函數去判定，將 trie 與 trie_rev 與 is_terminal 當參數傳入，並依照 exact search, prefix search, 和 suffix search 三種不同的 search 進行處理。

```
//query
for(int i=0;i<query_cnt;i++){
    int ok=1;
    for(int j=query[i].size()-1;j>=0;j--){
        if(query[i][j].second==1 && check(query[i][j].first,trie,trie_rev,is_terminal)==0){
            ok=0;
            break;
        }
        else if(query[i][j].second==0 && check(query[i][j].first,trie,trie_rev,is_terminal)==1){
            ok=1;
            break;
        }
    }
    if(ok){
        found[i].push_back(cur_data_num);
        file_been_queried=1;
    }
}
```

每次 query 得到的結果要把符合條件的 file 依照順序輸出，而我是從 0.txt, 1.txt 依序往下查詢，所以可以直接從 vector<int> found[10010] 的第 0 個 index 以正常順序開始依序輸出，把符合條件的標題寫入 output.txt 中。

Challenges I encounter in this project

我遇到的問題是如何在終端執行我的 code。因為已經很久沒碰到 cmd，一開始還傻傻地想直接在 IDE 上執行，之後才想到上學期修計程二時最後一個 mini project 有使用到終端去執行 code，所以我翻了一下以前的檔案後才想起來如何解決這個問題：下載 mingw64，修改一下環境變數並在 cmd 裡初始化一些功能後，便可以在 cmd 執行檔案。而執行這次 project 的方式則是參考 ppt 上助教提供的指令。

References that give me the idea

Trie 參考資料：

https://zh.wikipedia.org/wiki/Trie

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

https://www.hackerearth.com/practice/data-structures/advanced-data-structures/trie-keyword-tree/tutorial/