

A. How I implement my code

這次 project 我是以 Trie 的 algorithm 來實作 essay search。我透過老師的 PPT 清楚瞭解了 Trie 的 algorithm，並參考網路上 Trie 的 insert 和 search 進行修改。

我將每一篇 essay 分別建成一個 prefix trie 和一個 suffix trie，並用 vector 將他們存起來。當遇到 query 型態為 exact 或 prefix 時，我會 search 每篇文章的 prefix trie；而遇到 query 型態為 suffix 時，則 search 每篇文章的 suffix trie。

另外，在處理 query 的部分，我會先一行一行將他 getline，再用助教寫好的 `vector<string> split` 將每行的 query 用空白“ ”分割，儲存至 vector 中，每篇文章都用 for loop 跑一次 vector，判斷裡面 string 是甚麼型態。我有用一個 preOP 來記錄他上一個遇到的 operator 是 AND、OR 還是 NONE，並以此來對我目前為止的 ans 做運算。

```
for(auto & key : Querys){
    if(key==""){
        preOP=AND;
    }
    else if(key==" "){
        preOP=OR;
    }
    else if(isalpha(key[0])){
        if(preOP==NONE) ans=curRoot->Tri_search(key,prefix);
        else if(preOP==AND) ans=ans&&curRoot->Tri_search(key,prefix);
        else if(preOP==OR) ans=ans|curRoot->Tri_search(key,prefix);
        preOP=NONE;
    }
    else if(key[0]=='*'){
        string newkey=alpha(key);
        if(preOP==NONE) ans=curRoot->Tri_search(newkey,exact);
        else if(preOP==AND) ans=ans&&curRoot->Tri_search(newkey,exact);
        else if(preOP==OR) ans=ans|curRoot->Tri_search(newkey,exact);
        preOP=NONE;
    }
    else if(key[0]=='*'){
        string newkey=alpha(key);
        if(preOP==NONE) ans=curRoot->Tri_search(newkey,suffix);
        else if(preOP==AND) ans=ans&&curRoot->Tri_search(newkey,suffix);
        else if(preOP==OR) ans=ans|curRoot->Tri_search(newkey,suffix);
        preOP=NONE;
    }
}
```

B. Challenges I encounter in this project

本次 project 對我而言最困難的部分在於 debug，我的 insert function 和 search function 都很快就順利完成了。但當我完成 code 後，卻發現輸出都是 “Not Found!”，找來找去都看不出我到底哪裡寫錯了，cout 了很多東西出來檢查，有發現 search 的地方永遠不會走到 return true，並且每次都在 query 第一個字母時就找不到匹配的字母而 return false，讓我懷疑我的 insert 是不是 insert 了個寂寞，但我自己看和朋友們幫我看也都說我的 insert 看起來很對，找不出 bug。直到後來我一個朋友說他之前用 struct 來實作也因為疑似指標指向 null 炸開過，後來改成 class 就解決了，於是我也將我本來以 struct 實作的 “Root”(其包含我的 prefix root 和 suffix root)改成 class，並將 insert function 和 search function 都丟進 Root 的 class 中，想不

到如此一來我找了大半天的 bug 就解決了！(因為 struct 會需要指來指去，而 class 將 function 包在裡面就不會遇到這個可能指到 null 又找不出來的問題)普通 data 配 query 終於對了！但發現 data-more 配 query-more 竟然又有不對的地方，於是我將每行 query 結束都加一個換行“\n”來 debug，發現我沒有把 title insert 進 Trie 裡面。經過修改終於全對了(痛哭流涕)。

```
void Tri_insert(string key){
    node* tmpPos = pos;
    node* tmpNeg = neg;
    int len=key.length();
    for(int i=0;i<len;i++){
        int index = isupper(key[i]) ? key[i] - 'A': key[i] - 'a';
        if(! (tmpPos->child[index])){
            tmpPos->child[index] = getNode();
        }
        tmpPos = tmpPos->child[index];
    }
    tmpPos->is_leaf = true;

    for(int i = len-1;i>=0;i--){
        int index = isupper(key[i]) ? key[i] - 'A': key[i] - 'a';
        if(! (tmpNeg->child[index]))
            tmpNeg->child[index] = getNode();
        tmpNeg = tmpNeg->child[index];
    }
    tmpNeg->is_leaf = true;
}

bool Tri_search(string key,int type){
    int len=key.length();
    if(type==suffix){
        node* tmpR=neg;
        for(int i=len-1;i>=0;i--){
            int index = isupper(key[i]) ? key[i] - 'A': key[i] - 'a';
            if(!tmpR->child[index])
                return false;
            tmpR=tmpR->child[index];
        }
        return true;
    }
    else{
        node* tmpR=pos;
        for(int i=0;i<len;i++){
            int index = isupper(key[i]) ? key[i] - 'A': key[i] - 'a';
            if(!tmpR->child[index])
                return false;
            tmpR=tmpR->child[index];
        }
        if(type==prefix)
            return true;
        return (tmpR->is_leaf);
    }
}
```

(如上，將 function 放進 class 後我的 insert 和 search 就不需要傳 root 或 node 進入 function，可有效改善疑似指標指到 null 的問題)

C. References that give me the idea (github/paper...)

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

<https://www.digitalocean.com/community/tutorials/trie-data-structure-in-c-plus-plus>