Data Structure Final Project Report

109062206 張雅涵

1. How I implemented my code

1. 首先從 argv 中取得 data 所在資料夾，接著依照 i=0,1,2…的順序將檔案路徑設為資料夾/i.txt，每次打開檔案就用變數 fi 存取 fstream。
2. 接著 implement Trie 的 structure，以及它的 search 和 insert function。
3. 對於每個 txt 檔，分別用 file_to_trie, file_to_sufftrie 這兩個 function 建立:(1)一個儲存 file 中所有 word 的 trie (2)一個儲存 file 中每個 word 的所有 suffix 的 trie。

a. file_to_trie:

首先用一個 trie_pointer structure 儲存跟屬於某個 file 的 trie 有關的資訊，包括 file 的 index(index.txt)，title，一個指向 trie 的 root 的 pointer，還有為了建立儲存所有 trie 的 linked list 會用到的 next pointer 欄位。

```
struct trie_pointer{
    TrieNode *trie = NULL;
    trie_pointer *next = NULL;
    int file_idx = -1;
    string title;
};
```

接著開始針對 file 建 trie: 先 initialize 一個 trie root，並將指向該 root 的 pointer 存入以參數形式傳入的 trie_pointer 裡。接著用 getline() 從第一行開始逐行讀取 file，並將第一行存入 trie_pointer 的 title 欄位中。再對每一行進行 split 和 word_parse 取得 file 中所有 word 後逐一 insert 到 trie 裡。

```
void file_to_trie(fstream& fi, trie_pointer* p){
    TrieNode* root = getNode();
    p->trie = root;
    string tmp;
    vector<string> tmp_string;
    bool is_title = true;
    while(getline(fi, tmp)){
        if(is_title) p->title =tmp;
        tmp_string = split(tmp, " ");

        vector<string> content = word_parse(tmp_string);

        for(auto &word : content){
            Insert(root, word);
        }
        is_title = false;
    }
}
```

b. file_to_sufftrie:

建立 suffix trie 的過程大致與上面相同，除了 insert 的部分。這次儲存的是每個字的所有 suffix，因此對於每個 word_parse 後得到的 word，用一個迴圈取得他的所有 suffix, 也就是 word 的 substring word[word_len-1:word_len], word[word_len-2:word_len],…, word[0:word_len] ，並將這些 suffix 逐一 insert 入 suffix trie 中。

```
void file_to_sufftrie(fstream& fi, trie_pointer* p){
    TrieNode* root = getNode();
    p->trie = root;
    string tmp;
    vector<string> tmp_string;
    bool is_title = true;
    while(getline(fi, tmp)){
        if(is_title) p->title =tmp;
        tmp_string = split(tmp, " ");

        vector<string> content = word_parse(tmp_string);

        for(auto &word : content){
            int l = word.length();
            for(int i=l-1, cnt=1;i>=0;i--, cnt++){
                string ins = word.substr(i, cnt);
                Insert(root, ins);
            }
        }
        is_title = false;
    }
}
```

4. 上面建立好的所有 trie, suffix_trie 分別用兩個 linked list 儲存起來。
5. 最後是讀取並執行 query 的部分，一樣打開 query 檔案後，執行迴圈，每次讀入一行 query。然後對於每條 query:
   a. Split query string 後得到包含 "word", word, *word*, +, /這五種型態的 string 的 list。
   b. 用一個 string op 儲存當前正在等待執行 set operation(union 或 intersect)，初始值設為""。再用一個 set<int , string>(file_set)儲存所符合 query 的 file 的 index, title(根據 set 特性會依照 index 排序)。
   c. 用一個迴圈遍歷剛才 split 完 query string 得到 list:
      1. 如果這個迴圈遇到的 element 是"+"或是"/"，就更新 op = element。
      2. 如果是要搜索的關鍵字，先用第一個字判斷是要搜索 prefix, suffix 或完整字串，接著用一個 flag 紀錄是否是要搜索 prefix 傳給 search function。Search function 中如果搜尋的是 prefix，就算 traverse 到的最後一個 node 不是一個 word 的結尾也算找到關鍵字。
         接著對 linked list(如果是 prefix/完整字串，搜尋 trie 的 list, 否則搜尋 suffix_trie 的 list)裡每個 trie 進行 search。
      3. 每搜索一個關鍵字就用一個新的 set<int, string>儲存所有找到關鍵字的 file，再根據現在的 op 對新舊兩個 set 進行 union/intersect，並將 file_set 更新為得到的結果。
   d. 最後打開 output.txt，遍歷 file_set 並依序將<int, string>裡 string 的部分(title)寫入 output.txt。

2. Challenges

因為有時間限制，要花一些時間了解各個資料結構的使用方法跟計算時間。還有因為不熟悉 File 還有一些 container 的使用花了一點時間查資料，其他部分因為這學期寫作業時有練習，寫起來沒有遇到甚麼問題。

3. References

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