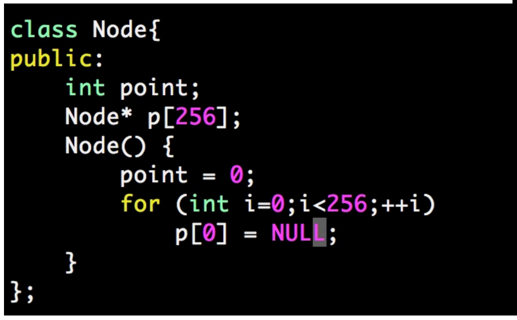
[Group 02]
Le Duong Cong Duc
Nguyen Hoang Gia Bao
Truong Huu Duc

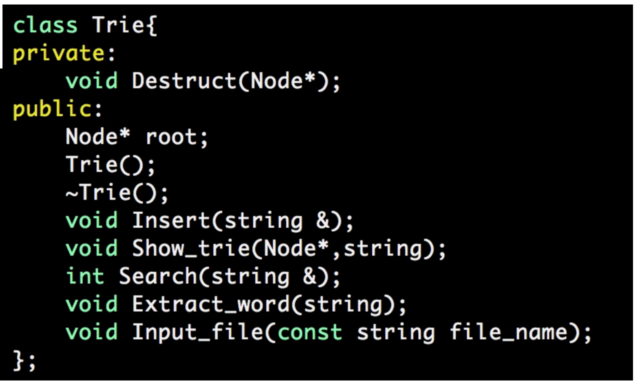
CS163 Final Project Proposal

Data structure: Trie

Trie is an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings. Unlike a binary search tree, no node in the tree stores the key associated with that node; instead, its position in the tree defines the key with which it is associated. All the descendants of a node have a common prefix of the string associated with that node, and the root is associated with the empty string. Values are not necessarily associated with every node. Rather, values tend only to be associated with leaves, and with some inner nodes that correspond to keys of interest.

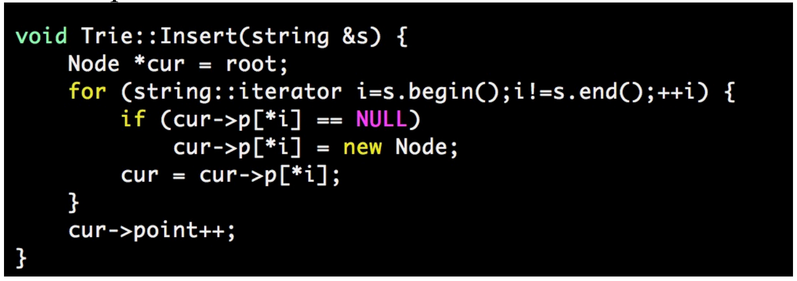


```
class Node{
public:
    int point;
    Node* p[256];
    Node() {
        point = 0;
        for (int i=0;i<256;++i)
            p[0] = NULL;
    }
};
```



```
class Trie{
private:
    void Destruct(Node*);
public:
    Node* root;
    Trie();
    ~Trie();
    void Insert(string &);
    void Show_trie(Node*,string);
    int Search(string &);
    void Extract_word(string);
    void Input_file(const string file_name);
};
```

Firstly, we will process the documents and store all words in our trie (one trie/one file).



```
void Trie::Insert(string &s) {
    Node *cur = root;
    for (string::iterator i=s.begin();i!=s.end();++i) {
        if (cur->p[*i] == NULL)
            cur->p[*i] = new Node;
        cur = cur->p[*i];
    }
    cur->point++;
}
```

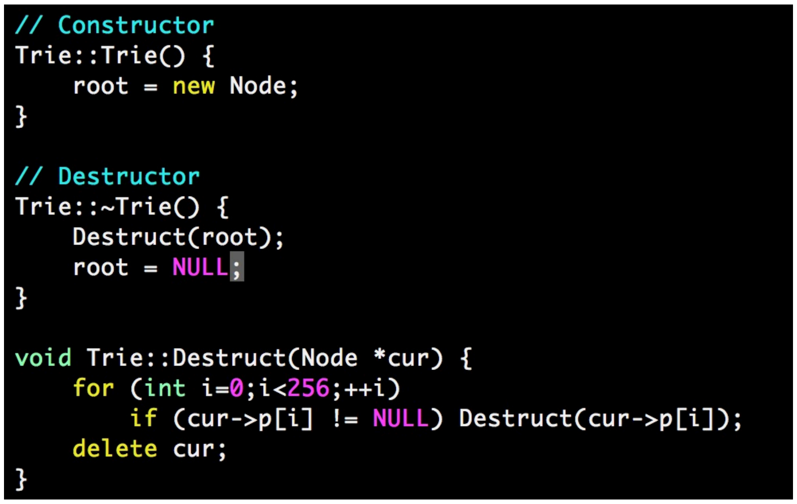
For every input query, we will process the query and search its keywords in the trie for each file and implement algorithm correspond with our query.

Solution for each query:

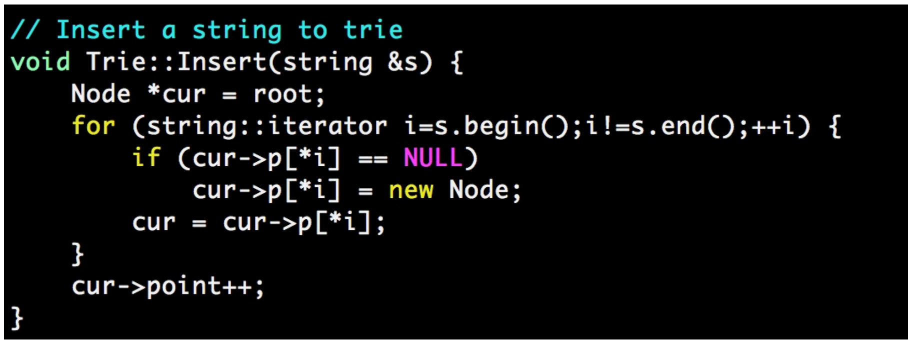
1. ***AND***
   For each text file we have one corresponding trie, we just need to search each operand in this trie. One file is fine if it contains all input operand of AND operator.
2. ***OR***
   For each text file we have one corresponding trie, we just need to search operand one by one in this trie. One file is fine if it contains at least one operand of OR operator.
3. ***Manchester –united***
   For example: A -B
   For each text file we have one corresponding trie, we just need to search for **A** in this trie. If a file has **A** we'll continue to search for **B** in this trie. One file is fine if it contains **A** and doesn't contain **B**.

[Group 02]
Le Duong Cong Duc
Nguyen Hoang Gia Bao
Truong Huu Duc

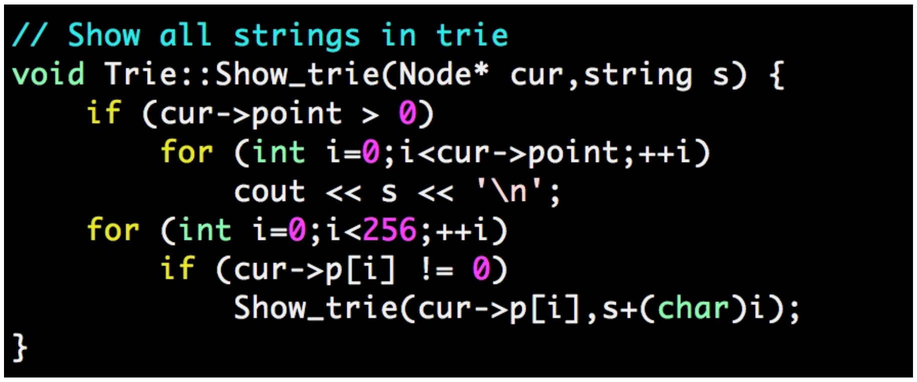
Some function of data structure



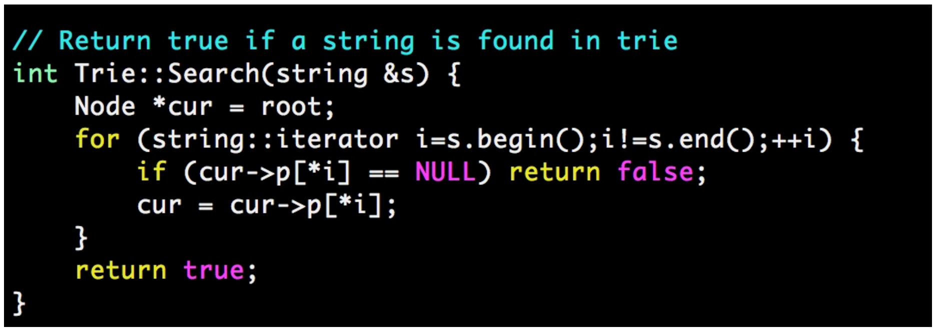
```
// Constructor
Trie::Trie() {
    root = new Node;
}

// Destructor
Trie::~Trie() {
    Destruct(root);
    root = NULL;
}

void Trie::Destruct(Node *cur) {
    for (int i=0;i<256;++i)
        if (cur->p[i] != NULL) Destruct(cur->p[i]);
    delete cur;
}
```



```
// Insert a string to trie
void Trie::Insert(string &s) {
    Node *cur = root;
    for (string::iterator i=s.begin();i!=s.end();++i) {
        if (cur->p[*i] == NULL)
            cur->p[*i] = new Node;
        cur = cur->p[*i];
    }
    cur->point++;
}
```



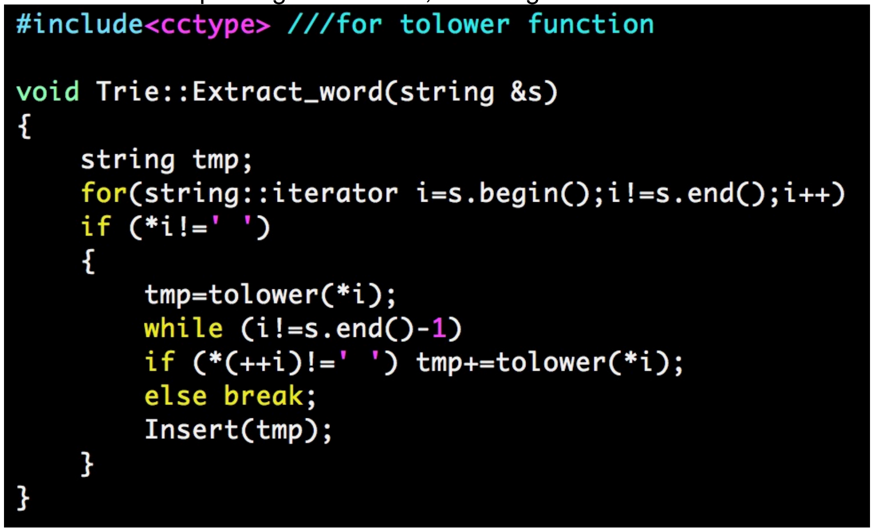
```
// Show all strings in trie
void Trie::Show_trie(Node* cur,string s) {
    if (cur->point > 0)
        for (int i=0;i<cur->point;++i)
            cout << s << '\n';
    for (int i=0;i<256;++i)
        if (cur->p[i] != 0)
            Show_trie(cur->p[i],s+(char)i);
}
```



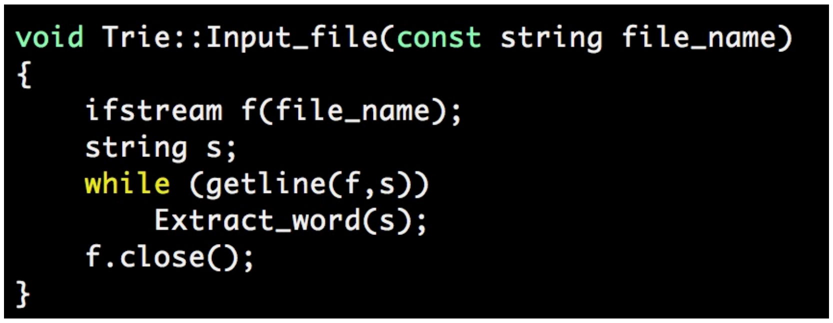
```
// Return true if a string is found in trie
int Trie::Search(string &s) {
    Node *cur = root;
    for (string::iterator i=s.begin();i!=s.end();++i) {
        if (cur->p[*i] == NULL) return false;
        cur = cur->p[*i];
    }
    return true;
}
```

[Group 02]
Le Duong Cong Duc
Nguyen Hoang Gia Bao
Truong Huu Duc

Source code for parsing documents, inserting data



```cpp
#include<cctype> ///for tolower function

void Trie::Extract_word(string &s)
{
    string tmp;
    for(string::iterator i=s.begin();i!=s.end();i++)
    if (*i!=' ')
    {
        tmp=tolower(*i);
        while (i!=s.end()-1)
        if (*(++i)!=' ') tmp+=tolower(*i);
        else break;
        Insert(tmp);
    }
}
```



```cpp
void Trie::Input_file(const string file_name)
{
    ifstream f(file_name);
    string s;
    while (getline(f,s))
        Extract_word(s);
    f.close();
}
```