Note Title 25-10-2018

COL 100

Oct 25, 2018

Last Class:-

ADT:- Abstract Data Types (Stanf)

Sets:- Collection of elements [No Duplicates, No Order]

Maps:- Collection of pairs of elements

(STL)	
① m.put(key,value)	m[key] = value
② m[key]	m[key]
③ m.get(key)	m.at(key)
③ m.remove(key)	m.erase(key)
④ m.contains(key(key)	

m.find(key) != m.end()

Dictionary:- (key, value)

map<string, int> dictionary;

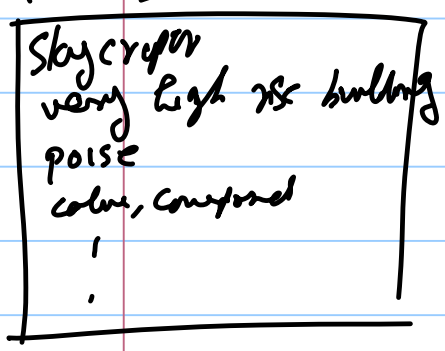
word — the time it occurs

<string, Word.strings> meaning

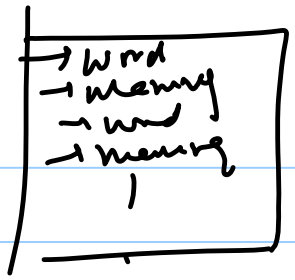
Program 1:-

Problem statement:-

File:- "dictionary.txt"



Write a program to read a file:- each line contains a word followed by its definition on next line
→ create a dictionary out of it.




```
int main() {
    string filename;
    ifstream inpfile; cin>> filename;
    inpfile.open(filename);
    map<string, string> dictionary;
    string word;
    while( getline(inpfile, word )) {
        string meaning;
        getline (inpfile, meaning);
        dictionary[word] = meaning;

    }
    inpfile.close();
```

```
while( true ) {
    string word;
    cout << "Enter during Word" << endl;
    cin >> word;
    If ( dictionary.find(word)
        != dictionary.end() ) {
        cout << "The meaning
        of " << word << " is:"
            << dictionary[word]
            << endl;
    } else {
        cout << "The " << word
            << " Does not exist in my Dictionary." << endl;
    }
}
```

// dictionary.contains(key) (word)) {

3

the {

3

3

```
map<string, int> phonebook;
  phonebook["Dracula"] = 1234;
  phonebook["Dean"] = 7356;

for (pair<string, int> name_phone : phonebook) {
        string name = name_phone.first;
        int phone = name_phone.second;
        cout << "I am going to call " << name;
        cout << " At Number " << phone << endl;
}
```

STL

Lexicographic

type

phone = phonebook[name];

```
for(string name : phonebook){
   int phone = phonebook[name];
   cout << ...
}
```

Standard

Input file: in.txt

```
to be or not be
hello world
the world is not enough
i think therefore i am
```

Output:

```
am : 1
be : 2
enough : 1
hello : 1
i : 2
is : 1
not : 2
or : 1
the : 1
therefore : 1
think : 1
to : 2
world : 2
```

Program 4:

```
be : 2
i : 2
not : 2
or : 2
world : 2
am : 1
1
!
.
.
.
```

in.txt

```
string word;
ifstream inpfile("in.txt");
map<string, int> wc;
while (inpfile >> word) {
    if (wc.find(word) == wc.end())
        wc[word] = 1;
    else
        wc[word] + +;
}
```

if (!wc.contains(key word))

map<int, set<string>>

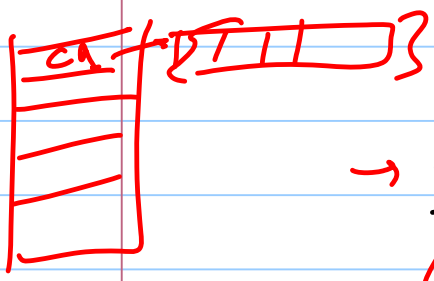
// wc: - map of words to counts

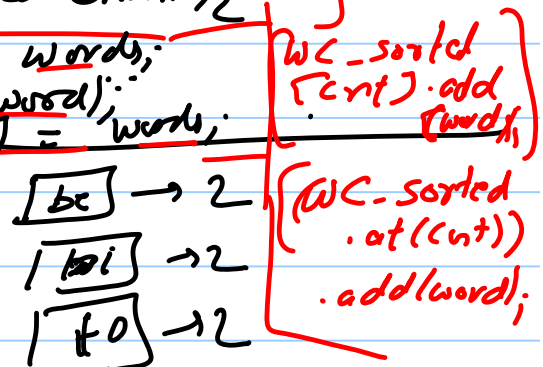
```
map<int, set<string>>  wc_sorted;
for (pair<string, int> w_cnt : wc) {
    string word = w_cnt.first;
    int cnt = w_cnt.second;
    if (wc_sorted.find(cnt) == wc_sorted.end()) {
        set<string> words;
        words.add(word);
        wc_sorted[cnt] = words;
    } else {
        (wc_sorted[cnt]).add(word);
    }
}
```

1 2 3

Last step

!(contains)





```
for(pair< int, set<string> > cnt_to_w : wc_sorted){
    int cnt = cnt_to_w.first;
    set<string> words = cnt_to_w.second;
    for(string word: words) {
        cout << word << ": "
             << cnt << endl;
    }
}
```

currName: "Gautam"

Gautam
Rajdhani
Masala Mix
Sanchita
Rajdhani
Rainbow
nigh_mess_Kailash
Mukesh
|
|

Suggestions:-

1. Map
from:-
name to set of places

2. map
from places → int
set<string>

Rajdhani	4
masala mix	4
Rainbow	2

|

Time Complexity: (Sets & Maps)

set<int> s1 set<int> s2; | Size is O(n)

is (s1 ⊆ s2) {1,2,3,5} {1,3,5,11}

→ ~~s1 ∪ s2~~

const const

bool isSubsetOf (set<int> s1, set<int> s2) {

O(n log n) → for (int a : s1) { → for (int b: s2) { if (a == b) { } if (s2.find(k) == s2.end()) return false;

return true;

}

Set Union

Set<int> union(Set<int> s1, Set<int> s2) {

Set<int> s3;

for(int a: s1) {
    s3.add(a);
}

for(int b: s2) {
    s3.add(b);
}

return s3;

}

O(nlogn)

O(

```
for( int a: s1) {
log(n)  if( s2.find(a) != s2.end() ) {
log(m)      s3.add(a);
        }
}
```

$\log n + \log n + \cdots + \log n$ (n terms)

$O(n \log n \times 2) \approx O(n \log n)$

m[key]: value O(log n)

map<int, string> O

↳ find a particular value O(n)

~~for(auto m:~~

for(auto a: m){

}

(S1 == S2)

Stanford (in built fn)] O(n)

→ [S.isSubsetOf(S2) → O(n) time

{0, 2, 3, 4} {1, 2, 3, 4, 5}

{0, 2, 4, 5} {1, 3, 4, 5}

for(int a : S1) {

  for(int b : S2) {

    if (a == b)

      break; (a == b)

  }

}

AreEqual(set S1 <int> set S1 2 int>)

{ for(int a : S1) {

  if (S2.find(a) != S2.end())

}