FREEMIND

Date

Page

Lecture-59 (Set in C++)

What is a Set? (Standard Template lib.)

→ Set is an STL container use to store unique values
→ store values in ordered state. (inc./dec.)
→ no indexing, elements are identified by their own values
→ ones value is inserted in a set, it can't be modified

Advantages of set:

→ Unique Values
→ Ordered
→ dynamic size, no overflowing errors
→ faster

insertion, deletion, search } O(log N)

↳ binary Search Tree (BST) / Red-Black Tree

Disadvantages:-

→ Cannot access elements using indexing.
→ Uses more memory than array.
→ Not suitable for large data size.

Declaration of a Set

```
#include <set>
set<data_type> set_name;

set<int> set1 = {1,2,3,4};
```

→ By default, values are stored in increasing order.

set <data_type, greater<datatype>> set_name;

↓ decreasing order

⭐ Inserting elements into a set

set_name.insert(Value)

Time O(log N)

set1.insert(4);
set1.insert(3);

→ returns an iterator to the inserted value.
↳ iterate through STL containers.

⭐ Traversal of a Set.

Iterator → for traversing a set

r.begin();
reverse
r.end();

Set_name.began() → pointing to 1st element

Set_name.end() → pointing to position after the last element.

```
Set<int>::iterator itr;
for (itr = Set1.begin(); itr != set1.end(); itr++) {
        cout << *itr << " ";
} cout << endl;

cout << set1.size() << endl;    //give Size
```

FREEMIND

Date

Page

→ Using for each loop

```
for (auto value : Set1){
        cout << value << " ";
} cout << endl;
```

```
auto itr = Set1.begin();
advance (itr, 3);
→ (move itr by 3 steps)
```

Deleting elements from a set.

1. set_name.erase (value) → O(log N)
2. set_name.erase (position) → O(log N)
3. set_name.erase (start_pos, end_pos) → O(N), N = no. of elements in the range

[delete elements from start_pos including it, till end_pos, excluding it]

Member functions of a Set container

→ size(), max_size() → max no. of elements set can hold.
→ empty()
→ clear() → removes all elements from set.
→ find() → returns position of element if present, else returns end iterator.

```
// Search operation
if (Set1.find(4) != Set1.end()){
        cout << "value is present" << endl;
} else {
        cout << "value not present" << endl;
}
```

FREEMIND

Date

Page

→ count() → returns no. of occurences of an element.
→ lower_bound() → returns element if present, else returns just greater value.
→ upper_bound() → returns the next greater value.

Q) Cherry's birthday is coming this month! She wants to plan a birthday party and is preparing an invite list with her friend Aashi. She asks Aashii to tell her names to add to the list.
Aashii is a random guy and keeps coming up with names of people randomly to add to the invite list, even if the name is already on the list! ~~cha~~ cherry hates ~~re~~ redundancy and hence, enlists the names only ~~one~~ once.

Find the final invite list, that contain names without repetition.

Input:- First line of each test contains an integer N, the no. of names that Aashi pops up with.

Output:- output the final invite-list with each name in a new line. The names in the final invite-list are sorted lexicographically.

Ans
```
#include<iostream>
#include<set>
using namespace std;
```

```
int main() {
    set<string> invitelist;

    int n
    cin>> n;

    while (n--) {

        string name;
        cin>> name;
        insert
        invitelist.insert(name);
    }
    for (auto name: invitelist) {
        cout << name << endl;
    }
    return 0;
}
```

Q2 => Add the common elements :-

Given 2 vectors V1 and V2. Find out the common elements b/w the ~~two~~ two and return the sum of them.

input: V1 = {1,1,2,3,3,3}
V2 = {5,6,7,5,2,3,6}

Output :- 5

Explanation :- The values common b/w V1 and V2 are : 2 & 3
so, sum is 2+3 = 5.

Sol

```
# include <vector>
# include <set>
```

```
int main() {
    int n, m;
    cin >> n >> m;
    vector<int> v1; vector<int> v2;
    cout << "Enter elements for vector v1" << endl;
    for (auto int i=0; i<n; i++) {
        cin >> v1[i];
    }
    cout << "Enter elements for vector v2" << endl;
    for (int i=0; i<m; i++) {
        cin >> v2[i];
    }

    set<int> s;
    int ans_sum = 0;

    for (auto ele : v1) {
        s.insert(ele);
    }
    for (auto ele : v2) {
        if (s.find(ele) != s.end()) {
            ans_sum += ele;
        }
    }
    cout << "ans: " << ans_sum << endl;

    return 0;
}
```

Q5 Check if string has all english alphabets.

input: abcdefghijklmnopqrstuvwxyz	output: Yes
input: Physics Wallah	output: No

FREEMIND

Date

Page

```
#include <set>
using namespace std;

bool checkallalphabets (string s) {

    if (s.length() < 26) {
        return false;
    }
    transform (s.begin(), s.end(), s.begin(), ::tolower);

    set<char> alphabets;
    for (auto ch : s) {
        alphabets.insert(ch);
    }
    return (alphabets.size() == 26);
}
int main() {
    String name;
    cin >> name;
    if (checkallalphabets (name)) {
        cout << "Yess" << endl;
    } else {
        cout << "No" << endl;
    }
    return 0;
}
```

(A) Unordered_set

→ Values are store in unordered fashion.

→ Unique Values

→ No indexing

→ Values can't be modified.

FREEMIND

Date

Page

Insertion
Deletion
Search
→ O(1)
avg. time complexity

Hashing

23, 45, 67, 32
hash function = mod 10

23 mod 10 = 3

2	32
3	23
4	
5	45
6	
7	67

#Code

```
#include <unordered_set>
using namespace std;
int main() {
    unordered_set<int> s1;
    s1.insert(3)
    s1.insert(1)
    s1.insert(2)
    s1.insert(1)
    for(auto ele:s1){
        cout<<ele<<" ";
    }
    return 0;
}
```

//213 (Unordered)

FREEMIND

Date

Page

(A) Member functions of Unordered_set

1) insert

single element → O(1) avg.
O(N) worst ← size ≥ capacity → rehashing

multiple elements → O(n) avg. → n is no. of elements being inserted
- O(n*(n+1)) worst
↓
size of unordered set

2) Deletion

erase (value)
erase (iterator)
erase (start_itr, end_itr)

~~Single Element~~
O(1) avg.
O(N) worst

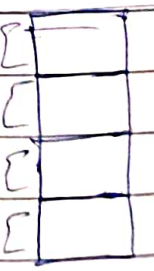
3) Count()

O(1) avg.
O(N) worst

4) find()

O(1) avg
O(N) Worst

Hashing



load_factor → size of unordered set / bucket_count

rehash(n)

sets the no. of bucket to n or more
O(N) avg
O(N²) worst

FREEMIND

Date

Page

Multiset?

→ it can store duplicate values.

→ other same as sets

```
#include <set>
using namespace std;

int main() {
    multiset<int> ms;
    ms.insert(1);
    ms.insert(3);
        (2);
        (3);
    for(auto value: ms) {
        cout << value << " ";   // 1 2 3 3
    }
    return 0;
}
```

```
multiset<int, greater<int>> ms;
// 3 3 2 1
```

⇒ Member functions of multiset

1) insert() → O(log N)

2) Deletion

erase(value) → O(log N)

erase(position)

erase(start_pos, end_pos) → O(N)

erase(2)

1 2 2 3 3

3) find()

is lowerbound of element searched if found, else end iterator — O(log N)

1 2 2 2 3 3

4) Count()
↳ no. of occurance

5) lower_bound() — O(log N)
↳ iterator pointing to first occurance of val if present else position the next greater value.

6) Upper_bound()
↳ position of next greater value.

★ Unordered_multiset
→ allows duplicate values
→ values are not ordered
→ values will be identified by itself.
→ values can't be modified.

⇒ Member functions of Unordered multiset.

1) insert()
- single element — O(1) avg — O(N) worst
- multiple elements — O(n) avg — O(n*(N+1)) worst (rehashing)

2) Deletion
erase()
O(1) avg
O(N) worst

3) find()
O(1) avg
O(N) worst

4) count() ↳ no. of occurances
O(n) avg
O(N) worst
↳ N → size

FREEMIND

Date

Page

Q→ Given n integers (can be duplicates), print the second smallest integer. If it does not exist print -1.

input: 4 output: 1
1 2 2 -4

Sol→
```
#include <set>
#include <vector>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> V[n]; set<int> s;
    for (int i=0; i<n; i++) {
        cin >> V[i];
    }
    for (auto val: V) {
        s.insert(val);
    }
    auto itr = s.begin();
    itr++;
    cout << "Second smallest number:" << *itr << endl;
    return 0;
}
```

Q→ Given the no. of question as n, and marks for the correct answer answer as p and q marks for the incorrect answer. One can either attempt to solve the question in an examination and get p marks if the answer is right, or q marks if the (either) answer is wrong, or leave the question unattended and get 0 marks.

FREEMIND

Date

Page

The task is to find the count of all the ~~front~~ different possible marks that one can score in the examination.

input: n = 2, p = 1, q = -1
output: 5
Explanation:- The different possible marks are: -2, -1, 0, 1, 2

Sol ⇒

```
# include <unordered_set>
Using namespace std;

int main () {
int n, p, q;
cin >> n >> p >> q;
unordered_set <int> s;
for (int i = 0; i < n ; i++) {
    for (int j = 0; j i < n; j++) {

        int correct = i;
        int incorrect = j;
        int unattended = n - (i + j);

        if (unattended >= 0) {
          int score = correct * p + incorrect * q
          s.insert (score);
        }
        else {
            break;
        }
    }
}
cout << "Ans:" << s.size() << endl;
return 0;
}
```

```
for (auto score : s) {
    cout << score << " ";
} cout << endl;
```