JUNIORS 2

SFE ACPC CLUB
CODE YOUR WAY

Non-Linear Data Structures

Session 4

Priority Queue

Priority queues are a type of container adaptors , specifically designed such that it sorts its elements decreasing with there priorities - first element(top element) has the highest priority

- To declare a priority queue:

```
priority_queue<dataType> name;
```

Priority is its element values by default

To sort elements in increasing order use:

- `priority_queue<data type,vector<int>,greater<int>> name;`

Important built-in functions of Priority Queue

empty	Test whether Priority Queue is empty	O(1)
size	Return size of the Priority Queue	O(1)
top	Access top element of the Priority Queue	O(1)
push	Insert an element on the top of the Priority Queue	O(Log(n))
pop	Remove the top element of the Priority Queue	O(Log(n))
swap	Swap contents of two Priority Queues	O(1)

```cpp
int main()
{
    //Priority is its element's values by default
    priority_queue<int> pq1;
    priority_queue<char> pq2;
    priority_queue<string> pq3;
    pq3.push("abc");
    pq3.push("def");
    pq3.push("abf");
    pq2.push('a');
    pq2.push('b');
    pq2.push('c');
    while(!pq2.empty()) {
        cout<<pq2.top()<<" "<<pq3.top()<<endl;
        pq2.pop();
        pq3.pop(); }
    return 0;
}
```

```
c def
b abf
a abc
```

Set

Sets are containers that store unique elements following a increasing order.

- To declare a set:

```
set<dataType> name;
```

Important functions of the Set:

begin	Return iterator to the beginning.	O(1)
end	Return iterator to the end.	O(1)
empty	Test whether set is empty	O(1)
size	Return size of the set.	O(1)
insert	Insert element.	O(Log(n))

Important functions of the Set:

erase	Erase element.	O(Log(n))
swap	Swap contents of two sets	O(1)
clear	Clear content.	O(n)
find	Get iterator of the element	O(Log(n))
count	Count elements with a specific value	O(Log(n))
lower_bound	Return the iterator of the lower bound.	O(Log(n))
upper_bound	Return the iterator of the upper bound.	O(Log(n))
equal_range	Get range of equal elements.	O(Log(n))

```cpp
set<int> s;
s.insert(0); //O(log n)
s.insert(5);
s.insert(10);
s.insert(15);
int x[8]={10,20,25,30,35,40,45,50}; // 10 already in set, not inserted
s.insert(x,x+8); //O(8*log n),inserting from x[0] to x[8-1]
for(auto it=s.begin();it!=s.end();++it) //auto it = set<int>::iterator it
    cout<<*it<<" ";
cout<<endl;
s.erase(30);//O(log n)
s.erase(s.begin());//O(1)
auto it=s.find(35);//O(log n)
s.erase(it,s.end());//O(s.end() - it)
for(int i : s) //Enhanced for
    cout<<i<<" ";
cout<<endl;
//we can use "count" as "find" but it will return 0 or 1 not an iterator
if(s.count(10)) cout<<"10 is found";
else cout<<"10 isn't found";
```

```
0 5 10 15 20 25 30 35 40 45 50
5 10 15 20 25
10 is found
```

Multi Set

- It’s the same of “Set” but it can have duplicate elements.
- To declare a multi set: multiset<dataType> name;

Unordered Set

- It’s the same of “Set” but it isn’t ordered and it’s quite faster than “Set”.
- To declare a unordered set: unordered_set<dataType> name;

Unordered Multiset

- -It’s the same of “Set” but it can have duplicate elements ,it isn’t ordered and it’s quite faster than “Set”.
- To declare a unordered multiset: unordered_multiset<dataType> name;

Map

Maps are associative containers that store elements formed by a combination of a key value and a mapped value, following a decreasing order.

In a map, the key values are generally used to sort and uniquely identify the elements, while the mapped values store the content associated to this key.

The types of key and mapped value may differ, and are grouped together in member type , which is a pair type combining both.

- To declare a map:

```
map<dataType(key),dataType(value)> name;
```

Important functions of the Map:

Map is the same as the Set
Except: Map has “operator[]”.

begin	Return iterator to the beginning.	O(1)
end	Return iterator to the end.	O(1)
empty	Test whether map is empty	O(1)
size	Return size of the map.	O(1)
insert	Insert element.	O(Log(n))

Important functions of the Map:

erase	Erase element.	O(Log(n))
swap	Swap contents of two maps	O(1)
clear	Clear content.	O(n)
find	Get iterator of the element	O(Log(n))
count	Count elements with a specific value	O(Log(n))
lower_bound	Return the iterator of the lower bound.	O(Log(n))
upper_bound	Return the iterator of the upper bound.	O(Log(n))
equal_range	Get range of equal elements.	O(Log(n))

```cpp
map<string ,int> mp;
mp["Book1"]=50;
mp["Book2"]=75;
mp["Book1"]=100;//overriding
mp.insert(make_pair("Book1",25));//"Book1" is already in map,not inserted
pair<string,int> x[3];
x[0]=make_pair("Book3",60);
x[1]=make_pair("Book4",40);
x[2]=make_pair("Book5",80);
mp.insert(x,x+3);
for(auto it=mp.begin(); it!=mp.end(); ++it)
    cout << it->first << " => " << it->second << endl;
    //we use "."operator to access an object directly
    //we use "->"operator to access an object that pointer or iterator points to
    //so we did not use "*"operator with the iterator
cout<<endl;
mp.erase("Book3");//O(long n) , we use key only to delete key and its value
mp.erase(mp.begin());//O(1);
auto it=mp.find("Book4");//O(log n)
mp.erase(it,mp.end());//O(mp.end() - it)
for(auto i : mp)//"i" here is a pair (i.e. object) not an iterator
    cout << i.first << " => " << i.second << endl;
```

```
Book1 => 100
Book2 => 75
Book3 => 60
Book4 => 40
Book5 => 80

Book2 => 75
```

Multi Map

- It's the same of "Map" but it can have duplicated keys.
- To declare a multimap: multimap<data type(key),data type(value)> name;
- It doesn't have "operator[]" because of multiple keys with equivalent values.

Unordered Map

- It's the same of "Map" but it isn't ordered and it's quite faster than "Map".
- To declare an unordered map: unordered_map<data type(key),data type(value)> name;

Unordered Multimap

- It's the same of "Map" but it can have duplicate keys ,it isn't ordered and it's quite faster than "Map".
- To declare a unordered multi map:unordered_multimap<data type(key),data type(value)> name;
- It doesn't have "operator[]" because of multiple keys with equivalent values.

Assignment Link

https://a2oj.com/contest?ID=35244