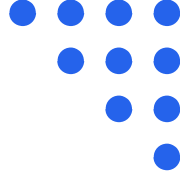


More Operations on Ordered Sets

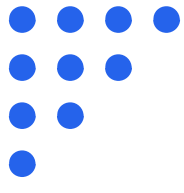
<https://usaco.guide/silver/intro-sorted-sets>



CP Initiative
joincpi.org



Ordered vs. Unordered Sets/Maps



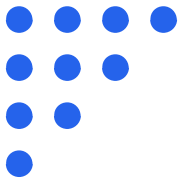
Ordered Sets/Maps

- Complexity: $O(\log n)$
- Stores elements in sorted order
 - For maps, sorts by key
- C++: `set` and `map`
- Java: `TreeSet` and `TreeMap`

Unordered Sets/Maps

- Complexity: $O(1)$ with high constant factor
- Stores elements in random order
- C++: `unordered_set` and `unordered_map`
- Java: `HashSet` and `HashMap`

Recommendation: Use **ordered sets/maps** unless you need to optimize complexity (rare)



Iterators

- Can be used to loop through sets
- Iterators used on unordered sets return elements in random order
- Consult USACO Guide for more information

```
// Java
Iterator it = set.iterator();
while(it.hasNext()){
    Integer i = (Integer)it.next();
    System.out.print(i + " ");
}
```

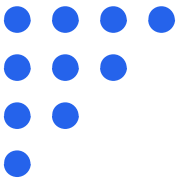
```
// C++
for (auto it = s.begin(); it != s.end(); it++) {
    cout << *it << "\n";
}
```

Iterators Warning

Warning: Don't modify the set while using iterators! This includes for-each loops, which use iterators internally.

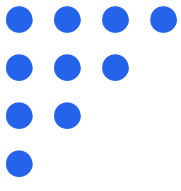
```
// Java
for (int i : set) {
    // don't modify the set here!
    System.out.print(i + " ");
}
```

```
// C++
for (int i : mySet) {
    // don't modify the set here!
    cout << i << " ";
}
```

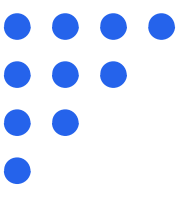


Multisets

- A multiset is a sorted set that allows you to store multiple copies of the same element.
- Normally with regular sets, you can only have one copy of an element at a time.
- C++: `multiset`
- Java: write your own with a `TreeMap`!



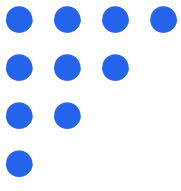
Multisets in C++



- `count()`: returns the number of times an element is present in the multiset.
 - However, this method takes time **linear** in the number of matches so you shouldn't use it in a contest.
- If you want to remove a value **once**, make sure to use `multiset.erase(multiset.find(val))` rather than `multiset.erase(val)`
The latter will remove **all** instances of `val`.



Multisets in Java



- While there is no multiset in Java, we can implement one using the `TreeMap` from values to their respective frequencies.
- We declare the `TreeMap` implementation globally so that we can write functions for adding and removing elements from it.

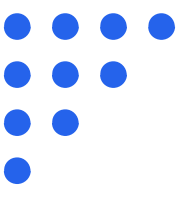
```

static TreeMap<Integer, Integer> multiset = new TreeMap<Integer, Integer>();
public static void main(String[] args){
    ...
}

static void add(int x){
    if(multiset.containsKey(x)){
        multiset.put(x, multiset.get(x) + 1);
    } else {
        multiset.put(x, 1);
    }
}

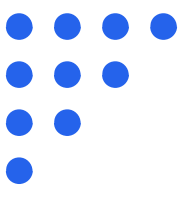
static void remove(int x){
    multiset.put(x, multiset.get(x) - 1);
    if(multiset.get(x) == 0){
        multiset.remove(x);
    }
}

```



Priority Queues

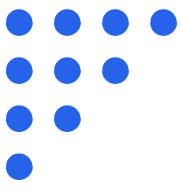
- A **priority queue** (or **heap**) supports the following operations:
 - Insertion of elements
 - Deletion of element with the highest “priority” (value)
 - Retrieval of highest “priority” (value) element
 - All in $O(\log N)$ time
- Simpler & faster than sets -- use Priority Queues over sets whenever possible



Priority Queues (Java)

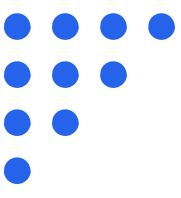
Java retrieves elements of the **lowest** “priority” (value) rather than highest.

```
// Java
PriorityQueue<Integer> pq = new PriorityQueue<Integer>();
pq.add(7); // [7]
pq.add(2); // [7, 2]
pq.add(1); // [7, 2, 1]
pq.add(5); // [7, 5, 2, 1]
System.out.println(pq.peek()); // 1
pq.poll(); // [7, 5, 2]
pq.poll(); // [7, 5]
pq.add(6); // [7, 6, 5]
```



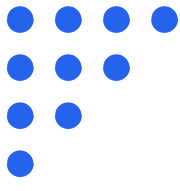
Priority Queues (C++)

```
// C++
priority_queue<int> pq;
pq.push(7); // [7]
pq.push(2); // [2, 7]
pq.push(1); // [1, 2, 7]
pq.push(5); // [1, 2, 5, 7]
cout << pq.top() << endl; // 7
pq.pop(); // [1, 2, 5]
pq.pop(); // [1, 2]
pq.push(6); // [1, 2, 6]
```

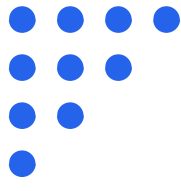


Ordered Sets Example Problem

[CSES - Concert Tickets](#)



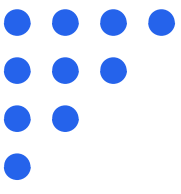
Concert Tickets Solution Sketch



- Put the ticket prices into a multiset
- For each customer, get the first ticket price that doesn't exceed their limit

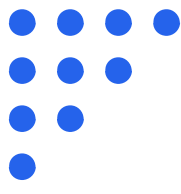
Concert Tickets Implementation Tips

- C++
 - `multiset.insert(x)`: Adds the element to the multiset
 - `multiset.lower_bound(x)`: Returns an iterator to the first element $\geq x$
 - `multiset.upper_bound(x)`: Returns an iterator to the first element strictly greater than x
 - `multiset.begin()`: Returns an iterator pointing to the smallest element in the multiset
 - `--it`: Moves the iterator back one position (so it points to the element before the current element).
- Java
 - Refer to USACO Guide & use a `TreeMap` to implement a multiset!



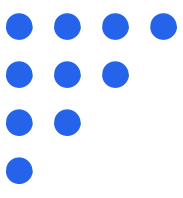
Concert Tickets Solution

[Solution - Concert Tickets \(CSES\) · USACO Guide](#)



Ordered Sets: Challenge Problem

[CSES - Traffic Lights](#)



Ordered Sets: Challenge Problem 2

[Codeforces 702C - Cellular Network](#)

