Sets

January 18, 2026

1 Sets and Unordered Sets - Associative Containers

1.1 External Resources

- C++ Reference for Set Associative Containers: https://en.cppreference.com/w/cpp/container.html
- YouTube Video - https://youtu.be/cHdo1y3oPWA
- YouTube Podcast - https://youtu.be/bJrbqhLTbfU
- NotebookLM learning materials - https://notebooklm.google.com/notebook/d0b8dfd3-cbf9-4808-8dd5-07b2870b1eb1

1.2 Ordered Set

- Documentation: https://www.cppreference.com/w/cpp/container/set.html
- A collection of unique elements, sorted by their values. It provides fast search, insertion, and deletion operations.
- Sets are usually implemented as `Red-black trees`
-
- Example usage:

[1]:
```
#include <iostream>
#include <set>
using namespace std;
```

[2]:
```
set<int> numbers = {5, 2, 8, 1, 1, 3, 2, 5};
```

[3]:
```
// nodes/elements are always sorted in ascending order
numbers
```

[3]:
```
{ 1, 2, 3, 5, 8 }
```

[4]:
```
numbers.insert(100);
```

[5]:
```
numbers
```

[5]:
```
{ 1, 2, 3, 5, 8, 100 }
```

[ ]:
```
// check if an element exists
// using find() method; member function
```

```
if (numbers.find(3) != numbers.end()) {
    cout << "3 found in the set." << endl;
} else {
    cout << "3 not found in the set." << endl;
}
```

```
3 found in the set.
```

[7]:
```
// Output the contents of the set
for (const int& num : numbers) {
    cout << num << " ";
}
```

```
1 2 3 5 8 100
```

1.3 Unordered Set

- https://www.cppreference.com/w/cpp/container/unordered_set.html
- A collection of unique elements, but unlike `set`, the elements are not sorted. They are organized into buckets based on their hash values.
- item lookup, insertion, and deletion have average time complexity of O(1)
- Example usage:

[8]:
```
#include <iostream>
#include <unordered_set>
using namespace std;
```

[10]:
```
unordered_set<int> numbers1 = {5, 2, 8, 1, 1, 3, 2, 5};
```

[ ]:
```
// no specific order of elements
numbers1
```

[ ]:
```
{ 3, 1, 8, 2, 5 }
```

[13]:
```
// check if an element exists
if (numbers1.find(100) != numbers1.end()) {
    cout << "100 found in the unordered set." << endl;
} else {
    cout << "100 not found in the unordered set." << endl;
}
```

```
100 not found in the unordered set.
```

[14]:
```
numbers1.insert(100);
```

[15]:
```
// Output the contents of the unordered set
for (const int& num : numbers1) {
    cout << num << " ";
}
```

```
3 100 1 8 2 5
```

```
[16]: numbers1.erase(2);
```

```
[17]: numbers1
```

```
[17]: { 3, 100, 1, 8, 5 }
```

```
[18]: numbers1.erase(2);
```

```
[19]: numbers1
```

```
[19]: { 3, 100, 1, 8, 5 }
```

1.4 Kattis problems for demo

- Biðröð - https://open.kattis.com/problems/bidrod
  - Hint: unordered_set to track unique songs
- Knights Move - https://open.kattis.com/problems/knightsmove
  - Hint: set (ordered) to track possible moves of the knight in sorted order

1.5 Kattis Problems

- Guest List - https://open.kattis.com/problems/guestlist
  - unordered set
- Korok Phrases - https://open.kattis.com/problems/korokphrases
  - unordered set
- Midjan - https://open.kattis.com/problems/midjan
  - ordered set - set differences
- CD - https://open.kattis.com/problems/cd
  - ordered set - set intersection
- Keyboardd - https://open.kattis.com/problems/keyboardd
  - unordered map
- Shopping List - https://open.kattis.com/problems/shoppinglist
  - sorted set
- Select Group - https://open.kattis.com/problems/selectgroup
  - stack and set
- Tag - https://open.kattis.com/problems/jage
  - Hint: use two sets to simulate hunters and cheaters
  - print the sorted cheaters after simulation
- Instagraph - https://open.kattis.com/problems/instagraph
  - Hint: use unordered_set to track unique followers

```
[ ]:
```