Using Set in C++ Standard Template Library (STL)

1. Introduction

When we say set, we refer to the ordered set implemented using the library.
std::set is an associative container in the C++ Standard Template Library (STL) that stores **unique elements** in a **sorted order**.

Unlike vectors or lists, a set does not allow duplicate values and automatically maintains ordering. Internally, a set is usually implemented using a **balanced binary search tree**, which allows efficient searching, insertion, and deletion.

2. Setup and initialization

To use a set in C++, the header must be included:

```
#include <set>
Declaring a Set
```

The syntax to declare a set is:

```
std::set<type> set_name;
```

Example:

```
std::set<int> s;
```

3. Core operations

A set does not support index-based access. Elements are accessed through iterators or search-related functions.

3.1/ Adding items (insert)

The **insert(item)** function adds an element to the set only if it does not already exist.

Time Complexity: O(log n)

Example:

```
s = {2, 5, 8}
```

```
s.insert(5)   -> {2, 5, 8}   -> 5 already exists
s.insert(3)   -> {2, 3, 5, 8}
```

3.2/ Accessing items (find)

find(item) searches for an element and returns an iterator to it if found, or **end()** if not found.

Time Complexity: O(log n)

Example:

```
s = {2, 3, 5, 8}

s.find(5) -> found
s.find(7) -> not found
```

3.3/ Removing items (erase)

The **erase(item)** function removes the specified element from the set.

Time Complexity: O(log n)

Example:

```
s = {2, 3, 5, 8}

s.erase(3)
s = {2, 5, 8}
```

4. Status operations

empty() Returns true if the set contains no elements.

size() Returns the number of elements currently in the set.

count(item) Returns 1 if the item exists in the set, otherwise returns 0.

5. Conclusion

std::set is commonly used for:

- Storing unique values
- Fast searching in sorted data
- Automatically removing duplicates
- Maintaining ordered collections

The set guarantees that all elements are unique and sorted, making it ideal when order and uniqueness are required.

6. Problems using std::set

6.1/ https://leetcode.com/problems/merge-similar-items/description/

This problem requires returning a 2D integer array where values are sorted in ascending order and unique. A set can be used to store elements as (value, weight) pairs. When duplicate values occur, their weights are summed. This approach efficiently maintains ordering and uniqueness while merging data from two vectors.

6.2/ https://leetcode.com/problems/most-frequent-ids/description/

In this problem, a custom structure can be used inside a set:

```
struct Node {
    int freq;
    int num;
    bool operator<(const Node& obj) const {
        return freq < obj.freq;
    }
};
```

The overloaded operator< allows the set to maintain elements sorted by frequency. At each step:

Update the frequency of an element if it exists

Remove the old node and insert the updated one

Retrieve the element with the maximum frequency efficiently

6.3/ https://leetcode.com/problems/longest-uploaded-prefix/

A set is used to record uploaded video IDs. An integer variable m tracks the current longest uploaded prefix (LUP). Each time a new video is uploaded, the set is checked for m + 1. If it exists, m is incremented.

The set is used for its fast searching capability.