Set

A set in STL is a container that stores unique elements in a particular order. Every operation on a set takes **O(1)** complexity in the average case and takes **O(n)** in the worst case.

Syntax:

```
set<object_type> variable_name;
```

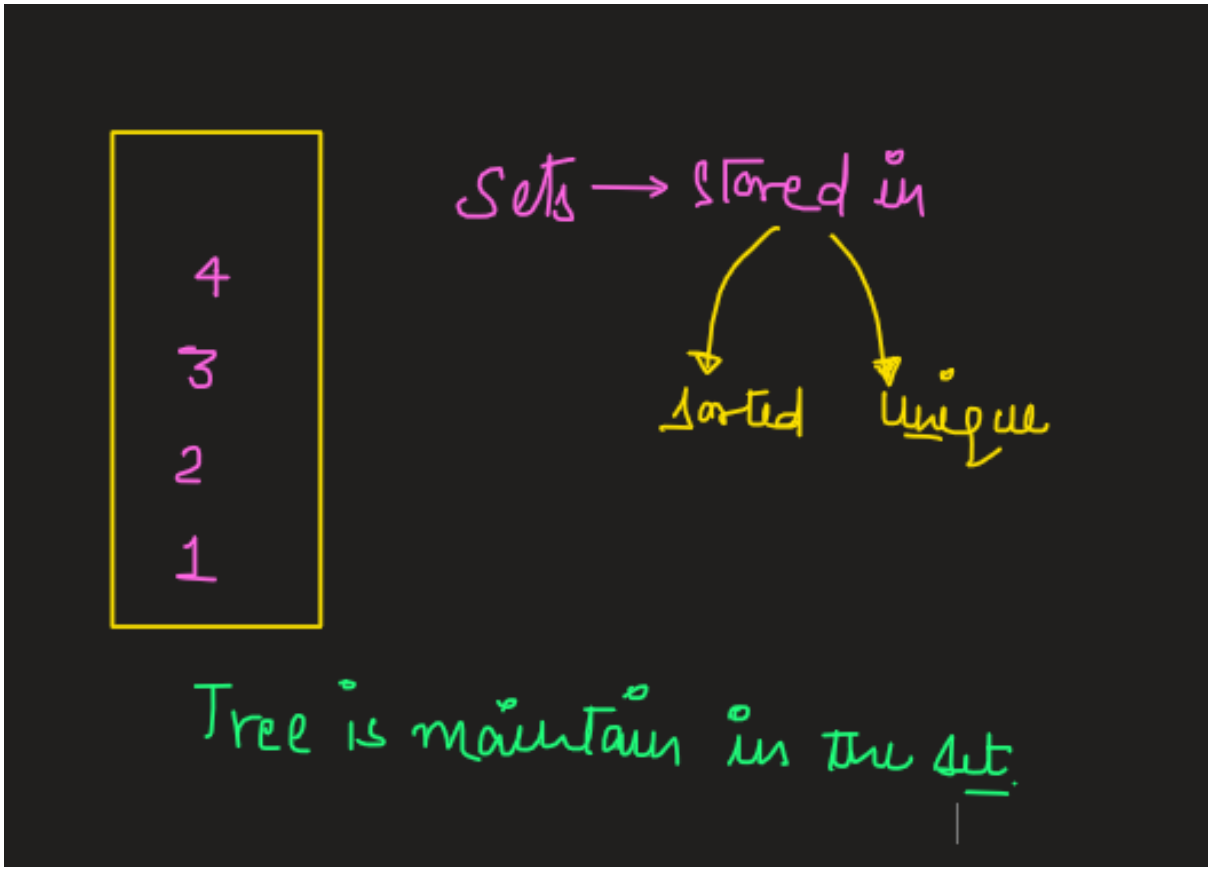
Example:

```
set<int> s;
set<string> str;
```

The Whole Code —

```
// Containers--> Sets
#include <bits/stdc++.h>
using namespace std;

// Set --> Stored in sorted order and unique
void explainSet()
```

```
{
	set<int> st;
	st.insert(1); // {1}
	st.emplace(2); //{1,2}
	st.insert(2); // {1,2}
	st.insert(4); //{1,2,4}
	st.insert(3); // {1,2,3,4}

	/*
	functionality of insert in vector can be used also,
	that only increases efficiency
	*/
	//begin(), end(), rbegin(), rend(), size(), empty() and swap() are same as those
of above

	// {1,2,3,4,5}
	auto it = st.find(3); // point to the iterator

	// {1,2,3,4,5}
	auto it = st.find(6); // return st.end()

	// {1,4,5};
	st.erase(5); // erases 5 (takes logarithmic time)

	int cnt = st.count(1); // It always 1 if the 1 is present otherwise 0

	auto it = st.find(3);
	st.erase(it); // Erase 3 fromt the array, it takes constant time

	// {1,2,3,4,5}
	auto it1 = st.find(2);
	auto it2 = st.find(4);
	st.erase(it1, it2); // after erase {1,4,5} using erase{first, last}

	// lower_bound() and upper_bound() function works in the same way

	// This is the syntax

	auto it = st.lower_bound(2);
	auto it = st.upper_bound(3);
}
int main()
{
	explainSet();
	return 0;
}

/*
Time complexity -- Logn
*/
```



Functions in set:

insert() – to insert an element in the set.

```
set<int> s;
s.insert(1);
s.insert(2);
```

begin() – return an iterator pointing to the first element in the set.

```
s.begin();
```

end() – returns an iterator to the theoretical element after the last element.

```
s.end();
```

count() – returns true or false based on whether the element is present in the set or not.

```
set<int> s;
s.insert(1);
s.insert(2);
s.count(2); //returns true
```

clear() – deletes all the elements in the set.

```
s.clear();
```

find() – to search an element in the set.

```
set<int> s;
s.insert(1);
s.insert(2);
if(s.find(2)!=s.end())
cout<<"true"<<endl;
```

erase() – to delete a single element or elements between a particular range.

```
s.erase();
```

size() – returns the size of the set.

```
s.size();
```

empty() – to check if the set is empty or not.

```
s.empty();
```

Striver’s Code

```
#include<bits/stdc++.h>

using namespace std;

int main() {
  set < int > s;
  for (int i = 1; i <= 10; i++) {
    s.insert(i);
  }

  cout << "Elements present in the set: ";
  for (auto it = s.begin(); it != s.end(); it++) {
    cout << * it << " ";
  }
  cout << endl;
  int n = 2;
  if (s.find(2) != s.end())
    cout << n << " is present in set" << endl;

  s.erase(s.begin());
  cout << "Elements after deleting the first element: ";
  for (auto it = s.begin(); it != s.end(); it++) {
    cout << * it << " ";
  }
  cout << endl;

  cout << "The size of the set is: " << s.size() << endl;

  if (s.empty() == false)
    cout << "The set is not empty " << endl;
  else
    cout << "The set is empty" << endl;
  s.clear();
  cout << "Size of the set after clearing all the elements: " << s.size();
}

Output:

Elements present in the set: 1 2 3 4 5 6 7 8 9 10
2 is present in set
Elements after deleting the first element: 2 3 4 5 6 7 8 9 10
The size of the set is: 9
The set is not empty
Size of the set after clearing all the elements: 0
```

Other functions:

- **cbegin()** – it refers to the first element of the set.
- **cend()** – it refers to the theoretical element after the last element of the set.
- **rbegin()** – it points to the last element of the set.
- **rend()** – it points to the theoretical element before the first element of the set.
- **bucket_size()** – gives the total number of elements present in a specific bucket in a set.
- **emplace()** – to insert an element in the set.
- **max_size()** – the maximum elements a set can hold.
- **max_bucket_count()** – to check the maximum number of buckets a set can hold.