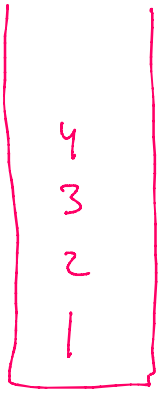
Set

15 June 2022 04:57 PM

Set : It store everything in sorted order and store unique elements.

```
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 also be used, that only increases efficiency.

```
// {1,2,3,4,5}
auto it = st.find(3);   // return an iterator which points to the '3'.
// {1,2,3,4,5}
auto it = st.find(6);   // if the element is not in the set it will always return st.end() i.e. the iterator that points to right after the
```

end.

```
// {1, 4, 5}
    st.erase(5); // erase 5 // take logarithmic time
    int cnt = st.count(1). // '1' occurence if it does'nt
                           have then '0'.

    auto it = st.find(3);
    st.erase(it); // it take 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} [first, last)
// lower_bound() and upperbound() iworks in the same
way as in vector it does.
// This the syntax
    auto it = st.lower_bound(2);
    auto it = st.upper_bound(3);
}
```

In set everything happen in $(\log N)$

Binary Search

Q: Check if X exists in sorted array or not?

Q: Check if X exists in sorted array or not?

A[] = {1, 4, 5, 8, 9}

a a+n

bool res = binary_search (a, a+n, 3); → check '3' exist or not

o/p ⇒ false

bool res = binary_search (a, a+n, 4);

o/p ⇒ True

lower_bound functions

a[] = {1, 4, 5, 6, 9, 9}

int ind = lower_bound (a, a+n, 4) - a; ⇒ -1 (Index) → to get index we do '-a'

int ind = lower_bound (a, a+n, 7) - a; ⇒ 4

int ind = lower_bound (a, a+n, 10) - a; ⇒ 6

lower bound STL returns the first occurence of the element if it occurs and if it doesn't occurs then it return the iterator pointing to the element which is the immediate next greater of the given element.

Syntax:

Starting value ↓

```
int ind = lower_bound(a.begin(), a.end(), a) - a.begin();
```

Upper Bound:

```
int ind = upper_bound(a.begin(), a.end(), a) - a.begin();
```

a[] = {1, 4, 5, 6, 9, 9}

```
int ind = upper_bound(a, a+n, 4) - a;
int ind = upper_bound(a, a+n, 7) - a;
int ind = upper_bound(a, a+n, 10) - a;
```

Q: Find the first occurrence of a X in a sorted array. If it doesn't exists, print -1.

X=4 (1)

A[] = {1, 4, 4, 4, 4, 9, 9, 10, 11}

X=2 (-1)

X=12 (-1)

```
int ind = lower_bound(a, a+n, X) - a;
if (ind != n && a[ind] == X) cout << ind;
else cout << -1;
```

X = 4 // 1

X = 2 // -1

X = 12 // -1

Q: Find the last occurrence of a X in a sorted array. If it doesn't exists, print -1.

$A[7] = \{1, 4, 4, 4, 4, 9, 9, 10, 11\}$

```
int ind = upper_bound(a, a+n, X) - a;
ind--;                                  // edge case
if (ind >= 0 && a[ind] == X) cout << ind;
else cout << -1;
```

X = 4 // 5

X = 2 // -1

X = 0 // -1

Q: Find the largest nos. smaller than X in a sorted array. If it doesn't exists, print -1.

$A[7] = \{1, 4, 4, 4, 4, 9, 9, 10, 11\}$

```
int ind = lower_bound(a, a+n, X) - a;
ind--;
if (ind >= 0) cout << a[ind]
```

else cout << -1;

---

x = 4 // 1

x = 2 // 1

x = 1 // -1