Maps (c++)and TreeMap(java)

Both map in C++ (std::map) and TreeMap in Java are sorted associative containers implemented using balanced binary search trees (Red-Black Tree). They maintain elements in sorted order of keys and provide logarithmic time complexity (O(log n)) for insertions, deletions, and lookups.

Use map (C++) / TreeMap (Java) when:

Use Case	map (C++)	TreeMap (Java)
Storing key-value pairs in sorted order	Yes	Yes
Fast lookups but with sorted keys (O(log n))	Yes	Yes
Efficient range queries (keys between X and Y)	(lower_bound(), upper_bound())	(subMap(), ceilingKey(), floorKey())
Finding the smallest/largest key	(begin(), rbegin())	(firstKey(), lastKey())
Finding closest greater/smaller key	(lower_bound(), upper_bound())	(ceilingKey(), floorKey())
Custom comparator sorting	yes	yes

Types of Questions Solved by map (C++) and TreeMap (Java)

Problem Type	**Why map/TreeMap?**	**Example**
Find Smallest/Largest Key	TreeMap.firstKey(), TreeMap.lastKey() in Java or map.begin() in C++	Find the lowest price in a stock order book
Efficient Range Queries	TreeMap.subMap(start, end), map.lower_bound(x) in C++	Find all sales between two timestamps
Find Closest Greater/Smaller Element	TreeMap.ceilingKey(x), TreeMap.floorKey(x), map.upper_bound(x) in C++	Given a deadline, find the next available event
Auto-Sorted Leaderboard	Maintains order automatically	Ranking system where players are sorted by scores
Schedule Conflicts (Meeting Room)	TreeMap helps in finding overlapping intervals	Check if a new meeting overlaps with existing ones
K-th Smallest/Largest Element	Elements are stored in sorted order	Find the K-th earliest event

QDisplaying contents of map

```
#include <iostream>
#include <map>

int main() {
    map<int, string> student;


    student[101] = "Alice";
    student[103] = "Bob";
    student[102] = "Charlie";


    // Iterating over the map (sorted order)
    for (const auto &entry : student) {
        cout << entry.first << " -> " << entry.second << endl;
    }

    return 0;
}
```

Output**:**

```
101 -> Alice
102 -> Charlie
103 -> Bob
```

```
import java.util.Map;
import java.util.TreeMap;

public class TreeMapExample {
    public static void main(String[] args) {
        Map<Integer, String> student = new TreeMap<>();

        student.put(101, "Alice");
        student.put(103, "Bob");
        student.put(102, "Charlie");

        // Iterating in sorted order
        for (Map.Entry<Integer, String> entry : student.entrySet()) {
            System.out.println(entry.getKey() + " -> " + entry.getValue());
        }
    }
}
```

Output (Sorted Order by Key):

```
101 -> Alice
102 -> Charlie
103 -> Bob
```

Q code for lookup

```
#include <iostream>
#include <map>

using namespace std;

int main() {
    // Create a map with key-value pairs
    map<int, string> studentMap;
    studentMap[101] = "Alice";
    studentMap[102] = "Bob";
    studentMap[103] = "Charlie";

    // Lookup for a key
    int key = 102;
    if (studentMap.find(key) != studentMap.end()) {
        cout << "Student ID " << key << " is " << studentMap[key] << endl;
    } else {
        cout << "Student ID " << key << " not found." << endl;
    }

    return 0;
}
```

Java

```
import java.util.TreeMap;

public class Main {
    public static void main(String[] args) {
        // Create a TreeMap with key-value pairs
        TreeMap<Integer, String> studentMap = new TreeMap<>();
        studentMap.put(101, "Alice");
        studentMap.put(102, "Bob");
        studentMap.put(103, "Charlie");

        // Lookup for a key
        int key = 102;
        if (studentMap.containsKey(key)) {
            System.out.println("Student ID " + key + " is " + studentMap.get(key));
        } else {
            System.out.println("Student ID " + key + " not found.");
        }
    }
}
```

Finding the Closest Greater/Smaller Key

C++ (using map)

```
#include <iostream>
#include <map>
using namespace std;


int main() {
  map<int, string> events;
  events[10] = "Meeting A";
  events[20] = "Meeting B";
  events[30] = "Meeting C";


  int query = 15;
  auto it = events.lower_bound(query); // First key >= query
  if (it != events.end())
    cout << "Next event: " << it->second << " at " << it->first << endl;

  return 0;
}
```

Java (using TreeMap)

```
import java.util.TreeMap;

public class Main {
  public static void main(String[] args) {
    TreeMap<Integer, String> events = new TreeMap<>();
    events.put(10, "Meeting A");
    events.put(20, "Meeting B");
    events.put(30, "Meeting C");

    int query = 15;
    Integer nextEvent = events.ceilingKey(query);
    if (nextEvent != null)
      System.out.println("Next event: " + events.get(nextEvent) + " at " + nextEvent);
  }
}
```

Output: **Next event: Meeting B at 20**

Range Queries

```
#include <map>
using namespace std;

int main() {
   map<int, string> orders;
   orders[100] = "Order 1";
   orders[200] = "Order 2";
   orders[300] = "Order 3";

   auto low = orders.lower_bound(150);
   auto high = orders.upper_bound(250);

   cout << "Orders between 150 and 250:" << endl;
   for (auto it = low; it != high; ++it)
      cout << it->first << ": " << it->second << endl;

   return 0;
}
```

Java

```
import java.util.*;

public class Main {
   public static void main(String[] args) {
      TreeMap<Integer, String> orders = new TreeMap<>();
      orders.put(100, "Order 1");
      orders.put(200, "Order 2");
      orders.put(300, "Order 3");

      NavigableMap<Integer, String> subOrders = orders.subMap(150, true, 250, true);
      System.out.println("Orders between 150 and 250: " + subOrders);
   }
}
```

Q Find k smallest

```
#include <iostream>
#include <map>

using namespace std;

int findKthSmallest(map<int, string>& studentMap, int K) {
   if (K <= 0 || K > studentMap.size()) {
      cout << "Invalid K value!" << endl;
      return -1;
   }
```

```
    auto it = studentMap.begin();
    advance(it, K - 1); // Move iterator to the K-th element (0-based index)

    cout << "K-th Smallest Element: " << it->first << " -> " << it->second << endl;
    return it->first;
}

int main() {
    map<int, string> studentMap = {
        {101, "Alice"},
        {102, "Bob"},
        {103, "Charlie"},
        {104, "David"},
        {105, "Eve"}
    };

    int K = 3;
    findKthSmallest(studentMap, K);

    return 0;
}
```

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        TreeMap<Integer, String> studentMap = new TreeMap<>();
        studentMap.put(101, "Alice");
        studentMap.put(102, "Bob");
        studentMap.put(103, "Charlie");
        studentMap.put(104, "David");
        studentMap.put(105, "Eve");

        int K = 3;
        findKthSmallest(studentMap, K);
    }

    public static void findKthSmallest(TreeMap<Integer, String> map, int K) {
        if (K <= 0 || K > map.size()) {
            System.out.println("Invalid K value!");
            return;
        }

        List<Map.Entry<Integer, String>> entryList = new ArrayList<>(map.entrySet());
        Map.Entry<Integer, String> kthElement = entryList.get(K - 1);
        System.out.println("K-th Smallest Element: " + kthElement.getKey() + " -> " +
kthElement.getValue());
    }
}
```

Kth largest element

```
#include <iostream>
#include <map>

using namespace std;

int findKthLargest(map<int, string>& studentMap, int K) {
    if (K <= 0 || K > studentMap.size()) {
        cout << "Invalid K value!" << endl;
        return -1;
    }

    auto it = studentMap.rbegin(); // Reverse iterator (largest element first)
    advance(it, K - 1); // Move iterator to the K-th largest element

    cout << "K-th Largest Element: " << it->first << " -> " << it->second << endl;
    return it->first;
}

int main() {
    map<int, string> studentMap = {
        {101, "Alice"},
        {102, "Bob"},
        {103, "Charlie"},
        {104, "David"},
        {105, "Eve"}
    };

    int K = 2;
    findKthLargest(studentMap, K);

    return 0;
}

import java.util.*;

public class Main {
    public static void main(String[] args) {
        TreeMap<Integer, String> studentMap = new TreeMap<>();
        studentMap.put(101, "Alice");
        studentMap.put(102, "Bob");
        studentMap.put(103, "Charlie");
        studentMap.put(104, "David");
        studentMap.put(105, "Eve");

        int K = 2;
        findKthLargest(studentMap, K);
    }

    public static void findKthLargest(TreeMap<Integer, String> map, int K) {
        if (K <= 0 || K > map.size()) {
            System.out.println("Invalid K value!");
            return;
        }

        List<Integer> keys = new ArrayList<>(map.descendingKeySet()); // Reverse order keys
        int kthLargestKey = keys.get(K - 1);

        System.out.println("K-th Largest Element: " + kthLargestKey + " -> " + map.get(kthLargestKey));
    }
}
```