Lecture 17

TreeMap and TreeSet

Last time:

We have looked at two different usages of binary trees:

- Binary Search Tree
- Huffman Coding (Data Compression)

Having a frequency table of characters and the algorithm to build Huffman tree, we can build a Huffman tree.

And, with the tree, we can find corresponding codes for each character.

In comparison to the fixed width coding, it shows that it is possible to have less number of bits for the same length of characters.

Today, let's look at the Java Collections Framework to see two classes that are based on tree data structure: TreeMap and TreeSet.

TreeMap

The TreeMap is one of the general-purpose Map implementations. It is based on Red-Black tree, which means the TreeMap is guaranteed to be balanced. It implements the NavigableMap interface.

As we discussed in previous lectures, the HashMap offers the best alternative for inserting, deleting and searching elements.

But, HashMap has one big limitation! It does not allow us to traverse the elements in any sorted order.

That's where TreeMap comes in handy!

Let's look at its usages.
Remember the following code from Hash in Java lecture?

```
Map<String, Integer> freqOfWords = new HashMap<String,
Integer>();
String[] words = "coming together is a beginning keeping together
is progress working together is success".split(" ");

for(String word: words) {
      Integer frequency = freqOfWords.get(word);
      if(frequency == null) {
            frequency = 1;
      } else {
            frequency++;
      }
      freqOfWords.put(word, frequency);
}
```

How do we print all of the keys in a sorted order?

```
TreeMap<String, Integer> sortedWords = new TreeMap<String,
Integer>(freqOfWords);
System.out.println(sortedWords);
```

How about descending order?

```

System.out.println(__________________________________________);
```

In comparison to HashMap, the TreeMap provides many other methods. The following is the list of the important methods:

- ceilingKey(key)
- floorKey(key)
- higherKey(key)
- lowerKey(key)
- pollFirstEntry()
- pollLastEntry()
- firstEntry()
- lastEntry()
- firstKey()
- lastKey()
- descendingMap()

TreeSet
The TreeSet is one of the general-purpose Set implementations. It implements the NavigableSet interface and based on TreeMap.

Note!

It is generally faster to add elements to a HashSet. And, if you need to traverse the elements in sorted order, convert the collection to a TreeSet.

Once again, a TreeSet is always balanced which means it guarantees ______________ running time complexity for insertion, deletion and search.

```
Set<String> distinctWords = new HashSet<String>();

String[] words = "coming together is a beginning keeping together
is progress working together is success".split(" ");

for(String word: words) distinctWords.add(word);

System.out.println(distinctWords.size()+" distinct words.");
System.out.println("They are: "+distinctWords);

TreeSet<String> sortedWords = new TreeSet<String>(distinctWords);
System.out.println(sortedWords);
```

TreeSet also provides many useful methods:

- ceiling(e)
- floor(e)
- higher(e)
- lower(e)
- pollFirst()
- pollLast()
- first()
- last()
- headSet(e)
- tailSet(e)
- descendingSet()

You should check the API document for TreeMap and TreeSet to check other available methods.

The other thing you have to remember is that **all of the elements in TreeSet and TreeMap must be sortable.**

That means, when you create your own class, you need to properly implement the Comparable interface or the Comparator interface.

Example application using TreeSet

Suppose you own a bus transportation company and the major route your company operates is between Pittsburgh and New York City.

The company has its own website where customers can find the schedule.

You found out that customers want to have the following lookups from the schedule.

- What is the last bus that leaves Pittsburgh before 4 pm?
- What is the first bus that leaves Pittsburgh after 10 pm?

What data structure would you use and how would you implement?

For now, to make it simpler, we assume that all of the times are in military time. (For example, 4 pm is 1600)

```
// set up the daily bus schedule
TreeSet<Integer> schedule = new TreeSet<Integer>();
schedule.add(1223);
schedule.add(1430);
schedule.add(1545);
schedule.add(1610);
schedule.add(1705);
schedule.add(2010);
schedule.add(2215);
schedule.add(2320);
schedule.add(2430);

// show the last bus that leaves PGH before 4 pm
System.out.println(schedule.___________________________________);

// show the first bus that leaves PGH after 10 pm
System.out.println(schedule.___________________________________);
```