TreeMaps

Prepared by Mahdi Ghamkhari

Map

AK ----- > 16

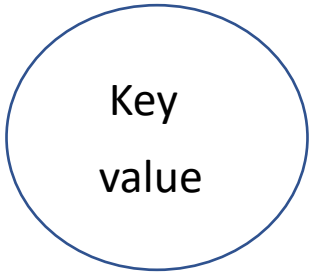
FL ----- > 13

MI ----- > 20

HI ----- > 43

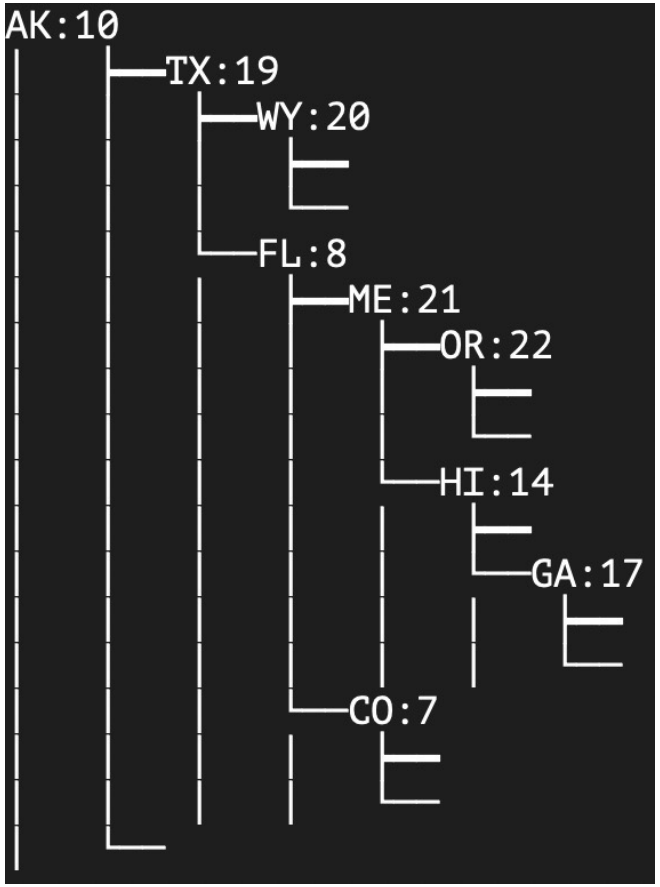
TreeMap

- A TreeMap is a Binary Search Tree (BST)
- Each node has these two variables: Key, value



- Sometimes value is referred to as data

TreeMap



TreeMap

- The add() method of the TreeMap uses the same logic we studied for insertion in BSTs
- The delete() method of the TreeMap uses the same logic we studied for deletion in BSTs
- The doMapping() method of the TreeMap uses the same logic we studied for searching in BSTs

TreeMap

- The add method of the TreeMap place a new node in the TreeMap by comparing the key variable of the new node with key variables of the existing nodes
- Keys could be of type String, int, double, float
- For the case of String keys, Java does not have > and < operators to compare such keys
- We have to develop a function to compare String Keys

Comparing String Keys

```
public boolean IsGreaterThan(String key1, String key2)

{
    if (LetterToNumber(key1.charAt(0))>LetterToNumber(key2.charAt(0)))
        return true;

    if (LetterToNumber(key1.charAt(0))<LetterToNumber(key2.charAt(0)))
        return false;

    if (LetterToNumber(key1.charAt(1))>LetterToNumber(key2.charAt(1)))
        return true;

    if (LetterToNumber(key1.charAt(1))<LetterToNumber(key2.charAt(1)))
        return false;

    return false;
}
```

Time Complexity

- Adding, deleting and searching nodes in TreeMaps take place in O(log(n)).

- For HashMaps Adding, deleting and searching have a time complexity of O(1), as long as the Hash Function is ideal.