TreeMap Internal Implementation

```
public class TreeMap<K,V>
extends AbstractMap<K,V>
implements NavigableMap<K,V>, Cloneable, Serializable
```

- TreeMap is Red-Black tree based NavigableMap implementation
- TreeMap is sorted according to the natural ordering of its keys, or by a Comparator provided at map creation time, depending on which constructor is used
- TreeMap provides guaranteed log(n) time cost for the containsKey, get, put and remove operations
- TreeMap is not **synchronized**
- The iterators returned by the iterator method of the collectionsare fail-fast. i.e if the map is structurally modified at any time after the iterator is created thenit will throw a `ConcurrentModificationException`
- For Thread Safe: SortedMapm = Collections.synchronizedSortedMap(new TreeMap(...));

OverloadedTreeMap Constructors:

TreeMap()

- Constructs a new, empty tree map, using the natural ordering of its keys.All keys inserted into the map must implement the Comparable interface
- All such keys must be mutually comparable: k1.compareTo(k2) must not throw a ClassCastException for any keys k1 and k2 in the map.

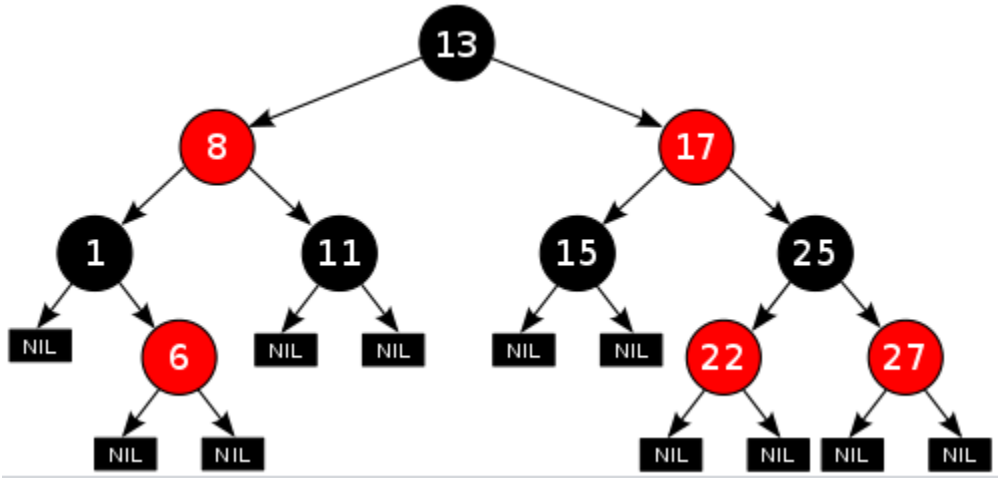
TreeMap(Comparator<? super K> comparator)

- Constructs a new, empty tree map, ordered according to the given comparator.
- All keys inserted into the map must be *mutually comparable* by the given comparator: comparator.compare(k1, k2) must not throw a ClassCastException for any keys k1 and k2 in the map

TreeMap(Map<? extends K,? extends V> m) - Constructs a new tree map containing the same mappings as the given map, ordered according to the natural ordering of its keys.

TreeMap(SortedMap<K,? extends V> m) - Constructs a new tree map containing the same mappings and using the same ordering as the specified sorted map.

Red-Black Tree Implementation:

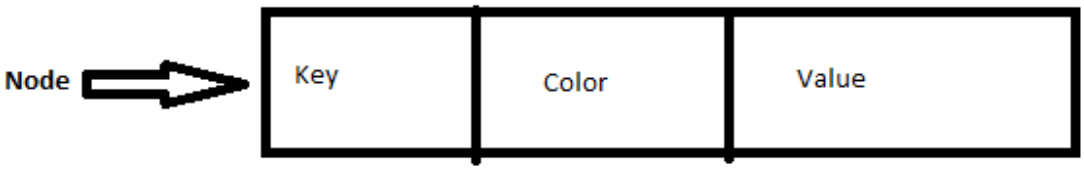


Source: Wiki

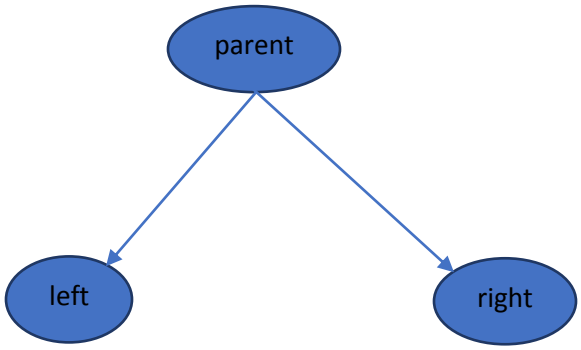
- Each node is either red or black.
- The root is black. This rule is sometimes omitted. Since the root can always be changed from red to black, but not necessarily vice versa, this rule has little effect on analysis.
- All leaves (NIL) are black.
- If a node is red, then both its children are black.
- Every path from a given node to any of its descendant NIL nodes contains the same number of black nodes.

TreeMap Red-Black Tree Implementation:

Each Node in TreeMap contains three elements i.e. key, value and color



Each Node can have three references i.e. parent, left and right elements



- The left element will always be logically less than the parent element.
- The right element will always be logically greater than OR equal to a parent element
- The logical comparison of Objects is done by natural order i.e. those object who implement Comparable interface and override compareTo(Object obj) method. Based on the return value

HashMap vs. TreeMap

HashMap	**TreeMap**
HashMap which stores key- value pairs	TreeMap which stores key- value pairs and sorted according to the natural ordering of its keys
HashMap uses hashing algorithm implementation.	TreeMap is a Red-Black tree based NavigableMap implementation. i.e. it sorts the TreeMap object keys using Red-Black tree algorithm.
HashMap implementation provides constant-time performance O(1) for get(), put() operation by assuming the hash function disperses the elements properly among the buckets.	TreeMap implementation provides guaranteed log(n) time cost for the containsKey,get,put and remove operations.
HashMap allows one null value as key	TreeMap does not allow null key but values can be null
HashMap does not provide methods for navigable map	TreeMap provides navigableMap operations like firstEntry,lastEntry, submap etc..
HashMap is fast	TreeMap is slow due to default sort in nature
HashMap is suitable if there is no requirement of sorting and requires one null key	TreeMap is suitable if you have requirement for sorting, find first or last entry, sub map etc.
HashMap constructor takes bucket size	TreeMap does not have constructor to specify the size of the elements.
HashMap fail fast in nature and throw ConcurrentModificationException	TreeMap fail-fast in nature and throw ConcurrentModificationException

If user wants to insert user defined object as a key in HashMap then class must override equals and hashcode method	`TreeMap():`
	All keys inserted into the map must implement the Comparable interface. Furthermore, all such keys must be mutually comparable: k1.compareTo(k2) must not throw a ClassCastException for any keys k1 and k2 in the map
	
	`TreeMap(Comparator<? super K> comparator):`
	
	All keys inserted into the map must be mutually comparable by the given comparator: comparator.compare(k1, k2) must not throw a ClassCastException for any keys k1 and k2 in the map