Alexandria University,

Faculty of engineering,

Data structures.

Assignment 2.

Name: Mohamed Mohamed Abdlhakem. (43)

Name: Mohamed Salah Osman. (41)

Problem statement:

Red Black Tree

A red black tree is a kind of self-balancing binary search tree in computer science. Each node of the binary tree has an extra bit, and that bit is often interpreted as the color (red or black) of the node. These color bits are used to ensure the tree remains approximately balanced during insertions and deletions. Balance is preserved by painting each node of the tree with one of two colors in a way that satisfies certain properties, which collectively constrain how unbalanced the tree can become in the worst case. When the tree is modified, the new tree is subsequently rearranged and repainted to restore the coloring properties. The properties are designed in such a way that this rearranging and recoloring can be performed efficiently.

Tree Map

A Red-Black tree based Navigable Map implementation. The map 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. This implementation provides guaranteed log (n) time cost for contains Key, get, put and remove operations.

Data structure used:

- Stack: implement DFS algorithm.
- Queue: implement BFS algorithm.
- Linked hash set: to preserve the insertion order of keys in tree map key set function.
- Linked list.

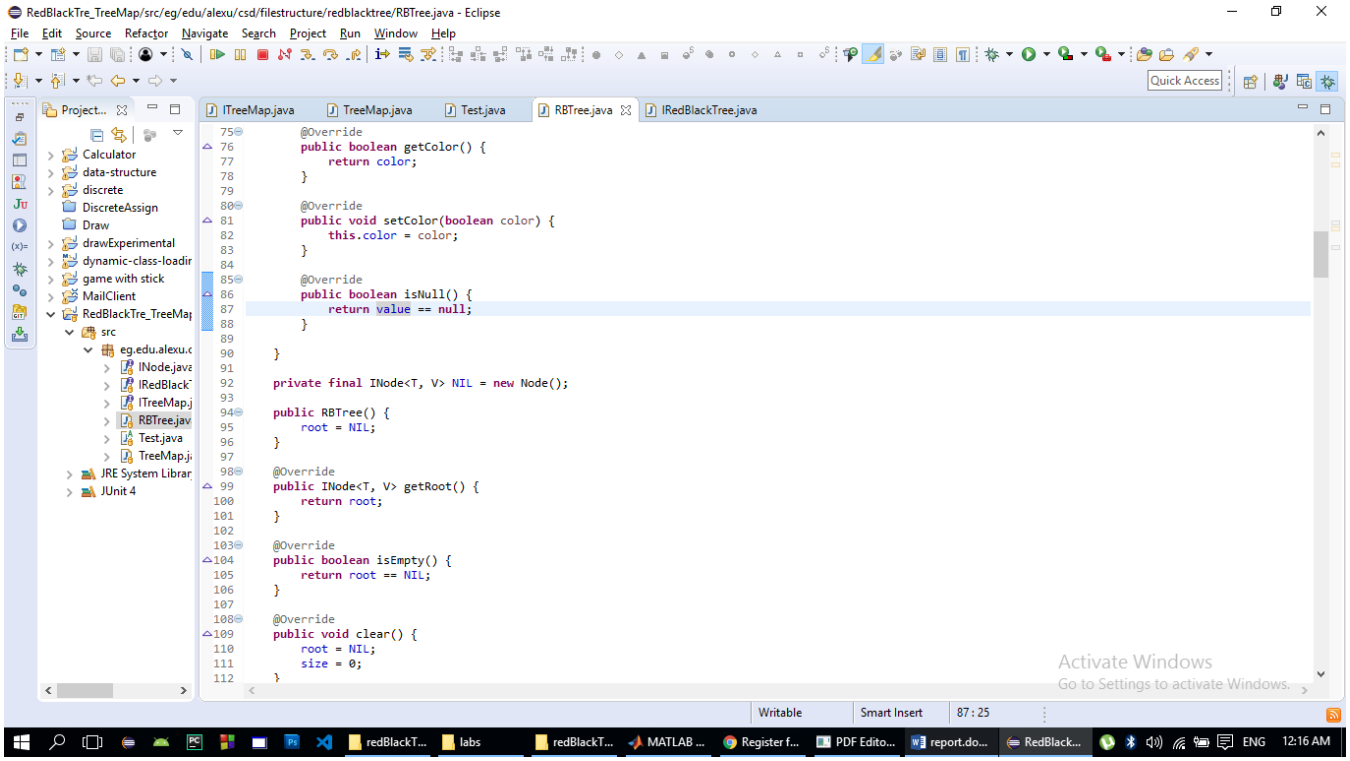
Code snippets:

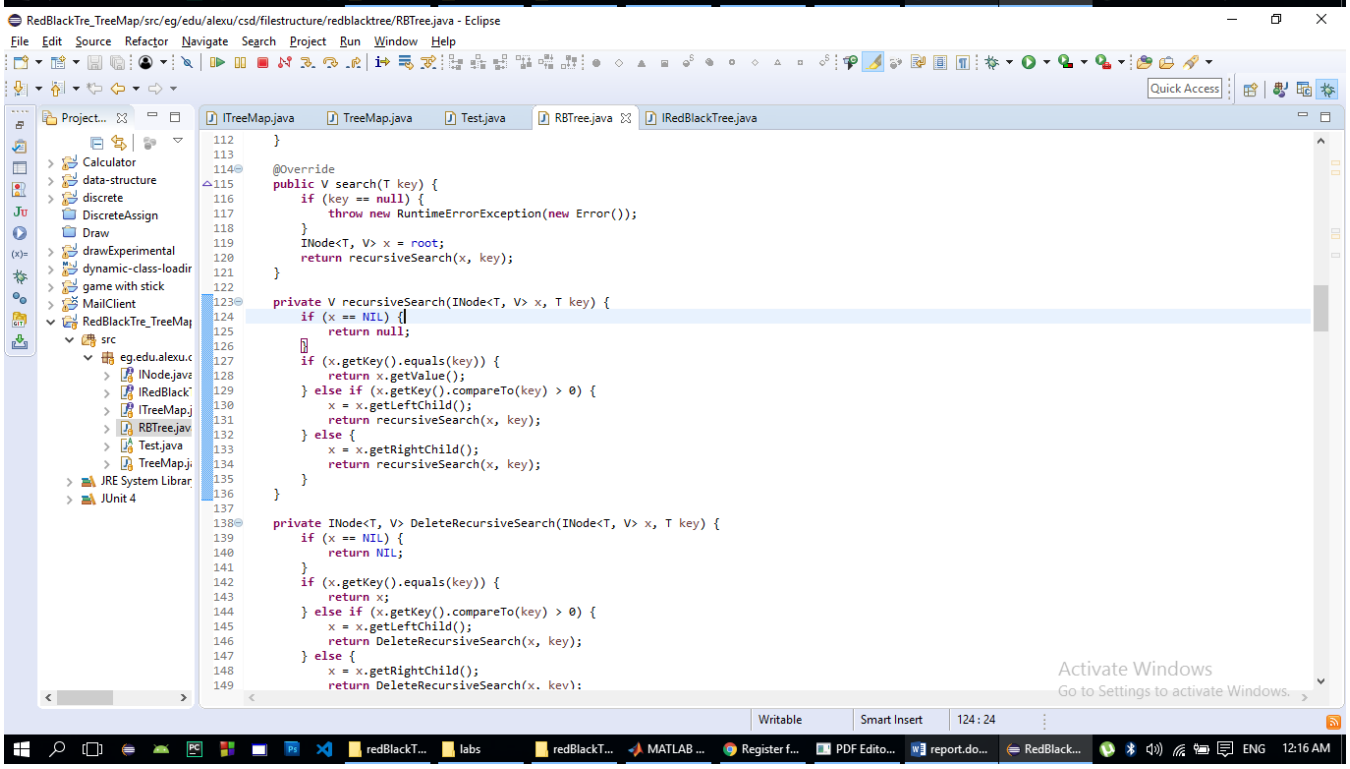
RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse

```java
package eg.edu.alexu.csd.filestructure.redblacktree;

import javax.management.RuntimeErrorException;

public class RBTree<T extends Comparable<T>, V> implements IRedBlackTree<T, V> {
    private INode<T, V> root;
    private int size = 0;
    class Node implements INode<T, V> {
        private INode<T, V> parent;
        private INode<T, V> l_child;
        private INode<T, V> r_child;
        private T key;
        private V value;
        private boolean color;

        public Node() {
            parent = NIL;
            l_child = NIL;
            r_child = NIL;
            value = null;
            key = null;
            color = INode.BLACK;
        }

        @Override
        public void setParent(INode<T, V> parent) {
            this.parent = parent;
        }

        @Override
        public INode<T, V> getParent() {
            return parent;
        }

        @Override
        public void setLeftChild(INode<T, V> leftChild) {
            this.l_child = leftChild;
        }
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse

```java
        }

        @Override
        public INode<T, V> getLeftChild() {
            return l_child;
        }

        @Override
        public void setRightChild(INode<T, V> rightChild) {
            this.r_child = rightChild;
        }

        @Override
        public INode<T, V> getRightChild() {
            return r_child;
        }

        @Override
        public T getKey() {
            return key;
        }

        @Override
        public void setKey(T key) {
            this.key = key;
        }

        @Override
        public V getValue() {
            return value;
        }

        @Override
        public void setValue(V value) {
            this.value = value;
        }

        @Override
```


RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project...
Calculator
data-structure
discrete
DiscreteAssign
Draw
drawExperimental
dynamic-class-loadir
game with stick
MailClient
RedBlackTre_TreeMap
src
eg.edu.alexu.c
INode.java
IRedBlack
ITreeMap.j
RBTree.jav
Test.java
TreeMap.j
JRE System Librar
JUnit 4
ITreeMap.java
TreeMap.java
Test.java
RBTree.java
IRedBlackTree.java
@Override
public boolean getColor() {
return color;
}
@Override
public void setColor(boolean color) {
this.color = color;
}
@Override
public boolean isNull() {
return value == null;
}
}
private final INode<T, V> NIL = new Node();
public RBTree() {
root = NIL;
}
@Override
public INode<T, V> getRoot() {
return root;
}
@Override
public boolean isEmpty() {
return root == NIL;
}
@Override
public void clear() {
root = NIL;
size = 0;
Activate Windows
Go to Settings to activate Windows.
Writable
Smart Insert
87 : 25
ENG
12:16 AM


RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project...
Calculator
data-structure
discrete
DiscreteAssign
Draw
drawExperimental
dynamic-class-loadir
game with stick
MailClient
RedBlackTre_TreeMap
src
eg.edu.alexu.c
INode.java
IRedBlack
ITreeMap.j
RBTree.jav
Test.java
TreeMap.j
JRE System Librar
JUnit 4
ITreeMap.java
TreeMap.java
Test.java
RBTree.java
IRedBlackTree.java
}
@Override
public V search(T key) {
if (key == null) {
throw new RuntimeErrorException(new Error());
}
INode<T, V> x = root;
return recursiveSearch(x, key);
}
private V recursiveSearch(INode<T, V> x, T key) {
if (x == NIL) {
return null;
}
if (x.getKey().equals(key)) {
return x.getValue();
} else if (x.getKey().compareTo(key) > 0) {
x = x.getLeftChild();
return recursiveSearch(x, key);
} else {
x = x.getRightChild();
return recursiveSearch(x, key);
}
}
private INode<T, V> DeleteRecursiveSearch(INode<T, V> x, T key) {
if (x == NIL) {
return NIL;
}
if (x.getKey().equals(key)) {
return x;
} else if (x.getKey().compareTo(key) > 0) {
x = x.getLeftChild();
return DeleteRecursiveSearch(x, key);
} else {
x = x.getRightChild();
Activate Windows
Go to Settings to activate Windows.
Writable
Smart Insert
124 : 24
ENG
12:16 AM

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse

```
149             return DeleteRecursiveSearch(x, key);
150         }
151     }
152
153     @Override
154     public boolean contains(T key) {
155         if (key == null) {
156             throw new RuntimeErrorException(new Error());
157         }
158         V v = recursiveSearch(root, key);
159         return v != null;
160     }
161
162     @Override
163     public void insert(T key, V value) {
164         if(key == null || value == null) {
165             throw new RuntimeErrorException(new Error());
166         }
167         INode<T, V> y = NIL;
168         INode<T, V> x = root;
169         while (x != NIL) {
170             y = x;
171             if (key.compareTo(x.getKey()) < 0) {
172                 x = x.getLeftChild();
173             } else if (key.compareTo(x.getKey()) > 0) {
174                 x = x.getRightChild();
175             } else if (key.compareTo(x.getKey()) == 0) {
176                 x.setValue(value);
177                 return;
178             }
179
180         }
181         size++;
182         INode<T, V> z = new Node();
183         z.setKey(key);
184         z.setValue(value);
185         z.setParent(y);
186         if (y == NIL) {
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse

```
186         if (y == NIL) {
187             root = z;
188
189         } else if (key.compareTo(y.getKey()) < 0) {
190             y.setLeftChild(z);
191         } else {
192             y.setRightChild(z);
193         }
194         z.setLeftChild(NIL);
195         z.setRightChild(NIL);
196         z.setColor(INode.RED);
197         insertFixup(z);
198     }
199
200     private void leftRotate(INode<T, V> x) {
201         if (x == NIL || x.getRightChild() == NIL) {
202             return;
203         }
204         INode<T, V> y = x.getRightChild();
205         x.setRightChild(y.getLeftChild());
206         if (y.getLeftChild() != NIL) {
207             y.getLeftChild().setParent(x);
208         }
209         y.setParent(x.getParent());
210         if (x.getParent() == NIL) {
211             root = y;
212         } else if (x == x.getParent().getLeftChild()) {
213             x.getParent().setLeftChild(y);
214         } else {
215             x.getParent().setRightChild(y);
216         }
217         y.setLeftChild(x);
218         x.setParent(y);
219     }
220
221     private void rightRotate(INode<T, V> x) {
222         if (x == NIL || x.getLeftChild() == NIL) {
223             return;
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse

```java
            return;
        }
        INode<T, V> y = x.getLeftChild();
        x.setLeftChild(y.getRightChild());
        if (y.getRightChild() != NIL) {
            y.getRightChild().setParent(x);
        }
        y.setParent(x.getParent());
        if (x.getParent() == NIL) {
            root = y;
        } else if (x == x.getParent().getLeftChild()) {
            x.getParent().setLeftChild(y);
        } else {
            x.getParent().setRightChild(y);
        }
        y.setRightChild(x);
        x.setParent(y);
    }

    private void insertFixup(INode<T, V> z) {
        if (root == z) {
            root.setColor(INode.BLACK);
            return;
        } else if (z.getParent().getColor() == INode.RED) {
            INode<T, V> uncle;
            if (z.getParent() == z.getParent().getParent().getLeftChild()) {
                uncle = z.getParent().getParent().getRightChild();
            } else {
                uncle = z.getParent().getParent().getLeftChild();
            }
            if (uncle.getColor() == INode.RED) {
                z.getParent().setColor(INode.BLACK);
                uncle.setColor(INode.BLACK);
                z.getParent().getParent().setColor(INode.RED);
                z = z.getParent().getParent();
                insertFixup(z);
            } else {
                if (z == z.getParent().getLeftChild() && z.getParent() == z.getParent().getParent().getLeftChild()) {
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse

```java
                if (z == z.getParent().getLeftChild() && z.getParent() == z.getParent().getParent().getLeftChild()) {
                    INode<T, V> g = z.getParent().getParent();
                    INode<T, V> p = z.getParent();
                    rightRotate(z.getParent().getParent());
                    g.setColor(INode.RED);
                    p.setColor(INode.BLACK);

                } else if (z == z.getParent().getRightChild()
                        && z.getParent() == z.getParent().getParent().getLeftChild()) {
                    INode<T, V> p = z.getParent();
                    leftRotate(p);
                    INode<T, V> g = z.getParent();
                    rightRotate(g);
                    g.setColor(INode.RED);
                    z.setColor(INode.BLACK);

                } else if (z == z.getParent().getLeftChild()
                        && z.getParent() == z.getParent().getParent().getRightChild()) {
                    INode<T, V> p = z.getParent();
                    rightRotate(p);
                    INode<T, V> g = z.getParent();
                    leftRotate(g);
                    g.setColor(INode.RED);
                    z.setColor(INode.BLACK);

                } else {
                    INode<T, V> g = z.getParent().getParent();
                    INode<T, V> p = z.getParent();
                    leftRotate(g);
                    g.setColor(INode.RED);
                    p.setColor(INode.BLACK);

                }
            }
        }

    }
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse

```java
297
298     @Override
299     public boolean delete(T key) {
300         if (key == null) {
301             throw new RuntimeErrorException(new Error());
302         }
303         INode<T, V> z = DeleteRecursiveSearch(root, key);
304         if (z == NIL) {
305             return false;
306         }
307         INode<T, V> y = z;
308         boolean yOriginColor = y.getColor();
309         INode<T, V> x;
310         if (z.getLeftChild() == NIL) {
311             x = z.getRightChild();
312             Transplant(z, z.getRightChild());
313         } else if (z.getRightChild() == NIL) {
314             x = z.getLeftChild();
315             Transplant(z, z.getLeftChild());
316         } else {
317             y = minimum(z.getRightChild());
318             yOriginColor = y.getColor();
319             x = y.getRightChild();
320             if (y.getParent() == z) {
321                 x.setParent(y);
322             } else {
323                 Transplant(y, y.getRightChild());
324                 y.setRightChild(z.getRightChild());
325                 y.getRightChild().setParent(y);
326             }
327             Transplant(z, y);
328             y.setLeftChild(z.getLeftChild());
329             y.getLeftChild().setParent(y);
330             y.setColor(z.getColor());
331         }
332         if (yOriginColor == INode.BLACK) {
333             deleteFixUp(x);
334         }
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse

```java
334         }
335         size--;
336         return true;
337     }
338
339     private void deleteFixUp(INode<T, V> x) {
340         while (x != root && x.getColor() == INode.BLACK) {
341             if (x == x.getParent().getLeftChild()) {
342                 INode<T, V> w = x.getParent().getRightChild();
343                 if (w.getColor() == INode.RED) {
344                     w.setColor(INode.BLACK); // case1
345                     x.getParent().setColor(INode.RED); // case1
346                     leftRotate(x.getParent());// case1
347                     w = x.getParent().getRightChild();// case1
348                 }
349                 if (w.getLeftChild().getColor() == INode.BLACK && w.getRightChild().getColor() == INode.BLACK) {
350                     w.setColor(INode.RED);
351                     x = x.getParent();
352                     continue;
353                 } else if (w.getRightChild().getColor() == INode.BLACK) {
354                     w.getLeftChild().setColor(INode.BLACK);
355                     w.setColor(INode.RED);
356                     rightRotate(w);
357                     w = x.getParent().getRightChild();
358                 }
359                 if (w.getRightChild().getColor() == INode.RED) {
360                     w.setColor(x.getParent().getColor());
361                     x.getParent().setColor(INode.BLACK);
362                     w.getRightChild().setColor(INode.BLACK);
363                     leftRotate(x.getParent());
364                     x = root;
365                 }
366             } else {
367                 INode<T, V> w = x.getParent().getLeftChild();
368                 if (w.getColor() == INode.RED) {
369                     w.setColor(INode.BLACK); // case1
370                     x.getParent().setColor(INode.RED); // case1
371                     rightRotate(x.getParent());// case1
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse

```java
                    rightRotate(x.getParent());// case1
                    w = x.getParent().getLeftChild();// case1
                }
                if (w.getRightChild().getColor() == INode.BLACK && w.getLeftChild().getColor() == INode.BLACK) {
                    w.setColor(INode.RED);
                    x = x.getParent();
                    continue;
                } else if (w.getLeftChild().getColor() == INode.BLACK) {
                    w.getRightChild().setColor(INode.BLACK);
                    w.setColor(INode.RED);
                    leftRotate(w);
                    w = x.getParent().getLeftChild();
                }
                if (w.getLeftChild().getColor() == INode.RED) {
                    w.setColor(x.getParent().getColor());
                    x.getParent().setColor(INode.BLACK);
                    w.getLeftChild().setColor(INode.BLACK);
                    rightRotate(x.getParent());
                    x = root;
                }
            }
        }
        x.setColor(INode.BLACK);
    }

    /**
     * @param x
     */



    private INode<T, V> minimum(INode<T, V> x) {
        while (x.getLeftChild() != NIL) {
            x = x.getLeftChild();
        }
        return x;
    }
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/RBTree.java - Eclipse

```java
            }
        }
        x.setColor(INode.BLACK);
    }

    /**
     * @param x
     */



    private INode<T, V> minimum(INode<T, V> x) {
        while (x.getLeftChild() != NIL) {
            x = x.getLeftChild();
        }
        return x;
    }

    private void Transplant(INode<T, V> u, INode<T, V> v) {
        if (u.getParent() == NIL) {
            root = v;
        } else if (u == u.getParent().getLeftChild()) {
            u.getParent().setLeftChild(v);
        } else {
            u.getParent().setRightChild(v);
        }
        v.setParent(u.getParent());
    }
    public INode<T, V> getNILNode(){
        return NIL;
    }
    public int size(){
        return size;
    }

}
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/TreeMap.java - Eclipse

```java
package eg.edu.alexu.csd.filestructure.redblacktree;

import java.util.AbstractMap;

public class TreeMap<T extends Comparable<T>, V> implements ITreeMap<T, V> {
    private RBTree<T, V> rb;
    private INode<T, V> NIL;
    class mapEntry implements Map.Entry<T, V> {
        private T key;
        private V value;

        public mapEntry(T key, V value) {
            this.key = key;
            this.value = value;
        }

        @Override
        public T getKey() {
            // TODO Auto-generated method stub
            return key;
        }

        @Override
        public V getValue() {
            // TODO Auto-generated method stub
            return value;
        }

        @Override
        public V setValue(V arg0) {
            V v = value;
            value = arg0;
            return v;
        }

    }

    public TreeMap() {
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/TreeMap.java - Eclipse

```java
    public TreeMap() {
        this.rb = new RBTree<>();
        NIL = rb.getNILNode();
    }

    @Override
    public Entry<T, V> ceilingEntry(T key) {

        if (key == null) {
            throw new RuntimeErrorException(new Error());
        }
        INode<T, V> x = rb.getRoot();
        Entry<T, V> e = null;
        INode<T, V> oldx = x;
        while (true) {
            if (x == NIL) {
                INode<T, V> y = oldx;
                while (y != NIL) {
                    if (y.getKey().compareTo(key) > 0) {
                        e = new mapEntry(y.getKey(), y.getValue());
                        break;
                    }
                    x = y;
                    y = y.getParent();
                }
                break;
            }
            if (key.compareTo(x.getKey()) < 0) {
                oldx = x;
                x = x.getLeftChild();
            } else if (key.compareTo(x.getKey()) > 0) {
                oldx = x;
                x = x.getRightChild();
            } else if (key.compareTo(x.getKey()) == 0) {
                e = new mapEntry(key, x.getValue());
                break;
            }
        }
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/TreeMap.java - Eclipse

```java
        }
        return e;
    }

    @Override
    public T ceilingKey(T key) {
        Entry<T, V> e = ceilingEntry(key);
        if (e == null) {
            return null;
        }
        return e.getKey();
    }

    @Override
    public void clear() {
        rb.clear();
    }

    @Override
    public boolean containsKey(T key) {
        return rb.contains(key);
    }

    @Override
    public boolean containsValue(V value) {
        if (value == null) {
            throw new RuntimeErrorException(new Error());
        }
        INode<T, V> root = rb.getRoot();
        if (root == rb.getNILNode())
            return false;
        Queue<INode<T, V>> q = new LinkedList<INode<T, V>>();
        q.add(root);
        while (!q.isEmpty()) {
            INode<T, V> n = q.poll();
            if (n.getValue().equals(value)) {
                return true;
            } else {
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/TreeMap.java - Eclipse

```java
            } else {
                if (n.getLeftChild() != rb.getNILNode()) {
                    q.add(n.getLeftChild());
                }
                if (n.getRightChild() != rb.getNILNode()) {
                    q.add(n.getRightChild());
                }
            }
        }
        return false;
    }

    @Override
    public Set<Entry<T, V>> entrySet() {
        INode<T, V> root = rb.getRoot();
        Stack<INode<T, V>> s = new Stack<INode<T, V>>();
        Set<Entry<T, V>> entrySet = new LinkedHashSet<Entry<T, V>>();
        if (root == rb.getNILNode())
            return entrySet;
        s.add(root);
        while (!s.isEmpty()) {
            INode<T, V> x = s.peek().getLeftChild();
            if (x != rb.getNILNode()) {
                s.push(x);
            } else {
                do {
                    x = s.pop();
                    entrySet.add(new AbstractMap.SimpleEntry<T, V>(x.getKey(), x.getValue()));
                    // System.out.println(x.getKey());
                } while (!s.isEmpty() && x.getRightChild() == rb.getNILNode());
                if (x.getRightChild() != rb.getNILNode()) {
                    s.push(x.getRightChild());
                }
            }
        }
        return entrySet;
    }
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/TreeMap.java - Eclipse

```java
    @Override
    public Entry<T, V> firstEntry() {
        INode<T, V> x = rb.getRoot();
        if (x == rb.getNILNode()) {
            return null;
        } else {
            while (x.getLeftChild() != rb.getNILNode()) {
                x = x.getLeftChild();
            }
        }
        return new mapEntry(x.getKey(), x.getValue());
    }

    @Override
    public T firstKey() {
        INode<T, V> x = rb.getRoot();
        if (x == rb.getNILNode()) {
            return null;
        } else {
            while (x.getLeftChild() != rb.getNILNode()) {
                x = x.getLeftChild();
            }
        }
        return x.getKey();
    }

    @Override
    public Entry<T, V> floorEntry(T key) {
        if (key == null) {
            throw new RuntimeErrorException(new Error());
        }
        INode<T, V> y = rb.getNILNode();
        INode<T, V> x = rb.getRoot();
        while (x != rb.getNILNode()) {
            y = x;
            if (key.compareTo(x.getKey()) < 0) {
                x = x.getLeftChild();
            } else if (key.compareTo(x.getKey()) > 0) {
                x = x.getRightChild();
            } else if (key.compareTo(x.getKey()) == 0) {
                return new mapEntry(x.getKey(), x.getValue());
            }
        }
        if (y != rb.getNILNode()) {
            return null;
        y = y.getParent();
        return new mapEntry(y.getKey(), y.getValue());
    }

    @Override
    public T floorKey(T key) {
        if (key == null) {
            throw new RuntimeErrorException(new Error());
        }
        INode<T, V> y = rb.getNILNode();
        INode<T, V> x = rb.getRoot();
        while (x != rb.getNILNode()) {
            y = x;
            if (key.compareTo(x.getKey()) < 0) {
                x = x.getLeftChild();
            } else if (key.compareTo(x.getKey()) > 0) {
                x = x.getRightChild();
            } else if (key.compareTo(x.getKey()) == 0) {
                return key;
            }
        }
        if (key.compareTo(y.getKey()) > 0) {
            return y.getKey();
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/TreeMap.java - Eclipse

```java
            return y.getKey();
        }
        while (y != rb.getNILNode() && y != y.getParent().getRightChild()) {
            y = y.getParent();
        }
        if (y == rb.getNILNode())
            return null;
        y = y.getParent();
        return y.getKey();
    }

    @Override
    public V get(T key) {
        if (key == null) {
            throw new RuntimeErrorException(new Error());
        }
        return rb.search(key);
    }

    @Override
    public ArrayList<Entry<T, V>> headMap(T toKey) {
        return headMap(toKey, false);
    }

    @Override
    public ArrayList<Entry<T, V>> headMap(T toKey, boolean inclusive) {
        if (toKey == null) {
            throw new RuntimeErrorException(new Error());
        }
        INode<T, V> x = rb.getRoot();
        Stack<INode<T, V>> stack = new Stack<>();
        ArrayList<Entry<T, V>> array = new ArrayList<Entry<T, V>>();
        while (!stack.isEmpty() || x != NIL) {
            while (x != NIL) {
                stack.push(x);
                x = x.getLeftChild();
            }
            x = stack.pop();
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/TreeMap.java - Eclipse

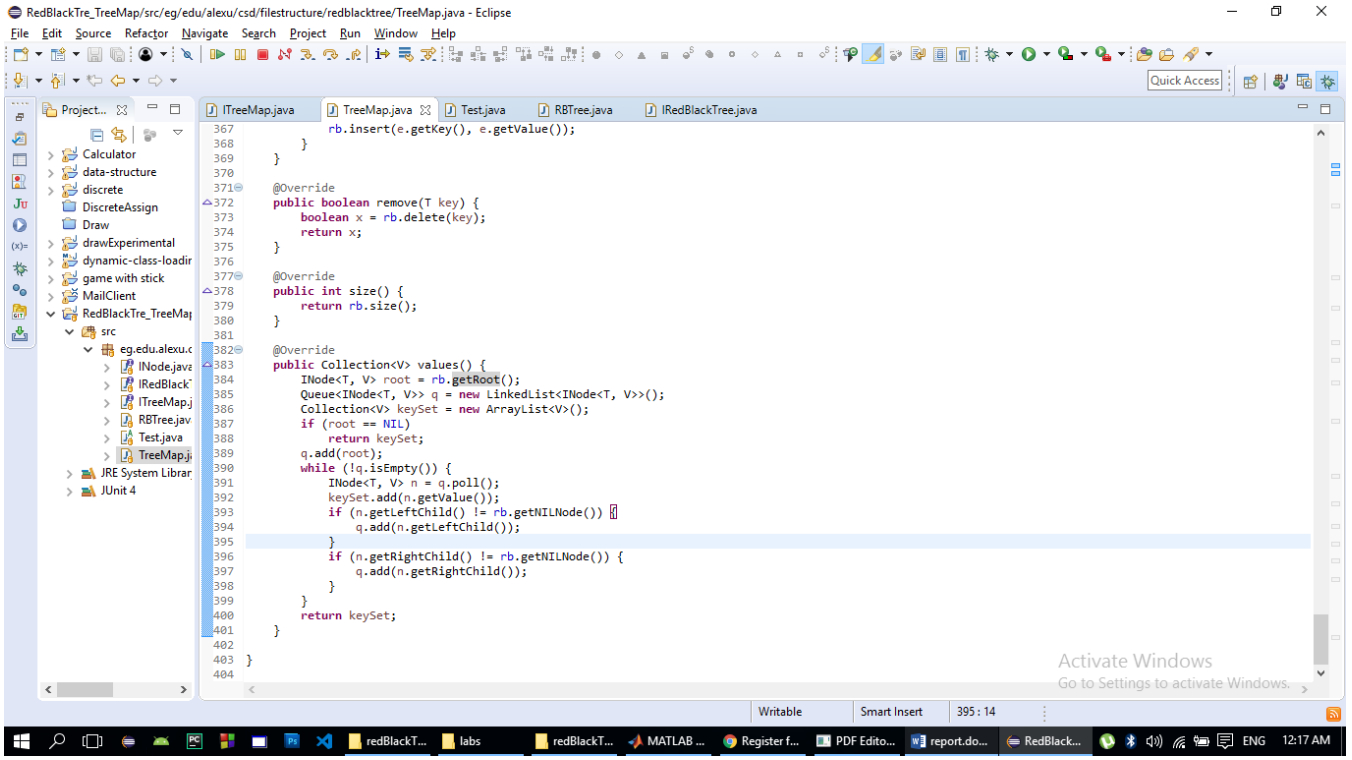
```java
            x = stack.pop();
            if (x.getKey().compareTo(toKey) < 0 || (inclusive && x.getKey().compareTo(toKey) == 0)) {
                array.add(new AbstractMap.SimpleEntry<T, V>(x.getKey(), x.getValue()));
            } else {
                break;
            }
            x = x.getRightChild();
        }
        return array;
    }

    @Override
    public Set<T> keySet() {
        INode<T, V> x = rb.getRoot();
        Stack<INode<T, V>> stack = new Stack<>();
        Set<T> keySet = new LinkedHashSet<T>();
        while (!stack.isEmpty() || x != NIL) {
            while (x != NIL) {
                stack.push(x);
                x = x.getLeftChild();
            }
            x = stack.pop();
            keySet.add(x.getKey());
            x = x.getRightChild();
        }
        return keySet;
    }

    @Override
    public Entry<T, V> lastEntry() {
        INode<T, V> x = rb.getRoot();
        if (x == rb.getNILNode()) {
            return null;
        } else {
            while (x.getRightChild() != rb.getNILNode()) {
                x = x.getRightChild();
            }
        }
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/TreeMap.java - Eclipse

```java
        }
        return new mapEntry(x.getKey(), x.getValue());
    }

    @Override
    public T lastKey() {
        INode<T, V> x = rb.getRoot();
        if (x == rb.getNILNode()) {
            return null;
        } else {
            while (x.getRightChild() != rb.getNILNode()) {
                x = x.getRightChild();
            }
        }
        return x.getKey();
    }

    @Override
    public Entry<T, V> pollFirstEntry() {
        INode<T, V> min = rb.getRoot();
        if (min == NIL) {
            return null;
        }
        while (min.getLeftChild() != NIL) {
            min = min.getLeftChild();
        }
        Entry<T, V> e = new AbstractMap.SimpleEntry<T, V>(min.getKey(), min.getValue());
        remove(min.getKey());
        return e;
    }

    @Override
    public Entry<T, V> pollLastEntry() {
        INode<T, V> max = rb.getRoot();
        if (max == NIL) {
            return null;
        }
        while (max.getRightChild() != NIL) {
```

RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/TreeMap.java - Eclipse

```java
        while (max.getRightChild() != NIL) {
            max = max.getRightChild();
        }
        Entry<T, V> e = new AbstractMap.SimpleEntry<T, V>(max.getKey(), max.getValue());
        remove(max.getKey());
        return e;
    }

    @Override
    public void put(T key, V value) {
        if (key == null) {
            throw new RuntimeErrorException(new Error());
        }
        rb.insert(key, value);
    }

    @Override
    public void putAll(Map<T, V> map) {
        if (map == null) {
            throw new RuntimeErrorException(new Error());
        }
        for (Entry<T, V> e : map.entrySet()) {
            rb.insert(e.getKey(), e.getValue());
        }
    }

    @Override
    public boolean remove(T key) {
        boolean x = rb.delete(key);
        return x;
    }

    @Override
    public int size() {
        return rb.size();
    }

    @Override
```


RedBlackTre_TreeMap/src/eg/edu/alexu/csd/filestructure/redblacktree/TreeMap.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project...
Calculator
data-structure
discrete
DiscreteAssign
Draw
drawExperimental
dynamic-class-loadir
game with stick
MailClient
RedBlackTre_TreeMa
src
eg.edu.alexu.c
INode.java
IRedBlack
ITreeMap.j
RBTree.jav
Test.java
TreeMap.j
JRE System Librar
JUnit 4
ITreeMap.java
TreeMap.java
Test.java
RBTree.java
IRedBlackTree.java
367 rb.insert(e.getKey(), e.getValue());
368 }
369 }
370
371 @Override
372 public boolean remove(T key) {
373 boolean x = rb.delete(key);
374 return x;
375 }
376
377 @Override
378 public int size() {
379 return rb.size();
380 }
381
382 @Override
383 public Collection<V> values() {
384 INode<T, V> root = rb.getRoot();
385 Queue<INode<T, V>> q = new LinkedList<INode<T, V>>();
386 Collection<V> keySet = new ArrayList<V>();
387 if (root == NIL)
388 return keySet;
389 q.add(root);
390 while (!q.isEmpty()) {
391 INode<T, V> n = q.poll();
392 keySet.add(n.getValue());
393 if (n.getLeftChild() != rb.getNILNode()) {
394 q.add(n.getLeftChild());
395 }
396 if (n.getRightChild() != rb.getNILNode()) {
397 q.add(n.getRightChild());
398 }
399 }
400 return keySet;
401 }
402
403 }
404
Activate Windows
Go to Settings to activate Windows.
Writable
Smart Insert
395 : 14
ENG
12:17 AM