```java
import java.util.Comparator;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Queue;

public class BTree<E>
{
    public Node<E> root;
    public Comparator<E> comp;
    public int order;

    public BTree(int theOrder, Comparator<E> theComp)
    {
        order = theOrder;
        comp = theComp;
        root = new Node<E>(theOrder, theComp);
    }

    public void add(E item)
    {
        Node<E> node = findLeaf(root, item);

        node.leafAdd(item);

        while(node.hasOverflow()) {
            node.split();
            node = node.parent;
        }
        if(root.parent != null) {
            root = root.parent;
        }

    }

    private Node<E> findLeaf(Node<E> curr, E item)
    {
        while(!curr.isLeaf()) {
            curr = curr.childToFollow(item);
        }
        return curr;
    }

    public boolean contains(E item)
    {
        return findNode(root, item) != null;
    }

    private Node<E> findNode(Node<E> curr, E item)
    {
        if(curr.contains(item)) {
            return curr;
        }
        else if(curr.isLeaf()) {
            return null;
        }
        else {
            return findNode(curr.childToFollow(item), item);
```

```
58          }
59      }
60
61      public void inorder(Visitor<E> visitor)
62      {
63          inorder(visitor, root);
64      }
65
66      private void inorder(Visitor<E> visitor, Node<E> curr)
67      {
68          if(curr == null) {
69              return;
70          }
71          else {
72              int dataSize = curr.data.size();
73              int childSize = curr.children.size();
74              for(int i = 0; i < dataSize ; ++i) {
75                  if(i < childSize) {
76                      inorder(visitor, curr.children.get(i));
77                  }
78                  visitor.visit(curr.data.get(i));
79              }
80              if(dataSize+1 == childSize) {
81                  inorder(visitor, curr.children.get(dataSize));
82              }
83          }
84      }
85
86      public String toStringSorted()
87      {
88          StringVisitor<E> visitor = new StringVisitor<>();
89          inorder(visitor);
90          return visitor.getValue();
91      }
92
93      public String toString()
94      {
95          String result = "";
96          LinkedList<Node<E>> queue = new LinkedList<Node<E>>();
97          queue.add(root);
98
99          while(!queue.isEmpty()) {
100             Node<E> curr = queue.poll();
101             LinkedList<Node<E>> children = curr.children;
102             for(Node<E> child: children) {
103                 queue.add(child);
104             }
105             String str = curr.data.toString();
106             result = result + " " +  str.replaceAll(",", "");
107         }
108         result = "[" + result.trim() + "]";
109         return result;
110     }
111 }
112
```