KAUNO TECHNOLOGIJOS UNIVERSITETAS

Duomenų struktūros

2 Labaratorinio darbo ataskaita

Atliko Benas Montvydas Iff 9-11

Atlikti metodai ir ištestuoti metodai:

BstSet:

remove(E element):

```java
@Override
public void remove(E element) {
    if(element == null)
    {
        throw new IllegalArgumentException("Element is null i remove(E element)");
    }

    root = removeRecursive(element, root);
}
```

```
Elementų šalinimas 4 elementu iš kopijos. Prieš šalinimą aibės dydis yra 9
Pašalinama: TA100=Renault_Laguna:1997 50000 1700.0. Kopijos dydis: 8
Pašalinama: TA101=Renault_Megane:2001 20000 3500.0. Kopijos dydis: 7
Pašalinama: TA102=Renault_Scenic:2008 68011 80805.6. Kopijos dydis: 6
Pašalinama: TA103=Renault_Laguna:2001 115900 700.0. Kopijos dydis: 5
Kopijos visualizacija:
>──┬●TA108=Honda_Civic:2007 10000 850.3
│  ┌──●TA107=Renault_Megane:1946 365100 950.0
└──┼●TA106=Renault_Laguna:2001 115900 7500.0
   │  ┌──●TA105=Honda_Civic:2001 36400 80.3
   └──┴●TA104=Renault_Megane:1946 365100 9500.0


 Visų elementų šalinimas elementu iš kopijos. Prieš šalinimą aibės dydis yra 5
Pašalinama: TA107=Renault_Megane:1946 365100 950.0. Kopijos dydis: 5
Pašalinama: TA104=Renault_Megane:1946 365100 9500.0. Kopijos dydis: 4
Pašalinama: TA105=Honda_Civic:2001 36400 80.3. Kopijos dydis: 3
Pašalinama: TA106=Renault_Laguna:2001 115900 7500.0. Kopijos dydis: 2
Pašalinama: TA107=Renault_Megane:1946 365100 950.0. Kopijos dydis: 1
Pašalinama: TA108=Honda_Civic:2007 10000 850.3. Kopijos dydis: 0
Pašalinama: TA100=Renault_Laguna:1997 50000 1700.0. Kopijos dydis: 0
```

removeRecursive(E element, BstNode<E> node):

```
private BstNode<E> removeRecursive(E element, BstNode<E> node) {
    if (node == null)
    {
        return node;
    }

    int cmp = c.compare(element,node.element); // Palyginimas

    //Paieška
    if(cmp<0)
    {
        node.left = removeRecursive(element, node.left);
    }
    else if(cmp > 0)
    {
        node.right = addRecursive(element, node.right);
    }
    else if(node.left != null && node.right != null)
    {
        BstNode<E> Nodemax = getMax(node.left);

        node.element = Nodemax.element;
        node.left = removeMax(node.left);
        size--;
    }
    else
        {
            if (node.left != null)
            {
                node = node.left;
            }
            else
            {
                node = node.right;
            }
            size--;
        }

    return node;
```

```
private BstNode<E> removeMax(BstNode<E> node)
{
    if (node == null){
        return null;
    }
    else if(node.right != null)
    {
        node.right = removeMax(node.right);
        return node;
    }
    else
        {return node.left;}
}
private BstNode<E> getMax(BstNode<E> node)
{
    BstNode<E> parent = null;
    while(node != null)
    {
        parent = node;
        node = node.right;
    }
    return parent;
}
```

remove():

```
@Override
public void remove() {
    if (stack == null) {
        throw new IllegalStateException();
    } else {
        stack.pop();
    }
}
```

addAllSets(Set<E> set):

```java
@Override
public void addAll(Set<E> set) {
    Object[] array = set.toArray();
    for (Object object: array)
    {
        if (!contains((E) object))
        {
            addRecursive((E)object,root);
        }
    }
}
```

```
30| Naudojamas metodas addAll, prie kopijos(vienas elementas toks pat, 2 skirtingi). Prieš pridėjima aibės dydis yra 7
31| Kopijos visualizacija prieš:
32| >──┬●TA108=Honda_Civic:2007 10000 850.3
   |  ┌──●TA107=Renault_Megane:1946 365100 950.0
   └──┼●TA106=Renault_Laguna:2001 115900 7500.0
      └──┬●TA104=Renault_Megane:1946 365100 9500.0
         |     ┌──●TA103=Renault_Laguna:2001 115900 700.0
         |  ┌──┴●TA102=Renault_Scenic:2008 68011 80805.6
         └──┴●TA101=Renault_Megane:2001 20000 3500.0

33| Po pridėjimo aibės dydis yra 9
34| Kopijos visualizacija po:
35| >──┬●TA108=Honda_Civic:2007 10000 850.3
   |  ┌──●TA107=Renault_Megane:1946 365100 950.0
   └──┼●TA106=Renault_Laguna:2001 115900 7500.0
      |  ┌──●TA105=Honda_Civic:2001 36400 80.3
      └──┼●TA104=Renault_Megane:1946 365100 9500.0
         |     ┌──●TA103=Renault_Laguna:2001 115900 700.0
         |  ┌──┴●TA102=Renault_Scenic:2008 68011 80805.6
         └──┼●TA101=Renault_Megane:2001 20000 3500.0
            └──●TA100=Renault_Laguna:1997 50000 1700.0
```

containsAll(Set<E> set):

```java
@Override
public boolean containsAll(Set<E> set) {
    Object[] objects = set.toArray();
    for (Object object : objects)
    {
        if (!this.contains((E) object))
        {
            return false;
        }
    }
    return true;
}
```

```
53|
54| Tikrinsime containsAll
55|
56| Kai false
57| Testavimo BSTSET:
58| BSTSET elementai: TA100=Renault_Laguna:1997 50000 1700.0
59| BSTSET elementai: TA105=Honda_Civic:2001 36400 80.3
60| BSTSET elementai: TA108=Honda_Civic:2007 10000 850.3
61|
62| Aibes elementai
63| Aibė elementai: TA104=Renault_Megane:1946 365100 9500.0
64| Aibė elementai: TA106=Renault_Laguna:2001 115900 7500.0
65| Aibė elementai: TA107=Renault_Megane:1946 365100 950.0
66| Aibė elementai: TA108=Honda_Civic:2007 10000 850.3
67| Atsakymas false
68|
```

```
69| Kai true
70| Testavimo BSTSET:
71| BSTSET elementai: TA100=Renault_Laguna:1997 50000 1700.0
72| BSTSET elementai: TA105=Honda_Civic:2001 36400 80.3
73| BSTSET elementai: TA108=Honda_Civic:2007 10000 850.3
74|
75| Aibes elementai
76| Aibė elementai: TA100=Renault_Laguna:1997 50000 1700.0
77| Aibė elementai: TA105=Honda_Civic:2001 36400 80.3
78| Aibė elementai: TA108=Honda_Civic:2007 10000 850.3
79| Atsakymas true
```

retainAll(Set<E> set):

```
@Override
public void retainAll(Set<E> set) {
    IteratorBst iter = new IteratorBst( ascendingOrder: true);
    while (iter.hasNext())
    {
        E tempElement = iter.next();
        if (!set.contains(tempElement))
        {
            remove(tempElement);
        }
    }
}
```

```
 81| Testuojame retainAll
 82|
 83| Pirma aibė:
 84| Pirmos aibės elementai: TA100=Renault_Laguna:1997 50000 1700.0
 85| Pirmos aibės elementai: TA104=Renault_Megane:1946 365100 9500.0
 86| Pirmos aibės elementai: TA105=Honda_Civic:2001 36400 80.3
 87| Pirmos aibės elementai: TA106=Renault_Laguna:2001 115900 7500.0
 88| Pirmos aibės elementai: TA107=Renault_Megane:1946 365100 950.0
 89| Pirmos aibės elementai: TA108=Honda_Civic:2007 10000 850.3
 90|
 91| Antra aibė:
 92| Antros aibės elementai: TA104=Renault_Megane:1946 365100 9500.0
 93| Antros aibės elementai: TA106=Renault_Laguna:2001 115900 7500.0
 94| Antros aibės elementai: TA107=Renault_Megane:1946 365100 950.0
 95| Antros aibės elementai: TA108=Honda_Civic:2007 10000 850.3
 96|
 97| Po retainAll
 98| Pirmos aibės elementai: TA104=Renault_Megane:1946 365100 9500.0
 99| Pirmos aibės elementai: TA106=Renault_Laguna:2001 115900 7500.0
100| Pirmos aibės elementai: TA107=Renault_Megane:1946 365100 950.0
101| Pirmos aibės elementai: TA108=Honda_Civic:2007 10000 850.3
102|
```

```
103| Trečia aibė:
104| Trečios aibės elementai: TA100=Renault_Laguna:1997 50000 1700.0
105| Trečios aibės elementai: TA105=Honda_Civic:2001 36400 80.3
106| Trečios aibės elementai: TA108=Honda_Civic:2007 10000 850.3
107|
108| Po retainAll
109| Pirmos aibės elementai: TA108=Honda_Civic:2007 10000 850.3
110|
```

headSet(E e):

```java
@Override
public Set<E> headSet(E element) {
    Set<E> tempBst = new BstSet<>();
    IteratorBst iter = new IteratorBst( ascendingOrder: true);
    while (iter.hasNext())
    {
        E tempElement = iter.next();
        if (tempElement == element)
        {
            break;
        }
        tempBst.add(tempElement);
    }
    return tempBst;
}
```

```
Pradinė aibė
Pradinės aibės elementai: TA100=Renault_Laguna:1997 50000 1700.0
Pradinės aibės elementai: TA104=Renault_Megane:1946 365100 9500.0
Pradinės aibės elementai: TA105=Honda_Civic:2001 36400 80.3
Pradinės aibės elementai: TA106=Renault_Laguna:2001 115900 7500.0
Pradinės aibės elementai: TA107=Renault_Megane:1946 365100 950.0
Pradinės aibės elementai: TA108=Honda_Civic:2007 10000 850.3
```

```
Elementas TA106=Renault_Laguna:2001 115900 7500.0

Kita aibė, po headSet
Po headSet aibės elementai: TA100=Renault_Laguna:1997 50000 1700.0
Po headSet aibės elementai: TA104=Renault_Megane:1946 365100 9500.0
Po headSet aibės elementai: TA105=Honda_Civic:2001 36400 80.3
```

tailSet(E e):

```java
@Override
public Set<E> tailSet(E element) {
    Set<E> tempBst = new BstSet<>();
    IteratorBst iter = new IteratorBst( ascendingOrder: true);
    while (iter.hasNext())
    {
        E tempElement = iter.next();
        if (tempElement.compareTo(element) >= 0)
        {
            tempBst.add(tempElement);
        }
    }
    return tempBst;
}
```

```
Pradinė aibė
Pradinės aibės elementai: TA100=Renault_Laguna:1997 50000 1700.0
Pradinės aibės elementai: TA104=Renault_Megane:1946 365100 9500.0
Pradinės aibės elementai: TA105=Honda_Civic:2001 36400 80.3
Pradinės aibės elementai: TA106=Renault_Laguna:2001 115900 7500.0
Pradinės aibės elementai: TA107=Renault_Megane:1946 365100 950.0
Pradinės aibės elementai: TA108=Honda_Civic:2007 10000 850.3
```

```
Elementas TA106=Renault_Laguna:2001 115900 7500.0

Kita aibė, po tailSet
Po tailSet aibės elementai: TA106=Renault_Laguna:2001 115900 7500.0
Po tailSet aibės elementai: TA107=Renault_Megane:1946 365100 950.0
Po tailSet aibės elementai: TA108=Honda_Civic:2007 10000 850.3
```

subSet(E element1, E element2):

```
    */
    @Override
    public Set<E> subSet(E element1, E element2) {
        Set<E> tempBst = new BstSet<>();
        IteratorBst iter = new IteratorBst( ascendingOrder: true);
        while (iter.hasNext())
        {
            E tempElement = iter.next();
            if (tempElement.compareTo(element1) >= 0)
            {
                tempBst.add(tempElement);
            }
            if (tempElement.compareTo(element2) >= 0)
            {
                break;
            }
        }
        return tempBst;
    }
```

```
Pradinė aibė
Pradinės aibės elementai: TA100=Renault_Laguna:1997 50000 1700.0
Pradinės aibės elementai: TA104=Renault_Megane:1946 365100 9500.0
Pradinės aibės elementai: TA105=Honda_Civic:2001 36400 80.3
Pradinės aibės elementai: TA106=Renault_Laguna:2001 115900 7500.0
Pradinės aibės elementai: TA107=Renault_Megane:1946 365100 950.0
Pradinės aibės elementai: TA108=Honda_Civic:2007 10000 850.3
```

```
135| Elementas TA100=Renault_Laguna:1997 50000 1700.0
136| Elementas TA106=Renault_Laguna:2001 115900 7500.0
137|
138| Kita aibė, po subSet
139| Po subSet aibės elementai: TA100=Renault_Laguna:1997 50000 1700.0
140| Po subSet aibės elementai: TA104=Renault_Megane:1946 365100 9500.0
141| Po subSet aibės elementai: TA105=Honda_Civic:2001 36400 80.3
142| Po subSet aibės elementai: TA106=Renault_Laguna:2001 115900 7500.0
143|
144| Automobiliu aibė su iteratoriumi:
```

AvlSet:

remove():

```java
@Override
public void remove(E element) {
    if (element == null) {
        throw new IllegalArgumentException("Element is null in remove(E element)");
    }
    root = removeRecursive(element, (AVLNode<E>) root);
}
```

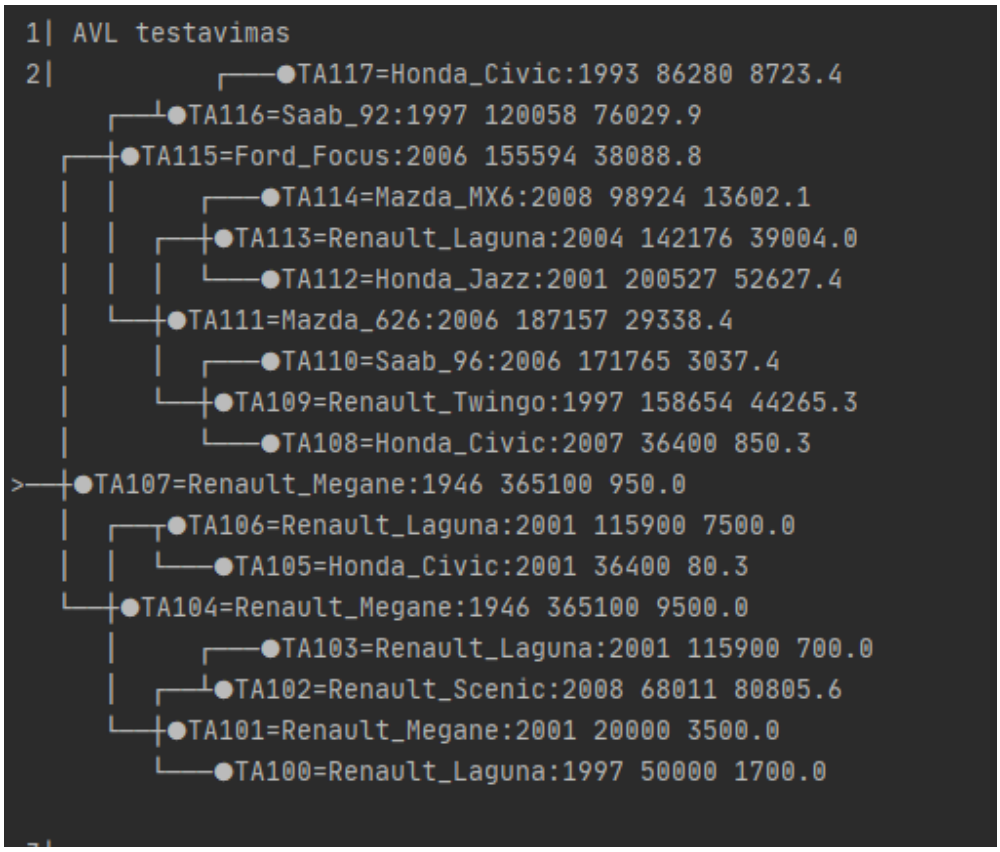
```
153| AVL remove, testavimas
154| Pradinės aibės elementai: TA100=Renault_Laguna:1997 50000 1700.0
155| Pradinės aibės elementai: TA101=Renault_Megane:2001 20000 3500.0
156| Pradinės aibės elementai: TA102=Renault_Scenic:2008 68011 80805.6
157|
158| Triname TA100=Renault_Laguna:1997 50000 1700.0
159|
160| Po vieno nario ištrynimo aibės elementai: TA101=Renault_Megane:2001 20000 3500.0
161| Po vieno nario ištrynimo aibės elementai: TA102=Renault_Scenic:2008 68011 80805.6
162|
163| Visus narius ištrynus
164| Ištrynus visus narius aibės ilgis 0
165| Bandome ištrinti, elementą iš tuščios aibės
166| Po bandymo, aibės ilgis 0
167|
```

removeRecursive(E element, BstNode<E> node):

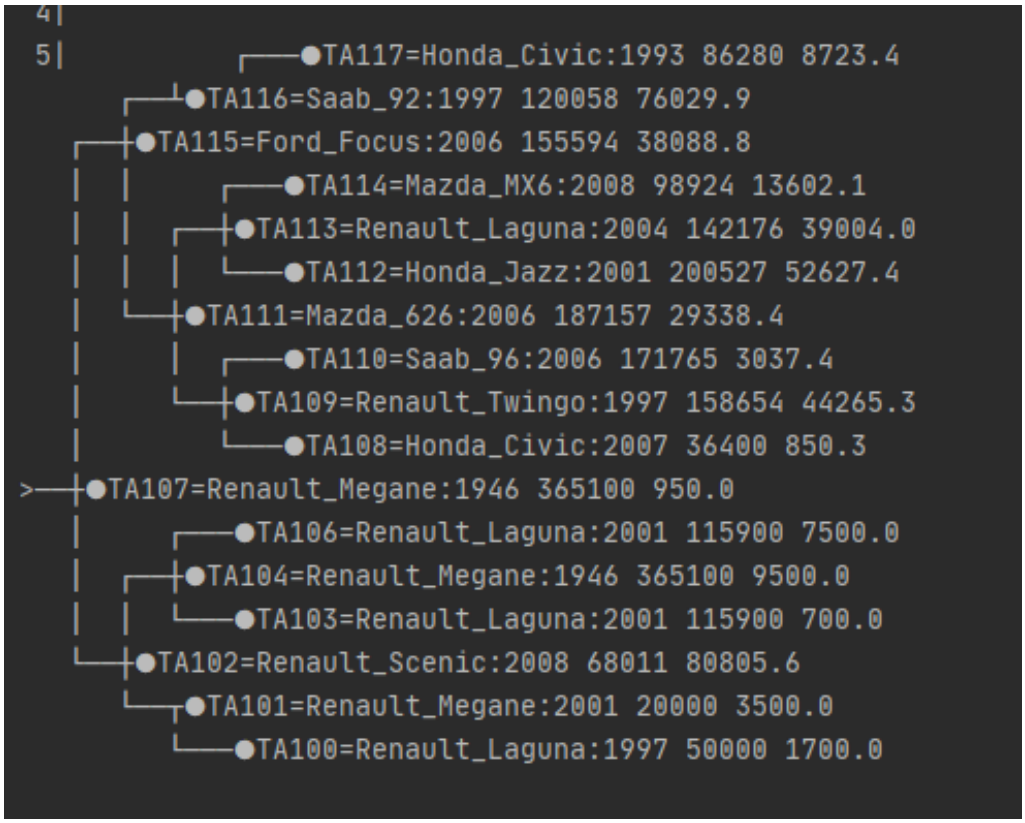
```java
private AVLNode<E> removeRecursive(E element, AVLNode<E> n) {

    if(n == null)
    {
        return n;
    }
    else
        {
            int cmp = element.compareTo(n.element);
            if(cmp < 0)
            {
                n.setLeft(removeRecursive(element, n.getLeft()));
            }
            else if(cmp > 0)
            {
                n.setRight(removeRecursive(element, n.getRight()));
            }
            else
                {
                    //If has no children
                    if (n.left == null && n.right == null) {
                        n = null;
                    }
                    //If has one child
                    else if (n.left == null && n.right != null) {
                        n = n.getRight();
                    }
                    else if (n.right == null && n.left != null) {
                        n = n.getLeft();
                    }
                    //If have two children
                    else {
                        AVLNode<E> temp = getLastoLeft(n.getRight());
                        n.element = temp.element;
                        n.setRight(removeRecursive(temp.element, n.getRight()));
                    }
                }
        }
    if(n != null)
```

```java
        }
    if(n != null)
    {
        n.height = Math.max(height(n.getLeft()), height(n.getRight())) + 1;
        if ((height(n.getLeft()) - height(n.getRight())) == 2) {
            n = (height(n.getLeft().getLeft()) >= height(n.getLeft().getRight())) ? rightRotation(n) : doubleRightRotation(n);
        } else if ((height(n.getLeft()) - height(n.getRight())) == -2) {
            n = (height(n.getRight().getRight()) >= height(n.getRight().getLeft())) ? leftRotation(n) : doubleLeftRotation(n);
        }
    }

    return n;
}

private AVLNode<E> getLastoLeft(AVLNode<E> node)
{
    AVLNode<E> temp = node;
    while (temp.getLeft() != null) {
        temp = temp.getLeft();
    }
    return temp;
}
```

Prieš išemimą



Po c6 išemimo



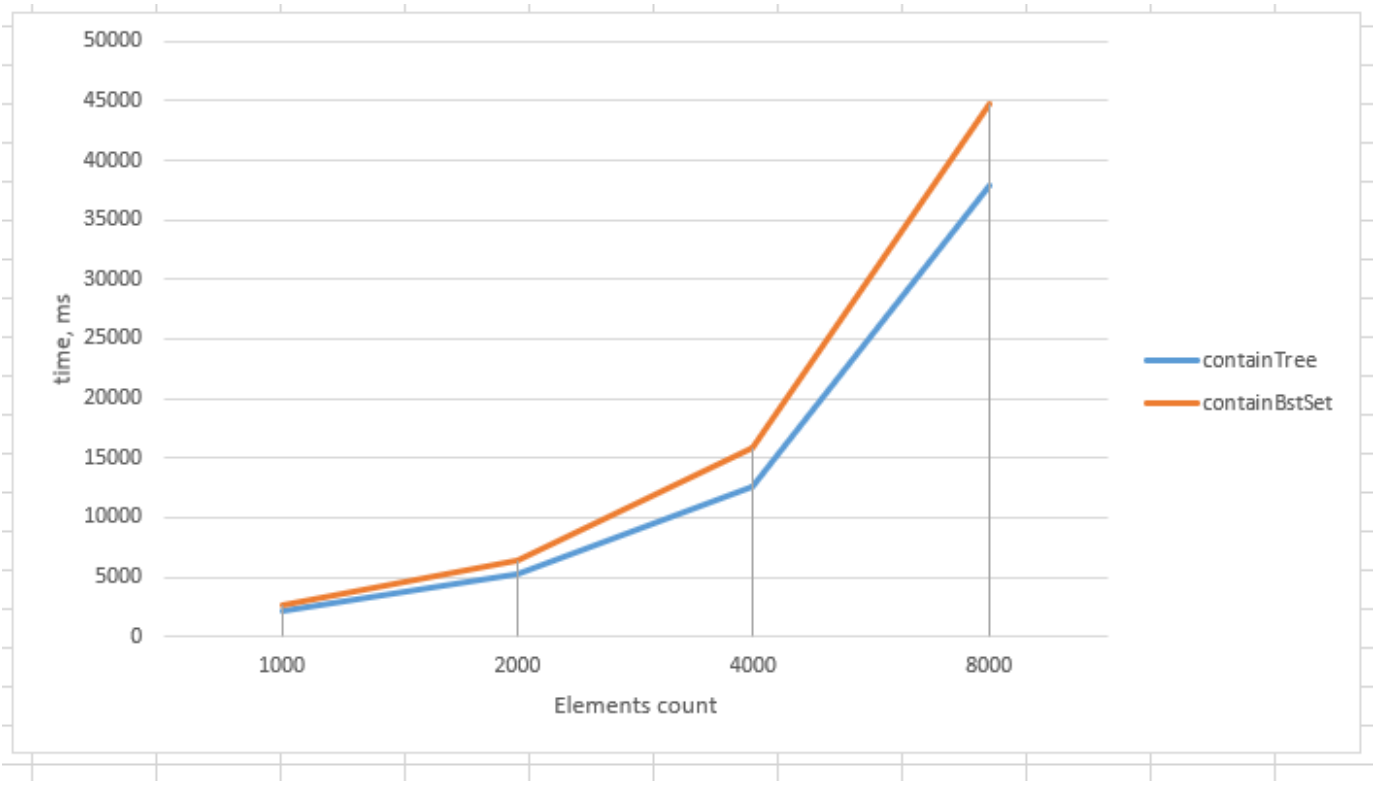
Greitaveikos testavimas:

7 variantas

Class BstSet: contains()	Class java.util.TreeSet<E>: contains()

Rezultatai :

```
REMEMBER: The numbers below are just data. To gain reusable insights, you need to follow up on
why the numbers are the way they are. Use profilers (see -prof, -lprof), design factorial
experiments, perform baseline and negative tests that provide experimental control, make sure
the benchmarking environment is safe on JVM/OS/HW level, ask for reviews from the domain experts.
Do not assume the numbers tell you what you want them to tell.

Benchmark                (elementCount)  Mode  Cnt      Score       Error  Units
Benchmark.BstContains             10000  avgt    5   2672,085 ±   483,728  us/op
Benchmark.BstContains             20000  avgt    5   6363,673 ±   709,280  us/op
Benchmark.BstContains             40000  avgt    5  15855,695 ±  3163,708  us/op
Benchmark.BstContains             80000  avgt    5  44703,766 ± 20806,555  us/op
Benchmark.TreeContains            10000  avgt    5   2233,502 ±   198,400  us/op
Benchmark.TreeContains            20000  avgt    5   5272,012 ±   472,397  us/op
Benchmark.TreeContains            40000  avgt    5  12658,327 ±  2251,865  us/op
Benchmark.TreeContains            80000  avgt    5  37945,472 ± 17683,493  us/op

Process finished with exit code 0
```



Išvados: TreeSet contains it BstSet contains sudetingumas yra O(log n), jei tyrima daryčiau porą kartu rezultatai, gali ir skirtis priklausomai ką veikiu prie kompiuteriu. Bet šių metodų greitaveika yra gana panaši.