KAUNO TECHNOLOGIJOS UNIVERSITETAS

Duomenų struktūros (P175B014)

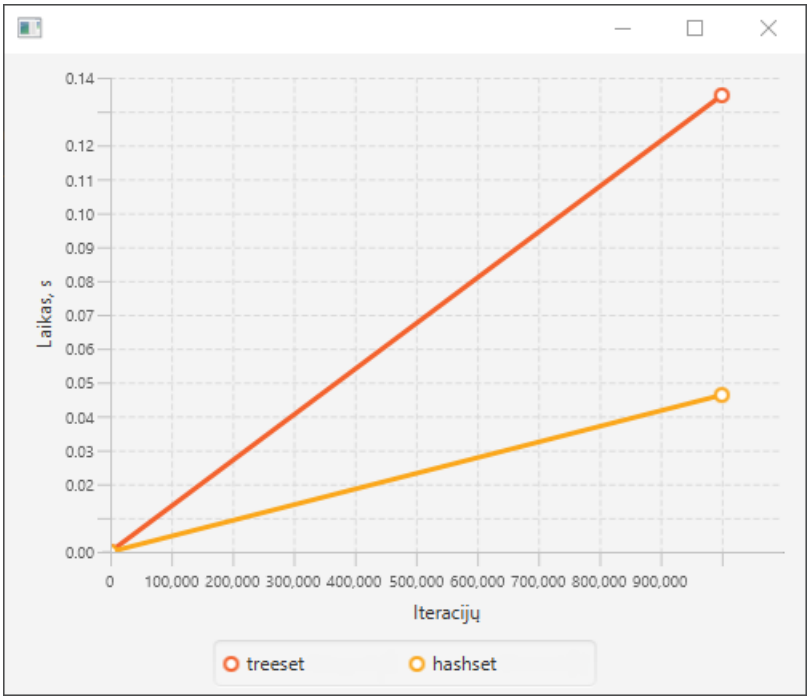
Laboratorinio darbo Nr.2 ataskaita

Atliko **Martynas Kemežys** gr. IF-8/1

Priėmė lekt. **Karčiauskas Eimutis**

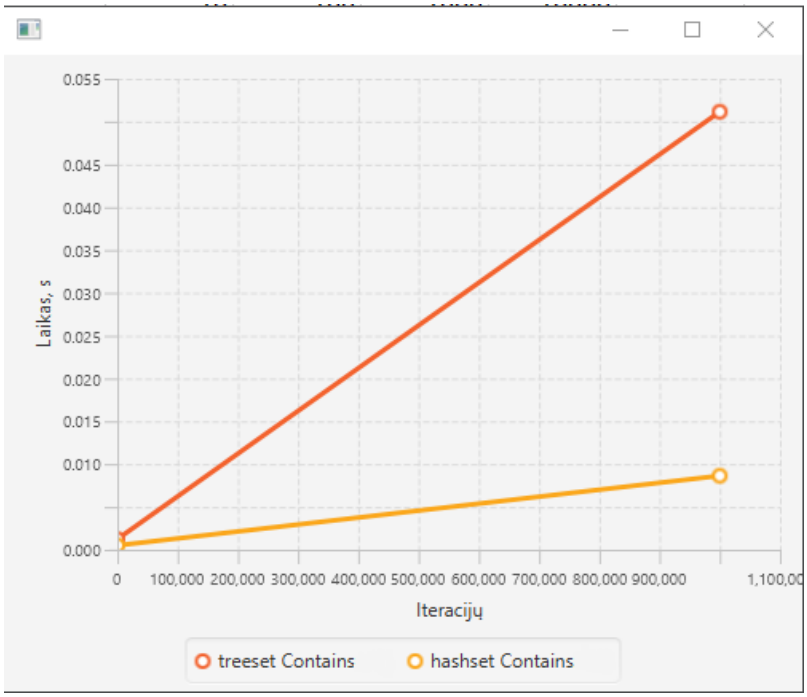
1. Metodų greitaveika TreeSet ir HashSet

1.1. Metodas add(Object c)



HashSet tokiom operacijom kaip paieška/pridėti/pašalinti vidutiniškai reik daug laiko. HashSet yra greitesnė negu TreeSet. HashSet įgyvendinama naudojant maišos lentele. TreeSet naudoja O(Log n) paieškai, įterpimui ir pašalinimui kuris yra aukštesnis negu HashSet, bet TreeSet saugo surūšiuotus duomenis.

1.2. Metodas contains(Object o)



HashSet contains() metodas dirba greičiau nes HashSet duomenų struktūroje elementai per maišos funkciją tiesiogiai priskiriami atminties adresams kaip masyve. Todėl jie pasiekiami tiesiogine kreiptimi. O TreeSet duomenų struktūroje reikia atlikti dvejetainę paiešką.

2. Individualiai nurodyti metodai

7.

```
public SortedSet<E> headSet(E element) {
    if (element == null) {
        throw new NullPointerException();
    }
    BstSet<E> set = new BstSet<>();
    for(E thi : this){
       if(element.compareTo(thi) == 0){
           set.add(thi);
           break;
       }
    set.add(thi);
    }
    return set;
}
```

15.

```
public SortedSet<E> subSet(E element1, E element2) {
    if (element1 == null || element2 == null) {
        throw new NullPointerException();
    }
    boolean foundFirst = false;
    BstSet<E> set = new BstSet<>();

    for(E thi : this){
       if(element1.compareTo(thi) == 0)foundFirst = true;
       if(foundFirst == true){
           if(element2.compareTo(thi) == 0){
           set.add(thi);
           break;
           }
        set.add(thi);
       }
    }
    return set;
}
```

16.

```
public SortedSet<E> tailSet(E element) {
    if (element == null) {
        throw new NullPointerException();
    }
    boolean found = false;
    BstSet<E> set = new BstSet<>();
    for(E thi : this){
       if(element.compareTo(thi) == 0) found = true;
       if(found == true){
         set.add(thi);
       }
    }
    return set;
}
```

10.

```
//Grąžina paskutinį (aukščiausią) elementą.
public E last() {
    if (root == null)
        return null;
    if (root.right == null) {
        return root.element;
    }

    BstNode<E> ref = root;
    while (ref.right.right != null)
        ref = ref.right;

    E data = ref.right.element;
    return data;
}
```

6.

```
public E floor(E e){
    return findFloor(root, null, e);
}
private E findFloor(BstNode<E> root, E result, E element){
    if(root == null)
        return null;

    if ((result == null && root.element.compareTo(element) < 0) || (result != null &&
    root.element.compareTo(element) < 0))
    result = root.element;

    else if (root.left != null)
    result = findLower(root.left, result, element);

    if (root.right != null)
    result = findLower(root.right, result, element);

return result;
}
```

11.

```
public E lower(E e){
    return findLower(root, null, e) ;
}
private E findLower(BstNode<E> root, E result, E element){
    if(root == null)
    return null;

    if ((result == null && root.element.compareTo(element) < 0) || (result != null &&
    root.element.compareTo(result) > 0 && root.element.compareTo(element) < 0))
    result = root.element;

    else if (root.left != null)
    result = findLower(root.left, result, element);

    if (root.right != null)
    result = findLower(root.right, result, element);

    return result;
}
```

3.Išvados

Subalansuotame dvejetainiame paieškos medyje ieškant elemento pirma iteracija atmetama pusė galimų reikšmių, todėl paieška jame vyksta sparčiai.

Nebalansuojamas dvejetainis medis gali „išsigimti“ ir tapti labiau panašus į sąrašą nei į medį. Tuomet paieškos operacijos sudėtingumas bus nebe log(N), o N. Tokiu atveju prarandami pagrindiniai medžio duomenų struktūros privalumai.