Lab 4

Metod:

```
public TreeMap<Integer, Integer> createTreeMap(Entry[] entries) {
    //TODO: Add implementation
    TreeMap <Integer, Integer> map = new  TreeMap<Integer, Integer>();
    for (Entry entry : entries) {
        if (!map.containsKey(entry.key) )
        map.put(entry.key, entry.value);
        }
    return map;
}
```

O(n) eftersom tidskomplexiteten ökar konstant och kollar varje värde genom en for loop.

Metod:

```
public Map<Integer, Integer> retrieve(TreeMap<Integer, Integer> tree, int[]
keys) {
    //TODO: Add implementation
    TreeMap<Integer, Integer> map = new TreeMap<Integer, Integer>();
    for (int key : keys) {
        if (tree.containsKey(key)) {
            map.put(key, tree.get(key));
        }
    }
    return map;
}
```

O(n) eftersom tidskomplexiteten ökar konstant och kollar varje värde genom en for loop.

Metod:

```
public Map<Integer, Integer> retrieve(TreeMap<Integer, Integer> tree, int
fromKey, int toKey) {
    //TODO: Add implementation
    return tree.subMap(fromKey, true, toKey, true);
}
```

O(1) eftersom programmet hittar värdet på engång och behöver inte gå igenom alla värderna för att hitta rätt. Den bara hämtar det värdet på den platsen man letar på.

Metod:

```
public Collection<Integer> retrieveAllKeys(TreeMap<Integer, Integer> tree) {
    //TODO: Add implementation
    return tree.keySet();
}
```

O(1) eftersom programmet hittar värdet på engång och behöver inte gå igenom alla värderna för att hitta rätt. Den bara hämtar det värdet på den platsen man letar på.

Metod:

```
public Collection<Integer> retrieveAllValues(TreeMap<Integer, Integer> tree) {
    //TODO: Add implementation
    return tree.values();
}
```

O(1) eftersom programmet hittar värdet på engång och behöver inte gå igenom alla värderna för att hitta rätt. Den bara hämtar det värdet på den platsen man letar på.