Trees Report.

Adrian Jonsson Sjödin

Fall 2022

Introduction

In this assignment we will take a closer look at *tree* structures, and in particular at tree structure called *binary trees*. The operations that we will look at and implement are: construction, adding and searching for and removing an item.

Task

Implement a binary tree structure that is ordered with smaller keys to the left and create the following two methods:

- `add(Integer key, Integer value)`: adds a new node (leaf) to the tree that maps the key to the value. If the key is already present we update the value of the node.
- `lookup(Integer key)`: find and return the value associate to the key. If the key is not found we return null.

benchmark the lookup algorithm and compare it to the benchmark of the binary search algorithm that was done in a previous assignment.

Also create an iterator that we can use to traverse the tree. The iterator should implements Java's Iterator class and override the `hasNext()` and `next()` method. Explain how you implemented the methods of the tree iterator class. Also describe what will happen if you create an iterator, retrieve a few elements snd then add new elements to the tree. Will it work, what is the state of the iterator, will we lose values?

Method & Theory

This data structure is called a *tree* since the structure originates in a root from where we spread out into branches, that then in turn can further divide into their own branches. A branch that does not divide further is terminated by a so called leaf.

A *binary tree* is a tree whose branch always divides into two branches, unless it terminates into a leaf.

Code Overview 1: Add integer to stack

```
public void add(int value) {
    Node newHead = new Node(value, this.head);
    this.head = newHead;
    this.size++;
}
```

Result

Discussion

All the code can be found here: GitHub