Examination Java Programming

Christian Rinderknecht

24 October 2013

1 Binary Search Trees

Consider the following classes implementing persistent binary search trees:

```
public class Pair<Fst,Snd> {
  protected final Fst fst;
  protected final Snd snd;
  public Pair (final Fst f, final Snd s) { fst = f; snd = s; }
  public Fst fst () { return fst; }
  public Snd snd () { return snd; }
}

public abstract class BST<Key extends Comparable<Key>> {
  public    abstract boolean isEmpty ();
  protected abstract Pair<Key,BST<Key>> min_aux (final Int<Key> p);
}

public final class Ext<Key extends Comparable<Key>> extends BST<Key>{
  public boolean isEmpty () { return true; }
  protected final Pair<Key,BST<Key>> min_aux (final Int<Key> p) {
    return new Pair<Key,BST<Key>>(p.root,p.right); }
}

public final class Int<Key extends Comparable<Key>>
       extends BST<Key> {
  protected final Key root;
  protected final BST<Key> left, right;
  public Int (final Key i, final BST<Key> l, final BST<Key> r) {
    root = i; left = l; right = r; }
  public boolean isEmpty () { return fakse; }
  public Pair<Key,BST<Key>> min_aux (final Int<Key> p) {
    Pair<Key,BST<Key>> m = left.min_aux(this);
    return new Pair<Key,BST<Key>>(m.fst(),
                          new Int<Key>(p.root,m.snd(),p.right));
  }
  public Pair<Key,BST<Key>> min () { return left.min_aux(this); }
}
```

Extend the classes `BST`, `Ext` and `Int` with a method `rm` (*remove*) which returns the same tree without a given key and whose signature is

```
public BST<Key> rm (final Key k);
```

Note: you must use the method `min_aux`.

2 Leaf trees

Using a functional style, design a binary tree whose leaves alone contain comparable keys, for instance numbers, while the other internal nodes do not. Such kind of tree is called a *leaf tree* and an example is shown in Figure 1. The nodes without information are simply called *nodes*, whereas the nodes carrying a comparable key are *leaves*.

Write a method `sort` which operates on a leaf tree and produces the nondecreasingly ordered stack of its leaves. The example in the figure yields the stack `(1,2,3,4)`, where the leftmost number is at the top of the stack. The signature is

```
public Stack<Key> sort ();
```

Give the best and worst costs with the corresponding configurations.

Note: you can reuse any of the methods of the class `Stack` we have defined during the course and you do not need to define insertion into the tree.

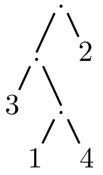
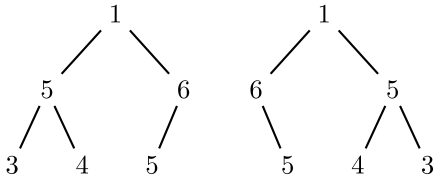


Figure 1: A leaf tree

3 Mirroring

Add a method `mirror` to the class `BST`, which takes a binary search tree and returns the same tree as in a mirror. Consider the facing example. What does the inorder traverval of the mirror yields?


1
5
6
3
4
5
1
6
5
5
4
3