Homework Week 3

Text Questions

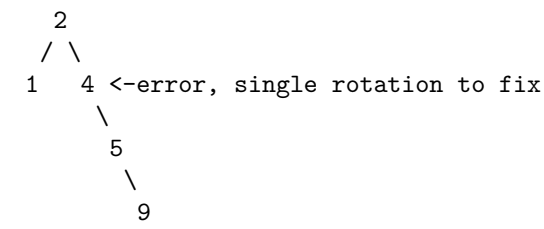
Monica Quaintance

mjq2102@columbia.edu

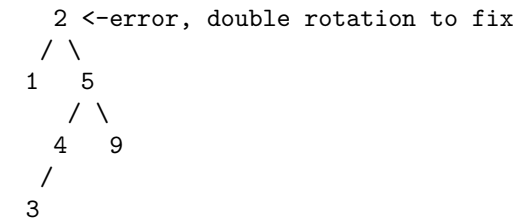
1. Weiss, Exercise 4.19

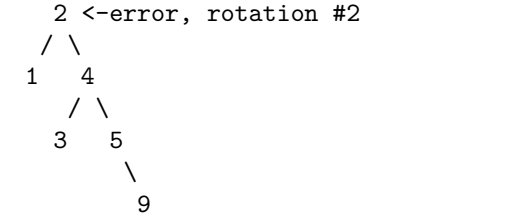
Show the result of inserting 2, 1, 4, 5, 9, 3, 6, 7 into an initially empty AVL tree.

insert 9:

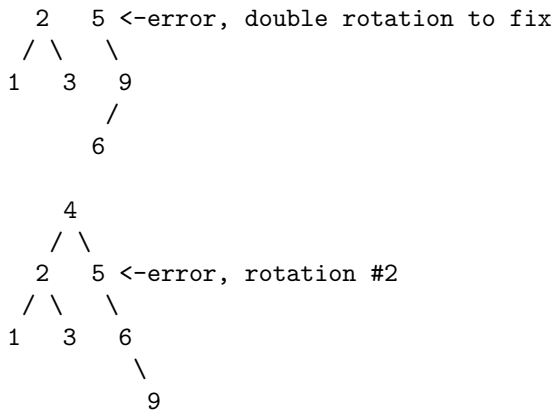


insert 3:

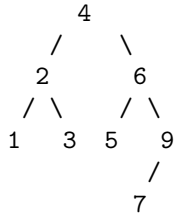




insert 6:



insert 7:



2. Weiss, Exercise 4.31

Write efficient methods that take only a reference to the root of a binary tree, T, and compute:

a. The number of nodes in T.

```
int nodeCount( treeNode t) {
  if (t==null)
    return 0;
  int nodeNum = nodeCount(t.left) + nodeCount(t.right) + 1;
  return nodeNum;
}
```

b. The number of leaves in T.

```
int leafCount( treeNode t) {
  if (t==null)
    return 0;
  else if (t.left == null && t.right == null)
```

```
    return 1;
  leafNum = leafCount(t.left) + leafCount(t.right);
  return leafNum;
}
```

c. The number of full nodes in T.

```
int fullCount( treeNode t) {
  if (t==null)
    return 0;

  int fullNum = 0;
  if (t.left != null && t.right != null )
    fullNum++;

  return fullNum + fullCount(t.left) + fullCount(t.right);
}
```

What is the running time of your routines?

All linear algos.

3. Prove that given a preorder and a postorder traversal of a binary

tree, the tree cannot be uniquely reconstructed.

"Binary tree" does not enforce ordering on the nodes. (Not BST)

Proof by counterexample, the following two binary trees have the same pre- and postorder traversals, respectively (3,2,1) and (1,2,3):

```
    3
   /
  2
 /
1

3
 \
  2
   \
    1
```

4. Weiss, Exercise 5.1

Given input {4371, 1323, 6173, 4199, 4344, 9679, 1989} and a hash function h(x) = x mod 10, show the resulting:

a. Separate chaining hash table.

```
0 []
1 [] - 4371
2 []
3 [] - 1323 - 6173
4 [] - 4344
5 []
6 []
7 []
8 []
9 [] - 4199 - 9679 - 1989
```

b. Hash table using linear probing.

```
0 [ 9679 ]
1 [ 4371 ]
2 [ 1989 ]
3 [ 1323 ]
4 [ 6173 ]
5 [ 4344 ]
6 []
7 []
8 []
9 [ 4199 ]
```

c. Hash table using quadratic probing.

```
0 [ 9679 ]
1 [ 4371 ]
2 []
3 [ 1323 ]
4 [ 6173 ]
5 [ 4344 ]
6 []
7 []
8 [ 1989 ]
9 [ 4199 ]
```

d. Hash table with second hash function h2(x) = 7 − (x mod 7).

```
0 []
1 [ 4371 ]
2 []
3 [ 1323 ]
4 [ 6173 ]
5 [ 9679 ]
6 []
7 [ 4344 ]
8 []
9 [ 4199 ]
```

1989 doesn't go, 5,1,7 and 3 are full.

5. Weiss, Exercise 5.2 - Use a new table size of 19, adjust the hash function accordingly.

Show the result of rehashing the hash tables in Exercise 5.1.

Showing non-null spaces:

a.

```
0 [] - 4199
1 [] - 4371
8 [] - 9679
12 [] - 1323 - 4344
13 [] - 1989
17 [] - 6173
```

b.

```
0 [ 4199 ]
1 [ 4371 ]
8 [ 9679 ]
12 [ 1323 ]
13 [ 1989 ]
14 [ 4344 ]
17 [ 6173 ]
```

c.

```
0 [ 4199 ]
1 [ 4371 ]
8 [ 9679 ]
```

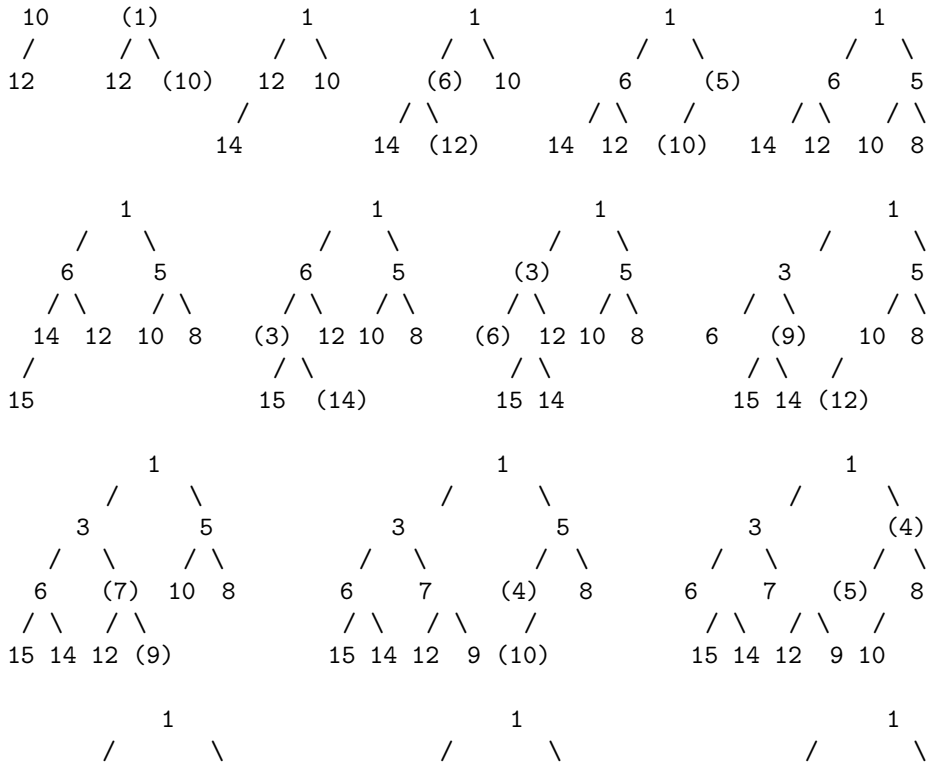
```
12 [ 1323 ]
13 [ 1989 ]
16 [ 4344 ]
17 [ 6173 ]
```

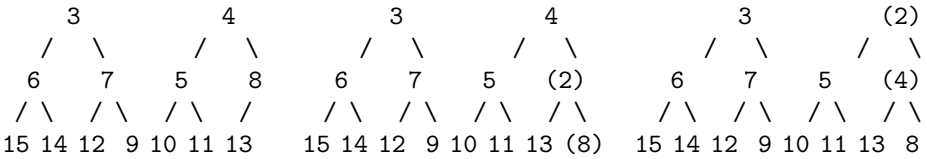
d.

```
0 [ 4199 ]
1 [ 4371 ]
8 [ 9679 ]
12 [ 1323 ]
13 [ 1989 ]
15 [ 4344 ]
17 [ 6173 ]
```

6. Weiss, Exercise 6.2 (show the contents of the heap after each step)

a. Show the result of inserting 10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13, and 2, one at a time, into an initially empty binary heap.





b. Show the result of using the linear-time algorithm to build a binary heap using the same input.

Dump all items into tree and percolate down from lowest interior nodes, up: