CSCI-UA.0310-004/005 Basic Algorithms March 30, 2015

Solutions to Problem 1 of Homework 6 (8 points)

Name: Jason Yao *Due: Wednesday, March 25*

Assume you are given a binary search tree T on n elements. Using a slight modification of the PostOrder-Tree-Walk procedure, argue (by writing the pseudocode!) that in time $\Theta(n)$ you can compute, for every node v, the number of nodes (call it $v.less$) *in v's sub-tree* which are less than v.
(**Hint**: In addition to $v.less$, also compute the total number of nodes in v's subtree.)

(a) (5 points) Start with the simpler case when all the elements are distinct.

Solution:

```
int lessNodeFind(node v)
  return lessNodeFindRec(v.left , v.value)
end lessNodeFind

int lessNodeFindRec(node node, int originalValue)
  if (node == null)
    return 0
  endif

  int L = lessNodeFindRec(node.left , originalValue)
  int R = lessNodeFindRec(node.right , originalValue)

  if (node.value < originalValue)
    return (1 + L + R)
  endif
  else
    return (L + R)
  endelse
END lessNodeFindRec
```

This will return all the number of nodes less than V in the tree, by traversing through V.left's subtree, and adding up all values less than V.value. □

(b) (3 points) Your solution for part (a) will probably not work if some of the elements of T could be the same. Show how to extend the solution for part (a) to handle this case. (Feel free to right away solve this general case for all 5+3=8 points.)

Solution:

```
int lessNodeFind(node v)
  return lessNodeFindRec(v.left, v.value)
end lessNodeFind

int lessNodeFindRec(node node, int originalValue)
  if (node == null)
    return 0
  endif

  int L = lessNodeFindRec(node.left, originalValue)
  int R = lessNodeFindRec(node.right, originalValue)

  if (node.value < originalValue)
    return (1 + L + R)
  endif
  else
    return (L + R)
  endelse
END lessNodeFindRec
```

□

CSCI-UA.0310-004/005 Basic Algorithms March 30, 2015

Solutions to Problem 2 of Homework 6 (6 points)

Name: Jason Yao *Due: Wednesday, March 25*

The PREORDER-TREE-WALK of a binary search tree T is: 6, 2, 1, 4, 3, 5, 7, 9, 8. Draw T.

Solution:

□

CSCI-UA.0310-004/005 Basic Algorithms March 30, 2015

Solutions to Problem 3 of Homework 6 (10 points)

Name: Jason Yao *Due: Wednesday, March 25*

Prove or show a counterexample for the following statements:[1]

(a) (5 points) For any binary search tree T and any element $x \notin T$, if one applies in sequence INSERT(x) followed by DELETE(x), then the result will be always T.

Solution:

True, the result will always be T, as INSERT(x) will always insert the value of x as a leaf, that means that DELETE(x) will always simply remove that leaf, leaving the rest of the tree the same as it was previously, since there are no rotations that need to be taken into account, such as in RedBlack or AVL trees. □

(b) (5 points) For any binary search tree T and any element $x \in T$, if one applies in sequence DELETE(x) followed by INSERT(x), then the result will be always T.

Solution:

False, as the following T shows, when x = 9:

□

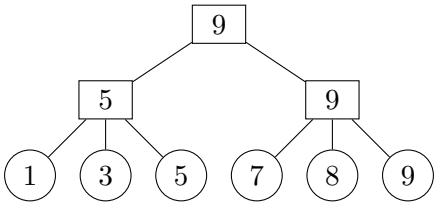
[1] If the statement is true, you need to give a general argument for any T. If false, you choose a specific T which illustrates the problem.

CSCI-UA.0310-004/005 Basic Algorithms March 30, 2015

Solutions to Problem 4 of Homework 6 (10 points)

Name: Jason Yao *Due: Wednesday, March 25*

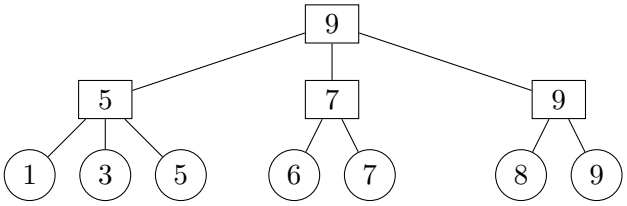
Consider the following 2-3 tree T:



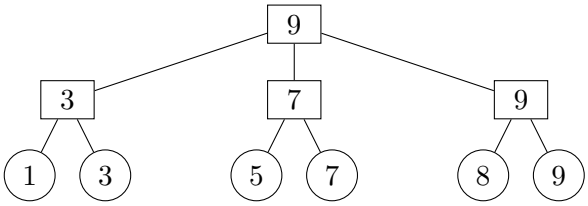
(a) (5 points) Show an element $x \notin T$ such that applying in sequence INSERT(x) and DELETE(x) will result with a tree T' that is different from T.

Solution:

For an element x = 6, the tree T' after insert will look like:



After deletion, the tree T' will look like:

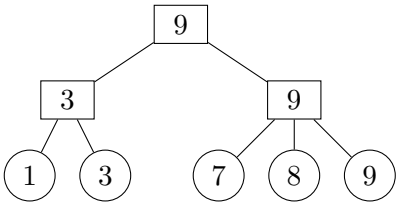


□

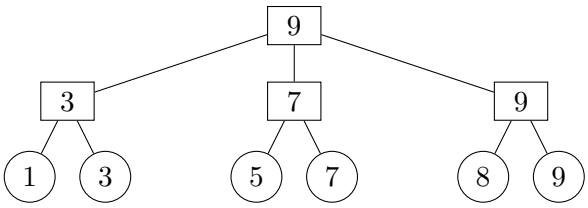
(b) (5 points) Show an element $x \in T$ such that applying in sequence DELETE(x) and INSERT(x) will result with a tree T' that is different from T.

Solution:

For an element x = 5, after deletion the tree T' will look like:



After insertion, the tree T' looks like:



□