Homework 10 sample solution

Due 10/05/16

September 30, 2016

Describe a modification of a Binary Search Tree dictionary that can return the minimum value in the dictionary in $\Theta(1)$ time. This change should *not* increase the asymptotic complexity of any other dictionary operation.

1. What new field(s) does the data structure need?

   Add a *min* field to the BST, which points to the node containing the *min* value. New BSTs should have $min = \text{NIL}$.

2. How does this change impact the min, insert, and delete methods of the BST? Note that insertion and deletion may change the minimum value in the BST.

Reference implementations of the min, insert, and delete functions appear below.

```
Algorithm: BSTDict.min()
return min
```

```
Algorithm: BSTDict.insert($new$)
$node = root$
while $node$ isn't NIL do
    if $node$.value $\leq new$ then
        if $node$.left $=$ NIL then
            Add $new$ as left child of $node$
            $node = node$.left
        end
        $node = node$.left
    else
        if $node$.right $=$ NIL then
            Add $new$ as right child of $node$
            $node = node$.right
        end
        $node = node$.right
    end
end
if $new < min.value$ then
    Let $min$ be the added node
end
```

```
Algorithm: BSTDict.delete($node$)
if $node = min$ then
    if $node$ has a right child then
        $min = node$.right
        while $min$ has a left child do
            $min = min$.left
        end
    else
        $min = parent$
    end
    /* $min = min$.successor() would also be acceptable */
end
if $node$ has two children then
    $swapnode = right$
    while $swapnode$ has a left child do
        $swapnode = swapnode$.left
    end
    Swap $node$'s parent and children links with $swapnode$
    if $node$ is the BST root then
        Set root to be $swapnode$
    end
end
if $node$ has no children then
    if $node$ is the root then
        Set root to be NIL
    else
        Set $node$.parent's child to be NIL
    end
else
    /* $node$ must have one child */
    if $node$ is the root then
        Set root to be $node$'s child
    else
        Set $node$.parent's child to be $node$'s child
    end
    Set $node$'s child's parent to be $node$.parent
end
```