Binary Search Trees

Binary Search Trees

Binary search tree property: a binary tree which is ordered such that every node's left subtree has only smaller values and right subtree has only larger (or equal) values

Useful operations: find, min, max, previous, next, insert, and delete.

Next

If no right child, want minimum ancestor ‘to the right’

```
NEXT($x$)
if right child exists then
    return min under right child
else
    return UP($x$)
```

```
UP($x$)
$p \leftarrow x.parent$
if $p = nil$ or $x = p.left$ then
    return $p$
else
    return UP($p$)
```

Insert

```
INSERT($n$)
if ... 
```

findParent

```
FINDPARENT($n$, curr, parent)
if curr is nil then
    return parent
if $n < curr$ then
    return FINDPARENT$(n, curr.left, curr)$
else
    return FINDPARENT$(n, curr.right, curr)$
```

Deletion Outline

3 cases of $delete(n)$:

1. no kids: pointer from parent ← nil
2. 1 child: substitute child for n at parent
3. 2 children: let `next(n)` be s
   a) s takes n's place at parent
   b) n's left subtree becomes s's
   c) somehow rest of n's right subtree becomes s's

let's split 3(c) into 2 cases…

Deletion Outline, Revised

4 cases of $delete(n)$:

1. no kids: pointer from parent ← nil
2. 1 child: substitute child for n at parent
3. s is n's right child:
   a) substitute s for n
   b) add n's left subtree as s's left subtree
4. s is deeper:
   a) substitute s's right subtree for s
   b) add n's right subtree as s's right subtree
   c) substitute s for n
   d) add n's left subtree as s's left subtree

Moving Subtrees

```
SUBSTITUTE($old, new$)
if $old$'s parent is nil then
    root $\leftarrow$ $new$
else
    if $old$ is parent's left child then
        parent's left child $\leftarrow$ $new$
    else
        parent's right child $\leftarrow$ $new$
if $new \neq$ nil then
    $new$'s parent $\leftarrow$ $old$'s parent
```

Deletion

```
DELETE($n$)
if $n$ has no left child then
    SUBSTITUTE($n$, $n$'s right subtree)
else if $n$ has no right child then
    SUBSTITUTE($n$, $n$'s left subtree)
else
    $s \leftarrow$ min in $n$'s right subtree
    if $n \neq s.parent$ then
        SUBSTITUTE($s$, $s$'s right subtree)
        $s$'s right subtree $\leftarrow$ $n$'s right subtree
        $s$'s right child's parent $\leftarrow s$
    SUBSTITUTE($n, s$)
    $s$'s left subtree $\leftarrow$ $n$'s left subtree
    $s$'s left child's parent $\leftarrow s$
```

Balanced Trees

Structure	Find	Insert	Delete
List (sorted)			
BST (unbalanced)			
BST (balanced)			

Reminder: maintaining a balanced binary tree is important