Delete BST: (p 295 – 298)

- BST: delete z, 3 cases:
  1. z has no children
  2. z has 1 child
  3. z has 2 children
Delete BST: Cases 1 and 2

- Case 1: Remove z, in parent replace z with NIL
- Case 2: z has one child r or l: Elevate child

\[ 
\begin{array}{c}
\text{Case 1: Remove z, in parent replace z with NIL} \\
\text{Case 2: z has one child r or l: Elevate child} \\
\end{array}
\]

- Replace z by r or l (transplant tree whose root is r or l to z)

Delete BST: Case 3, 3a

- Case 3: z has two children, y is z's successor (lies in right subtree and which has no left child)
  - Case 3a: y is z's right child:

\[ 
\begin{array}{c}
\text{Case 3: z has two children, y is z’s successor (lies in right subtree and which has no left child)} \\
\text{Case 3a: y is z’s right child:} \\
\end{array}
\]

Elevate y, and set \( y\text{.left} = l, \ l\text{.p} = y \)
Delete BST: Case 3b

- Case 3: z has two children, y is z’s successor (lies in right subtree and which has no left child)
- Case 3b: y is not z’s right child, but lies farther down right tree rooted at right child r

- Surgery! (replace y by its right child x, replace z by y)

Delete: Helper Functions

Page 295

<table>
<thead>
<tr>
<th>BST-Transplant(T, u, v)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 if u.p == NIL</td>
<td></td>
</tr>
<tr>
<td>2 T.root = v</td>
<td></td>
</tr>
<tr>
<td>3 elseif u == u.p.left</td>
<td></td>
</tr>
<tr>
<td>4 u.p.left = v</td>
<td></td>
</tr>
<tr>
<td>5 else u.p.right = v</td>
<td></td>
</tr>
<tr>
<td>6 if v != Nil</td>
<td></td>
</tr>
<tr>
<td>7 v.p = u.p</td>
<td></td>
</tr>
</tbody>
</table>

Page 323

<table>
<thead>
<tr>
<th>RB-Transplant(T, u, v)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 if u.p == T.NIL</td>
<td></td>
</tr>
<tr>
<td>2 T.root = v</td>
<td></td>
</tr>
<tr>
<td>3 elseif u == u.p.left</td>
<td></td>
</tr>
<tr>
<td>4 u.p.left = v</td>
<td></td>
</tr>
<tr>
<td>5 else u.p.right = v</td>
<td></td>
</tr>
<tr>
<td>6 v.p = u.p // Always!</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(T.nil ok)</td>
</tr>
</tbody>
</table>
BST Delete, p 323

**TREE-DELETE(T,z)**

1. if `z.left == NIL`  // no left child
2. TRANSPLANT(T,z,z.right)  // elevate right child
3. elseif `z.right == NIL`  // no right child
4. TRANSPLANT(T,z,z.left)  // elevate left child
5. else `y = TREE-MINIMUM(z.right)`  // successor to z
6. if `y.p != z`  // y is NOT z’s (right) child
7. TRANSPLANT(T,y,y.right)// Case 3b
8. `y.right = z.right`
9. `y.right.p = y`
10. TRANSPLANT(T,z,y)  // Otherwise, y IS z’s right child
11. `y.left = z.left`  // Case 3a
12. `y.left.p = y`

RB-DELETE(T,z), p 323

- RB-Transplant
- RB-Delete
- RB-DeleteFixup

**NOTE:** x is the node that moves into y’s position.
RB-DELETE-FIXUP(T, x) p 323

w is x’s sibling

Case 1: w is red

Case 2: w is black, w’s children are black

Case 3: w is red, w’s left child red, w right child black

Case 4: w is black, both w’s children are red

Examples
Delete each of these from the original
Delete 26
Delete 22
Delete 10
Delete 18
Delete 3