AVL Deletions

Deletions in AVL trees can be handled in a similar way as insertions, though it’s a little more complicated.

First, do a standard BST deletion. Keep track of where a leaf node was (ultimately) removed and fix up the heights above that point.

If the deletion created an imbalance, apply a suitable fix to the tree...

Case #1: deletion → left-left

Delete child on the right
Case #1: deletion $\rightarrow$ left-left

\[
\begin{array}{c}
\text{h+3} \\
\text{h+2} \\
\text{b} \\
\hline
\text{h+1} \\
X \\
\hline
\text{h} \\
\text{Y} \\
\hline
\text{h} \\
Z
\end{array}
\]

Single rotation

\[
\begin{array}{c}
\text{h+3} \\
\text{h+2} \\
\text{a} \\
\hline
\text{h+1} \\
\text{b} \\
\hline
\text{h+1} \\
X \\
\hline
\text{h} \\
\text{Y} \\
\hline
\text{h} \\
Z
\end{array}
\]

Case #2: deletion $\rightarrow$ left-right

\[
\begin{array}{c}
\text{h+3} \\
\text{h+2} \\
\text{a} \\
\hline
\text{h+1} \\
\text{b} \\
\hline
\text{Z}
\end{array}
\]

Delete child on the right

\[
\begin{array}{c}
\text{h+3} \\
\text{h+2} \\
\text{a} \\
\hline
\text{h+1} \\
\text{b} \\
\hline
\text{Z}
\end{array}
\]

Another deletion case?

\[
\begin{array}{c}
\text{h+3} \\
\text{h+2} \\
\text{a} \\
\hline
\text{h+1} \\
\text{b} \\
\hline
\text{Z}
\end{array}
\]

Delete child on the right

\[
\begin{array}{c}
\text{h+3} \\
\text{h+2} \\
\text{a} \\
\hline
\text{h+1} \\
\text{b} \\
\hline
\text{Z}
\end{array}
\]
Another deletion case?

Other cases

There are of course two more mirror image cases: right-left and right-right.

Let’s try an example...

Deletion example

Deletion example
Finishing up deletions

After a deletion, we update the heights and fix the first problem we find. Then...?

*Keep going up, looking for problems*

What is the complexity of doing a deletion?

$O(\log n)$