Map

BSTMap<K,V,I>

Node

<<interface>>

implements
RealNode

key: K
value: V
left, right: Node
info: I

Balancer<K,V,I>

<<interface>>

putFixup(Node): Node
removeFixup(Node): Node
rootFixup(Node)
newInfo(Node): I

NodeInfo

<<interface>>

recompute()

AVLInfo
RBInfo

AVLBalancer
TradRBBalancer
LLRBBalancer
right-right alone:

\[
\begin{align*}
&\text{A} \\
&\quad \alpha \\
&\quad \beta \\
&\quad \gamma \\
&\text{B} \\
&\quad \text{A} \\
&\quad \alpha \\
&\quad \beta \\
&\quad \gamma \\
&\text{B} \\
\end{align*}
\]

rotate

\[
\begin{align*}
&\text{A} \\
&\quad \alpha \\
&\quad \beta \\
&\quad \gamma \\
&\text{B} \\
\end{align*}
\]
right–left alone:

A

C

B

α

β γ

δ

A

α

C

B

β

γ δ

[−2]

(h)

(h+3)

[1]

(h+2)

[−1 or 0 or 1]

(h or h−1)

(h or h+1)

(h or h−1) β

γ (h or h−1)

[−2]

(h)

[0]

(h+2)

[h or h−1]

[h+1]

[1 or 0]

(h+1)

[h+1]

[−1 or −2]

(h or h−1)

[−1 or 0]

(h+1)

[0]

 rotated

A

B

C

α

β

γ

δ

[−1 or 0]

(h or h−1)

[−1 or −2]

(h+1)

[0]

(rotated)

A

B

C

α

β

γ

δ

[−1 or 0]

(h or h−1)

[−1 or −2]

(h+1)

[0]
right-left:

```
right-left:  
```

right-right:

```
right-right:  
```
Invariant 25 (Postconditions of `RealNode.put()` with AVLBalancer.)
Let $x$ be the root of a subtree on which `put()` is called and $y$ be the node returned, that is, the root of the resulting subtree. The subtree rooted at $y$ has no violations and the height of the subtree rooted at $y$ is equal to or one greater than the original height of the subtree rooted at $x$.

**Proof.** Suppose `put()` is called on node $x$ in a BST using AVL balancing which has no violations. There are three cases: $x$ is `null`, $x$ is a `RealNode` containing the key being searched for, or $x$ is a `RealNode` with a different key. We use structural induction with the first two cases as base cases.
Base case 1. Suppose $x$ is $\texttt{null}$, which has height 0. Then the node $y$ returned is a new $\texttt{RealNode}$ with $\texttt{null}$ as both children, height 1, and balance 0. The subtree rooted at $y$ has no violations and height one greater than the original height of $x$.

Base case 2. Suppose $x$ is a $\texttt{RealNode}$ whose key is equal to the key used for this $\texttt{put()}$. Then the value at node $x$ is overwritten but node $x$ itself is returned (so $y = x$ in this case) with the tree structure unchanged.

Inductive case. Suppose $x$ is a $\texttt{RealNode}$ and, without loss of generality, the key used for this $\texttt{put()}$ is greater than the key at $x$, and so $\texttt{put()}$ is called on the right child of $x$. Let $h_0$ be the height of the tree rooted at $x$ before this call to $\texttt{put()}$ on the right child, and let $z$ the root of the subtree that results from this call to $\texttt{put()}$ on the right child. Our inductive hypothesis is that the subtree rooted at $z$ has no violations and that its height is equal to or one greater than the height of the original right child of $x$. 
Let $h_1$ be the height of the tree rooted at $x$ after the call to \texttt{put()} on the right child but before the call to \texttt{putFixup()} with $x$.

Since since at most the height of its right subtree has increased by one, either $h_1 = h_0$ or $h_1 = h_0 + 1$. By supposition, the balance of $x$ before the call to \texttt{put()} was no less than $-1$, since we supposed the tree had no violations. Since at most the height of its right subtree has increased by one, the balance of $x$ is now no less than $-2$. We now have two subcases: Either the balance of $x$ is greater than $-2$ or it is equal to $-2$.

Suppose the balance of $x$ is greater than $-2$. Then the call to \texttt{putFixup()} with $x$ returns $x$ unchanged, which is also returned as the result of \texttt{put()} (again $y = x$), a tree with no violations and height $h_1$.

On the other hand, suppose the balance of $x$ is equal to $-2$. Then $y$ is a node other than $x$ returned by \texttt{putFixup()}. Let $h_2$ be the height of the subtree rooted at $y$ when \texttt{putFixup()} returns. By inspection of the right-right and right-left subcases given above, the subtree rooted at $y$ has no violations and either $h_2 = h_1$ or $h_2 = h_1 - 1$. In either of those cases $h_2 = h_0$ or $h_2 = h_0 + 1$. □
\[ B_h = \begin{cases} 
1 & \text{if } h = 1 \\
2 & \text{if } h = 2 \\
B_{h-2} + B_{h-1} + 1 & \text{otherwise}
\end{cases} \]

\[ B_{h+1} = \begin{cases} 
2 & \text{if } h = 1 \\
3 & \text{if } h = 2 \\
(B_{h-2} + 1) + (B_{h-1} + 1) & \text{otherwise}
\end{cases} \]
\[ B_h + 1 > \frac{\phi^{h+2}}{\sqrt{5}} - 1 \]

\[ B_h + 2 > \frac{\phi^{h+2}}{\sqrt{5}} \]

\[ \sqrt{5}(B_h + 2) > \phi^{h+2} \]

\[ h + 2 < \log_\phi(\sqrt{5}B_h) \]

\[ h < \log_\phi(\sqrt{5}B_h) - 2 \]

\[ = \log_\phi B_h + \log_\phi \sqrt{5} - 2 \]

\[ = \frac{1}{\lg \phi} \lg B_h + \log_\phi \sqrt{5} - 2 \]