BB[$\alpha$] tree — definition

- Let $\text{size}(v)$ be the number of nodes in the subtree of $v$.
- A node $v$ is of bounded balance $\alpha$ if
  \[
  \text{size}(v.\text{left}) \geq \lfloor \alpha \cdot \text{size}(v) \rfloor \quad \text{and} \quad \text{size}(v.\text{right}) \geq \lfloor \alpha \cdot \text{size}(v) \rfloor
  \]
- A BB[$\alpha$] tree ($\alpha < 0.5$) is a binary search tree such that every node $v$ is of bounded balance $\alpha$.
- The height of a BB[$\alpha$] tree with $n$ nodes is at most $\log_{1/(1-\alpha)} n$.

\[
\alpha = \frac{1}{3}
\]
To insert an element to a BB[\( \alpha \)] tree, insert it as a new leaf. Let \( v \) be the highest node that is not of bounded balance \( \alpha \). If \( v \) exists, replace the subtree of \( v \) by a balanced tree containing the same elements.
The time complexity is \( \Theta(\log n + \text{size}(v)) \), which is \( \Theta(n) \) in the worst case.
To insert an element to a BB[$\alpha$] tree, insert it as a new leaf. Let $v$ be the highest node that is not of bounded balance $\alpha$. If $v$ exists, replace the subtree of $v$ by a balanced tree containing the same elements.

The time complexity is $\Theta(\log n + \text{size}(v))$, which is $\Theta(n)$ in the worst case.
Insertion

- To insert an element to a BB[$\alpha$] tree, insert it as a new leaf. Let $v$ be the highest node that is not of bounded balance $\alpha$. If $v$ exists, replace the subtree of $v$ by a balanced tree containing the same elements.
- The time complexity is $\Theta(\log n + \text{size}(v))$, which is $\Theta(n)$ in the worst case.

$\alpha = 1/3$
Amortized complexity

- The actual cost of an insert operation is 
  \( \text{depth}(u) + \text{size}(w) \), where \( u \) is the new leaf, and \( w \) is the 
  node whose subtree was replaced (\( \text{depth}(u) \) is the depth 
  after the first stage).

**Claim**

The amortized cost of insert is 
\[
(1 + \frac{1}{1-2\alpha}) \log_{1/(1-\alpha)} n + O(1).
\]

- Let \( \Delta_v = |\text{size}(v.\text{left}) - \text{size}(v.\text{right})| \).
- We keep the following invariant: Every node \( v \) with 
  \( \Delta_v \geq 2 \) stores \( \frac{1}{1-2\alpha} \Delta_v \) dollars.
- For the first stage of an insert operation, we use at most 
  \( \log_{1/(1-\alpha)} n \) charged dollars to pay for the cost \( \text{depth}(u) \). 
  We also put \( \frac{1}{1-2\alpha} \) dollars in each node \( v \) whose \( \Delta_v \) value 
  increased. This uses at most \( \frac{1}{1-2\alpha} \log_{1/(1-\alpha)} n \) charged 
  dollars.
Amortized complexity

The figure below shows the $\Delta_v$ values. Each node with $\Delta_v \geq 2$ stores $3\Delta_v$ dollars.
Amortized complexity

The figure below shows the $\Delta_v$ values. Each node with $\Delta_v \geq 2$ stores $3\Delta_v$ dollars. After the insertion, we need to add 3 dollars to the root and to the left child of the root.
To pay for the second stage of an insert operation, we use the dollars stored in \( w \) if \( \text{size}(w) \geq \frac{1}{1 - 2\alpha} \) (if \( \text{size}(w) < \frac{1}{1 - 2\alpha} \) we use the charged dollars).

Since \( w \) is not of bounded balance \( \alpha \) after the first stage, either

\[
\text{size}(w.\text{left}) \leq \lfloor \alpha \cdot \text{size}(w) \rfloor - 1 \leq \alpha \cdot \text{size}(w) - 1 \\
\text{size}(w.\text{right}) \geq \text{size}(w) - (\alpha \cdot \text{size}(w) - 1) - 1
\]

or vice versa.

Thus,

\[
\Delta_w \geq (1 - 2\alpha) \cdot \text{size}(w) + 1 \geq 2
\]

so \( w \) contains at least \( \text{size}(w) - \frac{1}{1-2\alpha} = \text{size}(w) - O(1) \) dollars.