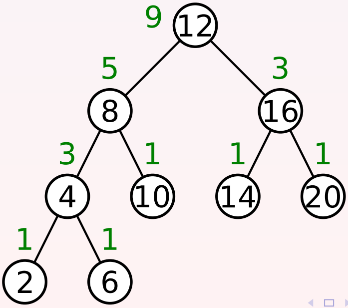
BB[α] tree — definition

- Let $\mathrm{size}(v)$ be the number of nodes in the subtree of v.
- A node v is of bounded balance α if

  $$\mathrm{size}(v.\mathrm{left}) \geq \lfloor \alpha \cdot \mathrm{size}(v) \rfloor \text{ and } \mathrm{size}(v.\mathrm{right}) \geq \lfloor \alpha \cdot \mathrm{size}(v) \rfloor$$

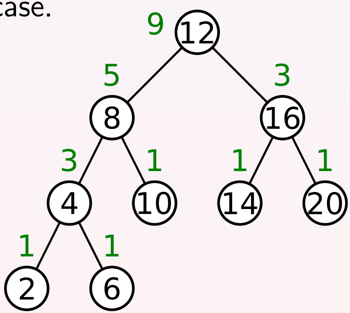
- A BB[α] tree ($\alpha < 0.5$) is a binary search tree such that every node v is of bounded balance α
- The height of a BB[α] tree with n nodes is at most $\log_{1/(1-\alpha)} n$.



$\alpha = 1/3$

Insertion

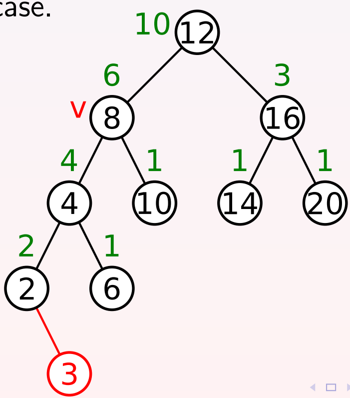
- To insert an element to a BB[α] tree, insert it as a new leaf. Let *v* be the highest node that is not of bounded balance α. If *v* exists, replace the subtree of *v* by a balanced tree containing the same elements.
- The time complexity is $\Theta(\log n + \mathrm{size}(v))$, which is $\Theta(n)$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



$\alpha = 1/3$

Insertion

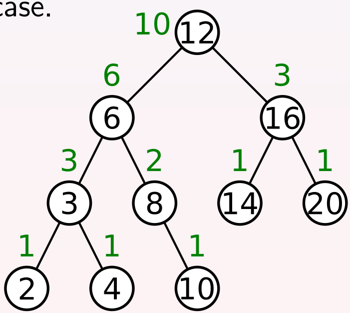
- To insert an element to a BB[α] tree, insert it as a new leaf. Let v be the highest node that is not of bounded balance α. If v exists, replace the subtree of v by a balanced tree containing the same elements.
- The time complexity is $\Theta(\log n + \mathrm{size}(v))$, which is $\Theta(n)$ in the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



$\alpha = 1/3$

Insertion

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



$\alpha = 1/3$

Amortized complexity

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

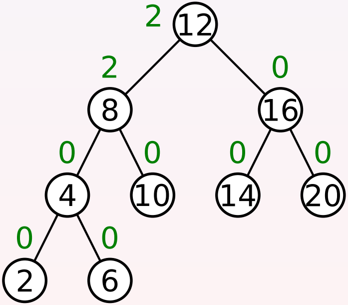
Claim

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

- Let $\Delta_v = |\mathrm{size}(v.\mathrm{left}) - \mathrm{size}(v.\mathrm{right})|$.
- 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 $\mathrm{depth}(u)$. We also put $\frac{1}{1-2\alpha}$ dollars in each node v whose Δ_v value increased. This uses at most $\frac{1}{1-2\alpha} \log_{1/(1-\alpha)} n$ charged dollars.

Amortized complexity

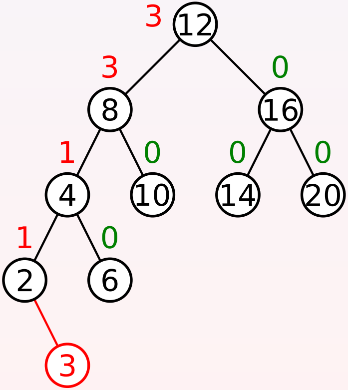
The figure below shows the Δ_v values. Each node with $\Delta_v \geq 2$ stores $3\Delta_v$ dollars.



$\alpha = 1/3$

Amortized complexity

The figure below shows the Δ_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.



$\alpha = 1/3$

Amortized complexity

- To pay for the second stage of an insert operation, we use the dollars stored in w if $\mathrm{size}(w) \geq 1/(1-2\alpha)$ (if $\mathrm{size}(w) < 1/(1-2\alpha)$ we use the charged dollars).
- Since w is not of bounded balance α after the first stage, either

$$\mathrm{size}(w.\mathrm{left}) \leq \lfloor \alpha \cdot \mathrm{size}(w) \rfloor - 1 \leq \alpha \cdot \mathrm{size}(w) - 1$$
$$\mathrm{size}(w.\mathrm{right}) \geq \mathrm{size}(w) - (\alpha \cdot \mathrm{size}(w) - 1) - 1$$

  or vice versa.
- Thus,

$$\Delta_w \geq (1-2\alpha) \cdot \mathrm{size}(w) + 1 \geq 2$$

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