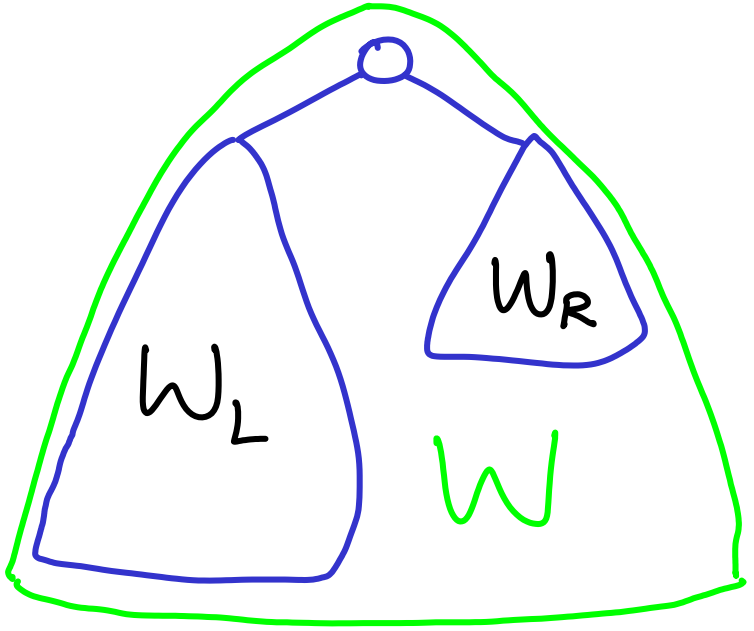
Binary search trees of bounded balance α → BB(α)

(weight-balanced trees)

Binary search trees of bounded balance α → BB(α)

(weight-balanced trees)

node with subtree weight W and
- left subtree weight W_L
- right subtree weight W_R

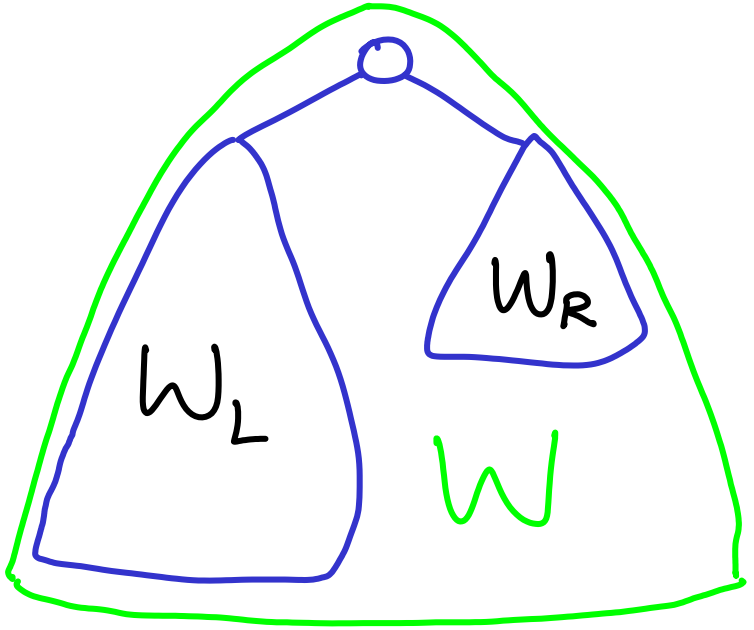


Binary search trees of bounded balance α → BB(α)

(weight-balanced trees)

For every node with subtree weight W and
- left subtree weight W_L
- right subtree weight W_R

$$W_L \geq \alpha W$$
$$W_R \geq \alpha W$$
$$0 < \alpha \leq \frac{1}{2}$$



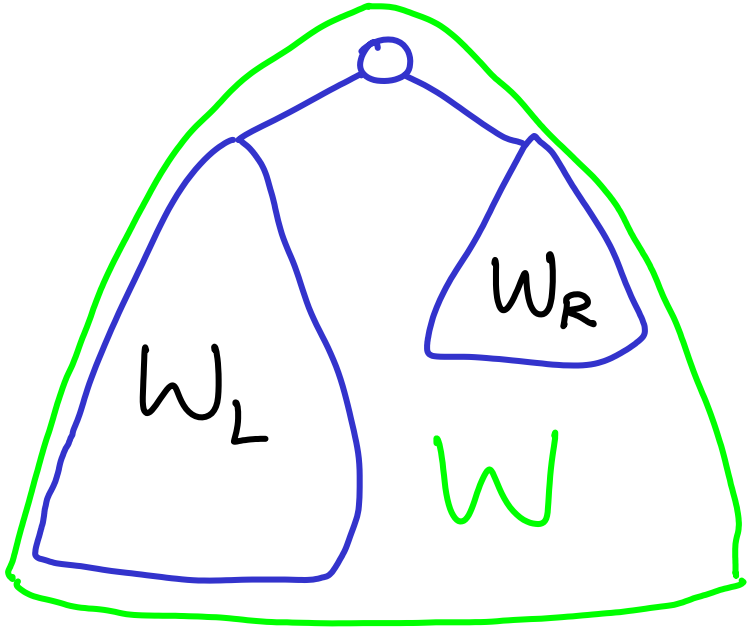
Binary search trees of bounded balance α → BB(α)

(weight-balanced trees)

For every node with subtree weight W and
- left subtree weight W_L
- right subtree weight W_R

$$W_L \geq \alpha W$$
$$W_R \geq \alpha W$$
$$0 < \alpha \leq \frac{1}{2}$$

Height: ?



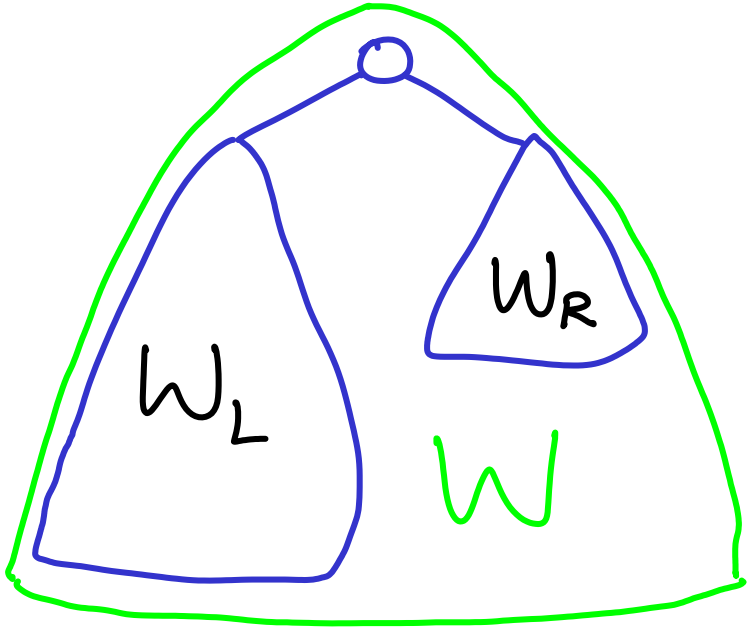
Binary search trees of bounded balance α → BB(α)

(weight-balanced trees)

For every node with subtree weight W and
- left subtree weight W_L
- right subtree weight W_R

$$W_L \geq \alpha W$$
$$W_R \geq \alpha W$$
$$0 < \alpha \leq \frac{1}{2}$$

Height: $H(n) \leq 1 + H((\underbrace{1-\alpha}_{\geq \frac{1}{2}})n)$



Binary search trees of bounded balance α → BB(α)

(weight-balanced trees)

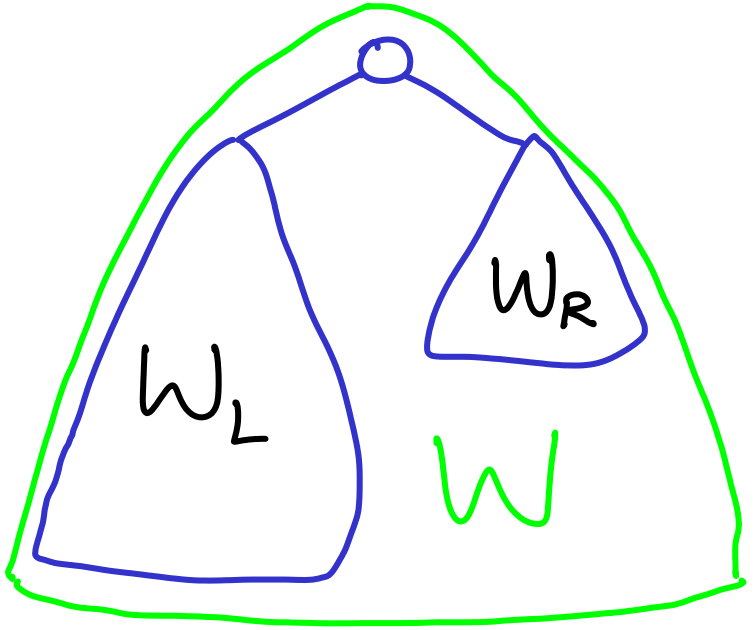
For every node with subtree weight W and
- left subtree weight W_L
- right subtree weight W_R

$$W_L \geq \alpha W$$
$$W_R \geq \alpha W$$
$$0 < \alpha \leq \frac{1}{2}$$

Height: $H(n) \leq 1 + H((1-\alpha)n)$, where $(1-\alpha) \geq \frac{1}{2}$

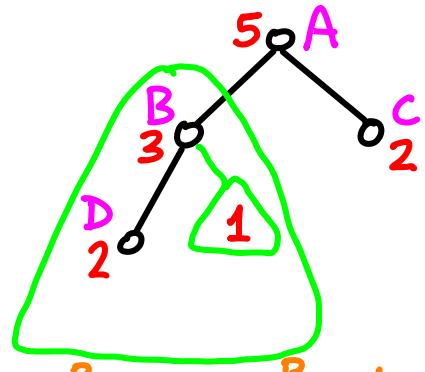
standard geometric series:

$H \sim \log_{1/(1-\alpha)} n = \Theta(\log n)$



Note on counting convention: $W = 1 + \#\text{nodes}$

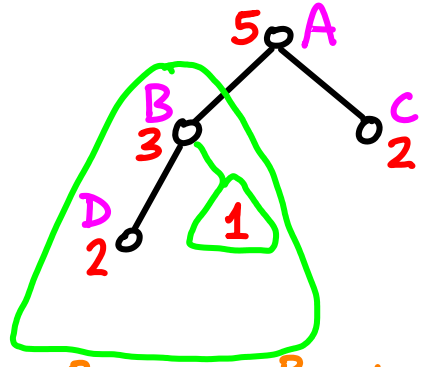
not universally accepted



$W^B = 3$, $W_R^B = 1$

Note on counting convention: $W = 1 + \#\text{nodes}$

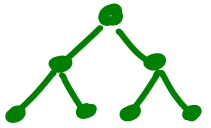
not universally accepted



$W^B = 3$, $W_R^B = 1$

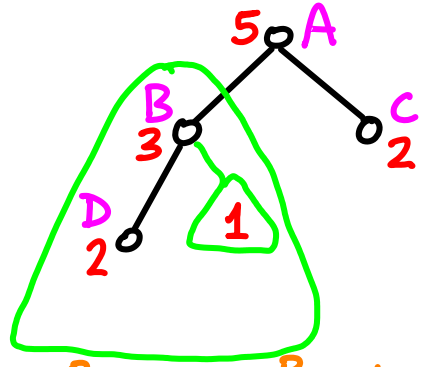
Interesting & strange: For all $\frac{1}{3} < \alpha < \frac{1}{2}$:

if a tree is $BB(\alpha)$ then it is $BB(\frac{1}{2})$



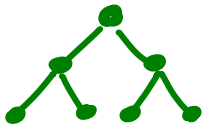
Note on counting convention: $w = 1 + \#\text{nodes}$

not universally accepted



$w^B = 3, \; w_R^B = 1$

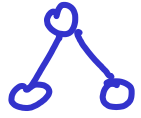
Interesting & strange : For all $\frac{1}{3} < \alpha < \frac{1}{2}$:



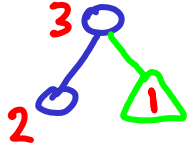
if a tree is $BB(\alpha)$ then it is $BB(\frac{1}{2})$

1) Clearly true for single node

2)



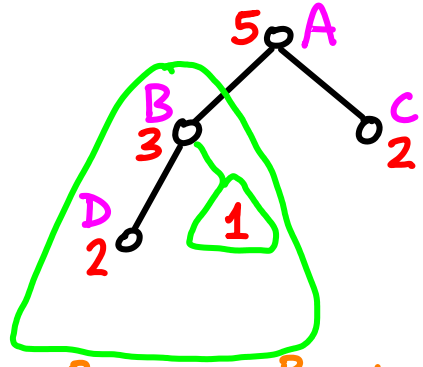
trivial



$\alpha \leq \frac{1}{3}$

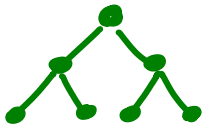
Note on counting convention: $W = 1 + \#\text{nodes}$

not universally accepted



$W^B = 3,\ W^B_R = 1$

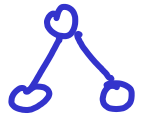
Interesting & strange : For all $\frac{1}{3} < \alpha < \frac{1}{2}$:



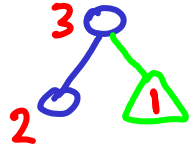
if a tree is $BB(\alpha)$ then it is $BB(\frac{1}{2})$

1) Clearly true for single node

2)



trivial

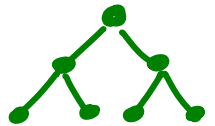


$\alpha \leq \frac{1}{3}$

} Proof by smallest counterexample...

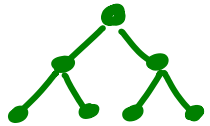
by tree height

For all $\frac{1}{3} < \alpha < \frac{1}{2}$: if a tree is BB($\alpha$) then it is BB($\frac{1}{2}$)



True for height 1 or 2.

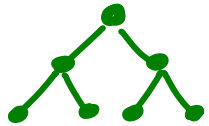
For all $\frac{1}{3} < \alpha < \frac{1}{2}$: if a tree is BB($\alpha$) then it is BB($\frac{1}{2}$)



True for height 1 or 2.

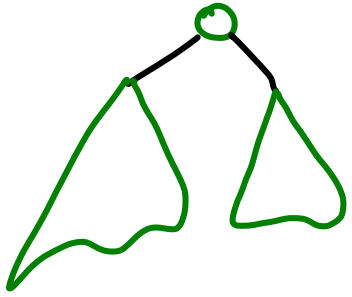
Suppose h = height of shortest tree where claim fails ($h > 2$)

For all $\frac{1}{3} < \alpha < \frac{1}{2}$: if a tree is BB($\alpha$) then it is BB($\frac{1}{2}$)



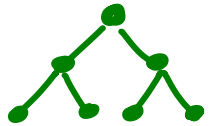
True for height 1 or 2.

Suppose h = height of shortest tree where claim fails ($h > 2$)



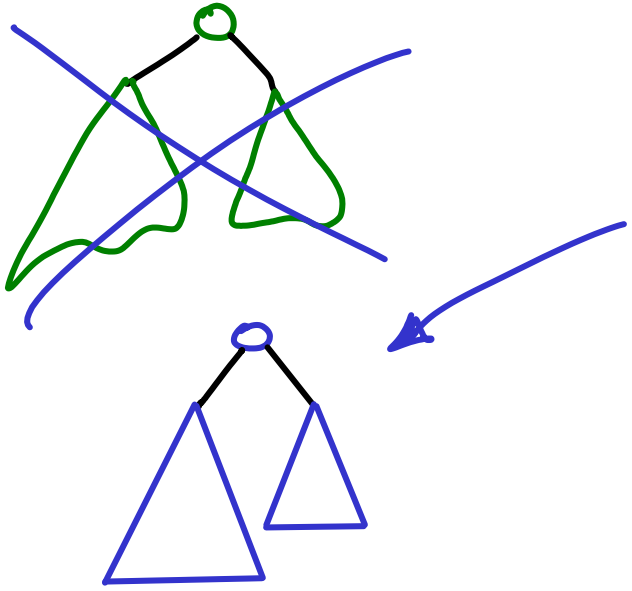
$\exists$ tree that is $(>\frac{1}{3})$-balanced but not $\frac{1}{2}$-balanced

For all $\frac{1}{3} < \alpha < \frac{1}{2}$: if a tree is $BB(\alpha)$ then it is $BB(\frac{1}{2})$



True for height 1 or 2.

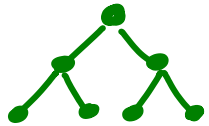
Suppose h = height of shortest tree where claim fails $(h > 2)$



$\exists$ tree that is $(>\frac{1}{3})$-balanced but not $\frac{1}{2}$-balanced

Subtrees are shorter, so they are $\frac{1}{2}$-balanced

For all $\frac{1}{3} < \alpha < \frac{1}{2}$: if a tree is BB($\alpha$) then it is BB($\frac{1}{2}$)

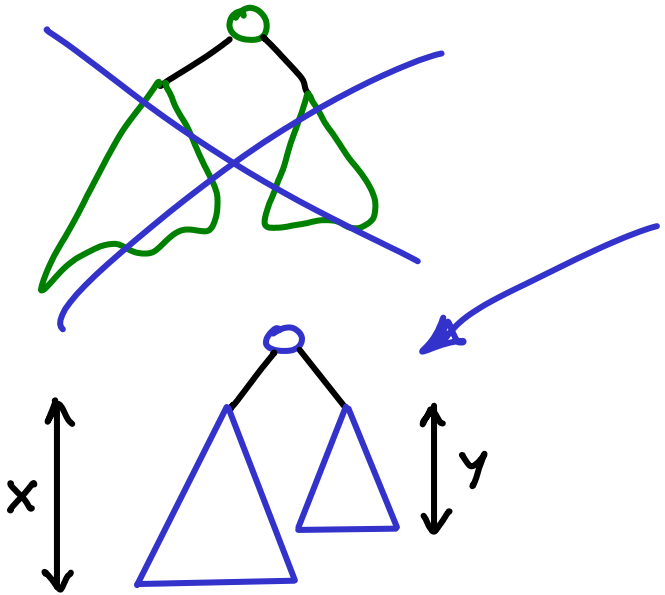


True for height 1 or 2.

Suppose h = height of shortest tree where claim fails ($h > 2$)

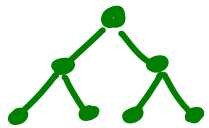
$\exists$ tree that is $(>\frac{1}{3})$-balanced but not $\frac{1}{2}$-balanced

Subtrees are shorter, so they are $\frac{1}{2}$-balanced



$x > y$

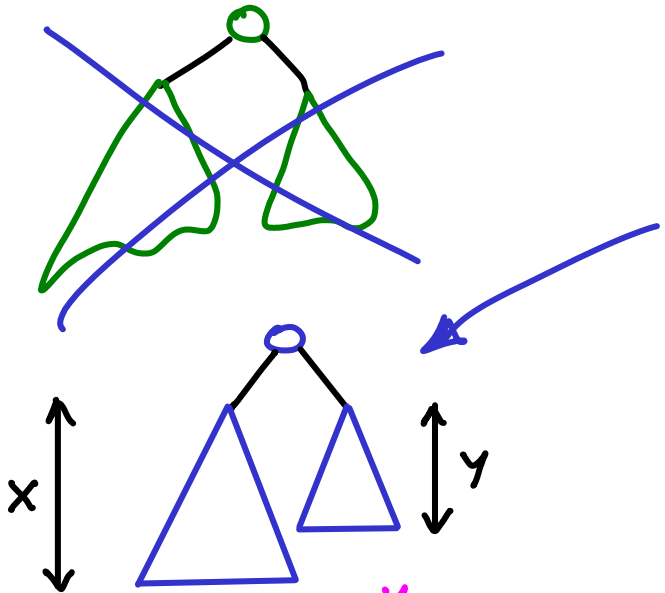
For all $\frac{1}{3} < \alpha < \frac{1}{2}$: if a tree is BB($\alpha$) then it is BB($\frac{1}{2}$)



True for height 1 or 2.

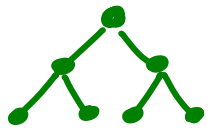
Suppose h = height of shortest tree where claim fails ($h > 2$)

$\exists$ tree that is $(>\frac{1}{3})$-balanced but not $\frac{1}{2}$-balanced



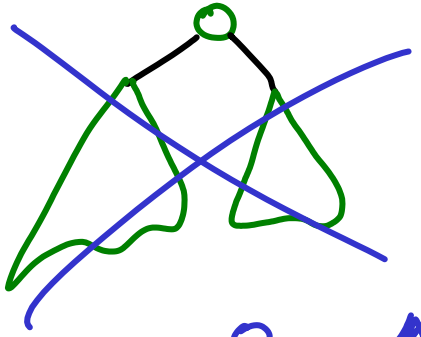
Subtrees are shorter, so they are $\frac{1}{2}$-balanced

For all $\frac{1}{3} < \alpha < \frac{1}{2}$: if a tree is BB($\alpha$) then it is BB($\frac{1}{2}$)



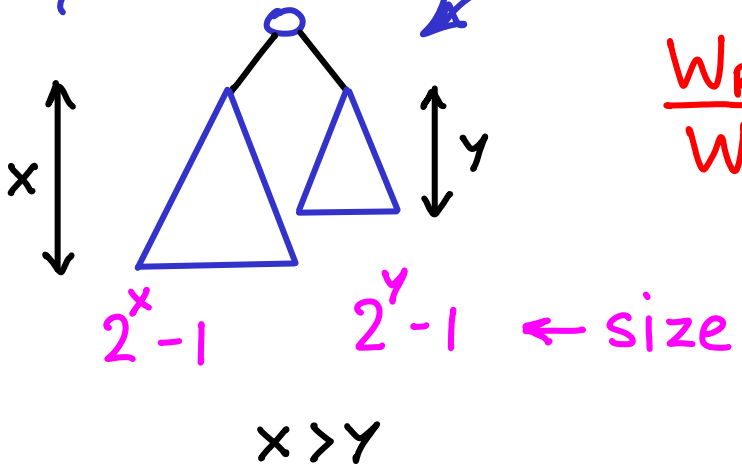
True for height 1 or 2.

Suppose h = height of shortest tree where claim fails ($h > 2$)



$\exists$ tree that is $(>\frac{1}{3})$-balanced but not $\frac{1}{2}$-balanced

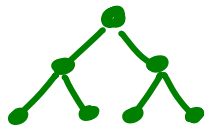
Subtrees are shorter, so they are $\frac{1}{2}$-balanced



$$\frac{W_R}{W} = \frac{(2^y - 1) + 1}{(2^x - 1) + (2^y - 1) + 1 + 1}$$

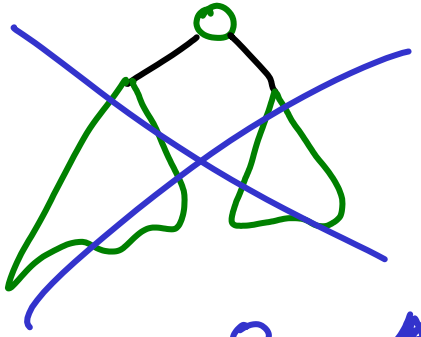
$(2^x - 1)$: L; $(2^y - 1)$: R; $+1$: root; $+1$: convention

For all $\frac{1}{3} < \alpha < \frac{1}{2}$: if a tree is BB($\alpha$) then it is BB($\frac{1}{2}$)



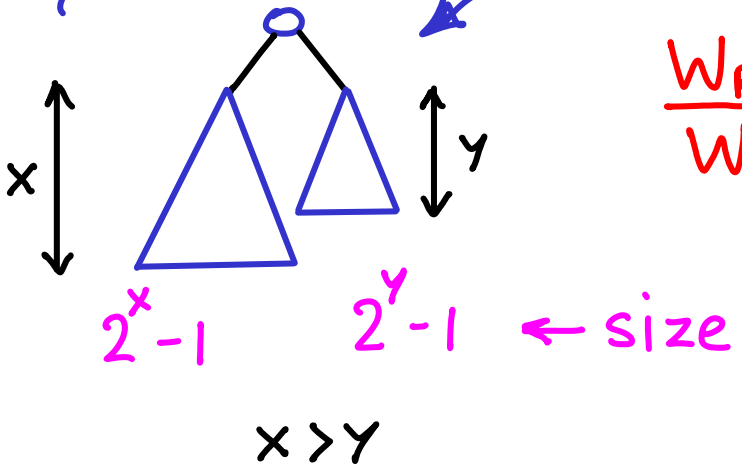
True for height 1 or 2.

Suppose h = height of shortest tree where claim fails ($h > 2$)



$\exists$ tree that is $(>\frac{1}{3})$-balanced but not $\frac{1}{2}$-balanced

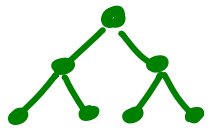
Subtrees are shorter, so they are $\frac{1}{2}$-balanced



$$\frac{W_R}{W} = \frac{(2^y-1)+1}{(2^x-1)+(2^y-1)+1+1} = \frac{2^y}{2^x+2^y}$$

(2^x-1): L; (2^y-1): R; $+1$: root; $+1$: convention

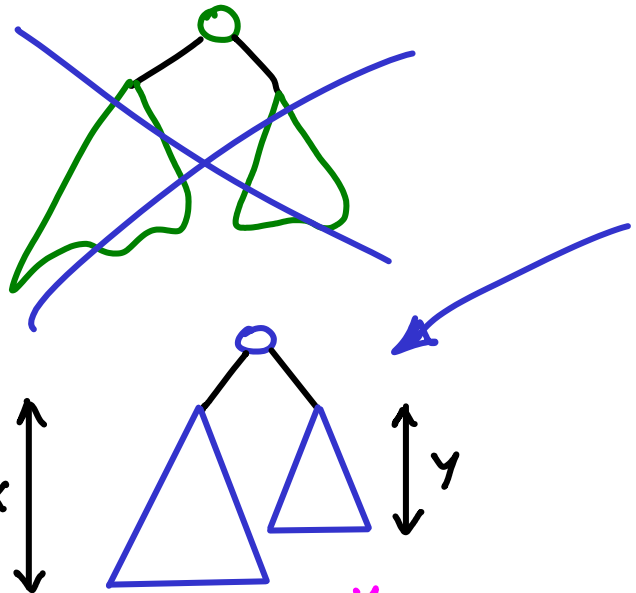
For all $\frac{1}{3} < \alpha < \frac{1}{2}$: if a tree is BB($\alpha$) then it is BB($\frac{1}{2}$)



True for height 1 or 2.

Suppose h = height of shortest tree where claim fails $(h > 2)$

$\exists$ tree that is $(>\frac{1}{3})$-balanced but not $\frac{1}{2}$-balanced

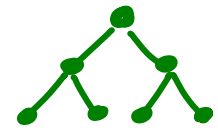


Subtrees are shorter, so they are $\frac{1}{2}$-balanced

$$\frac{W_R}{W} = \frac{(2^y - 1) + 1}{(2^x - 1) + (2^y - 1) + 1 + 1} = \frac{2^y}{2^x + 2^y} = \frac{1}{2^{x-y} + 1}$$

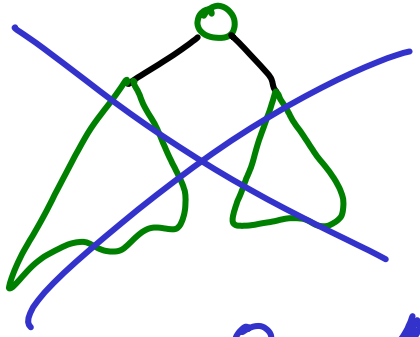
$(2^y - 1)$: L; $(2^y - 1)$: R; 1: root; last 1: convention

For all $\frac{1}{3} < \alpha < \frac{1}{2}$: if a tree is BB($\alpha$) then it is BB($\frac{1}{2}$)



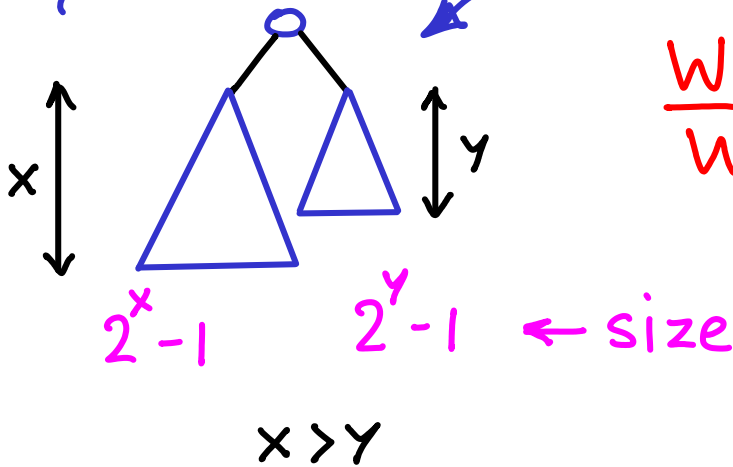
True for height 1 or 2.

Suppose h = height of shortest tree where claim fails $(h > 2)$



$\exists$ tree that is $(>\frac{1}{3})$-balanced but not $\frac{1}{2}$-balanced

Subtrees are shorter, so they are $\frac{1}{2}$-balanced



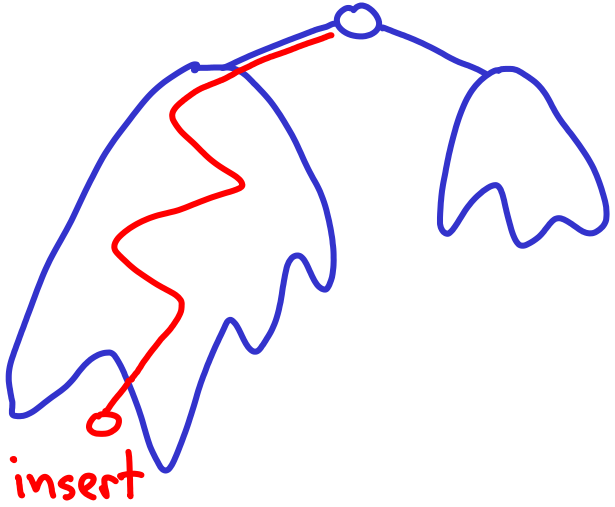
$$\frac{W_R}{W} = \frac{(2^y-1)+1}{(2^x-1)+(2^y-1)+1+1} = \frac{2^y}{2^x+2^y} = \frac{1}{2^{x-y}+1} < \frac{1}{3}$$

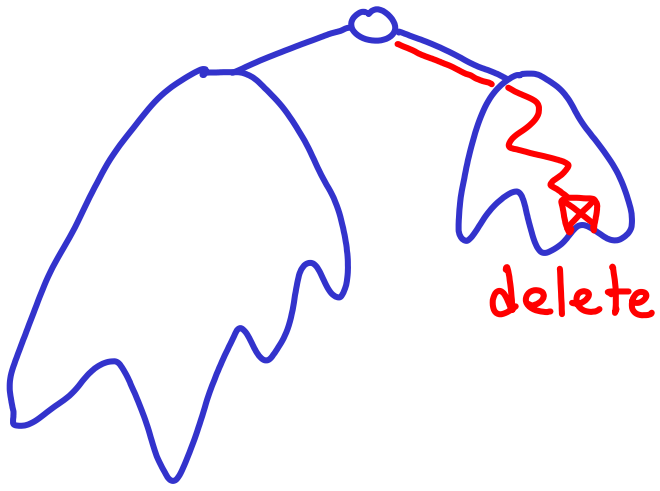
L R 1 root — convention

CONTRADICTION

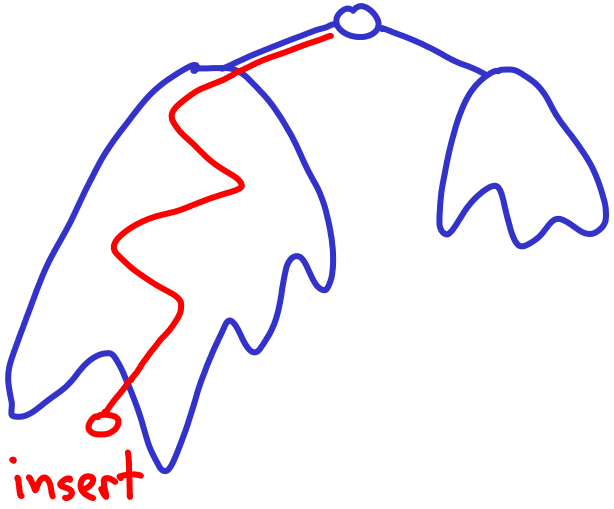
□

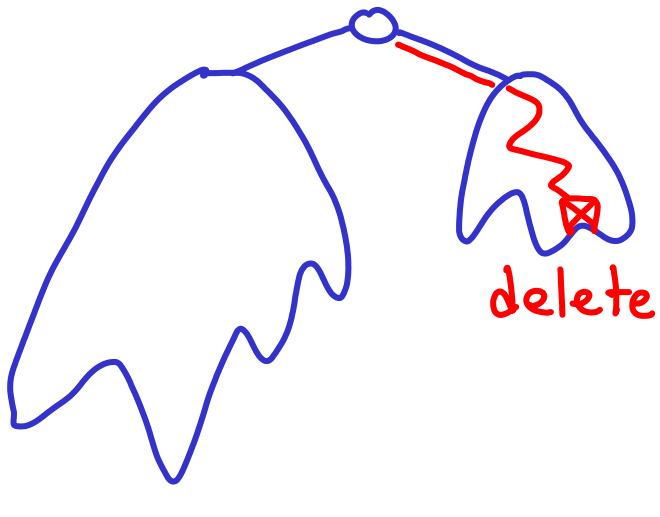
Insertion / Deletion in BB(α) weight-balanced tree





Insertion / Deletion in BB(α) weight-balanced tree

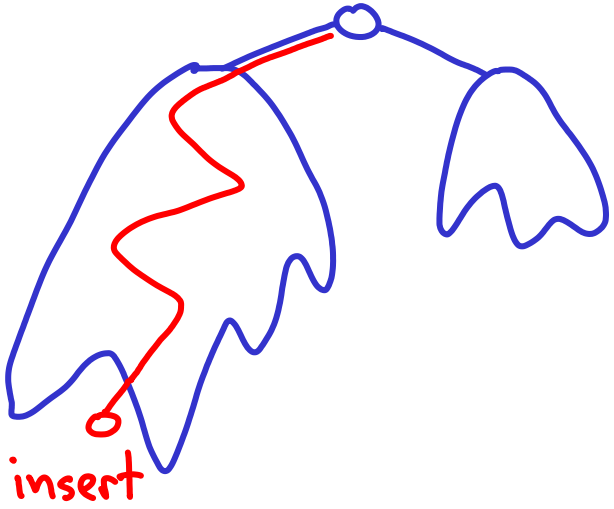


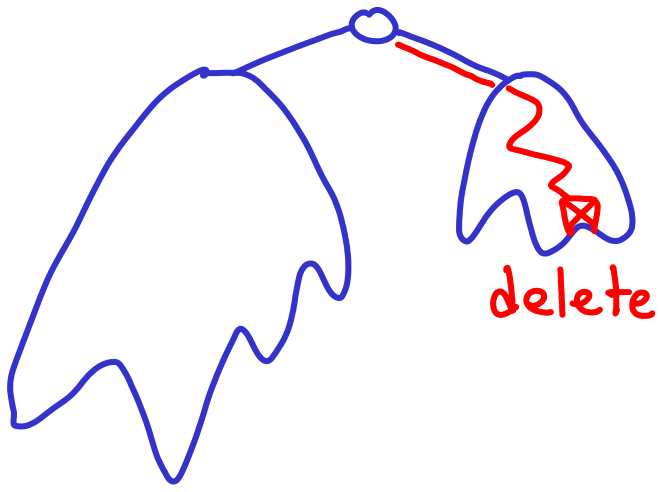


Any ancestor could now have:

$W_R < \alpha W$ (wlog)

Insertion / Deletion in BB(α) weight-balanced tree



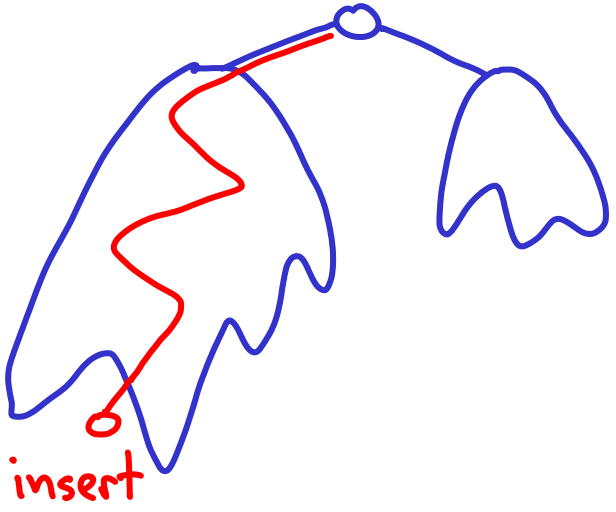


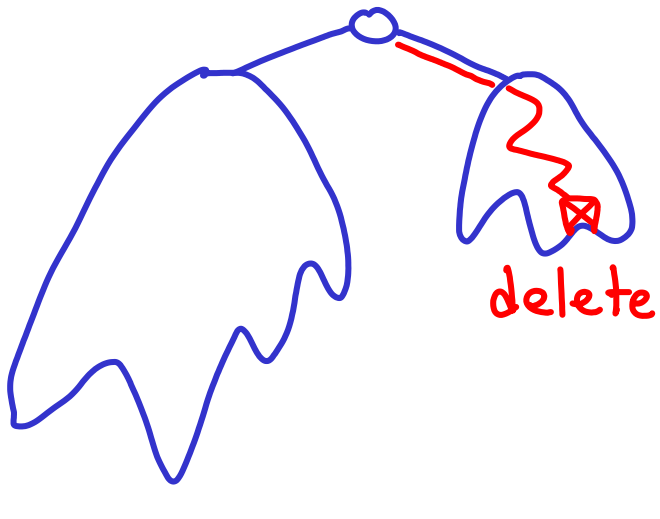
Any ancestor could now have:

$W_R < \alpha W$ (wlog)

Solution: rotate

Insertion / Deletion in BB(α) weight-balanced tree



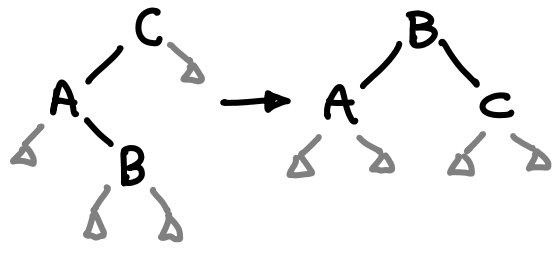


Any ancestor could now have:

$W_R < \alpha W$ (wlog)

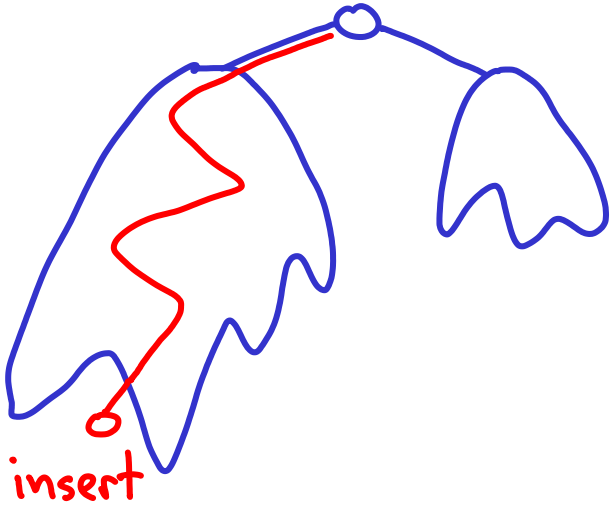
Solution: rotate

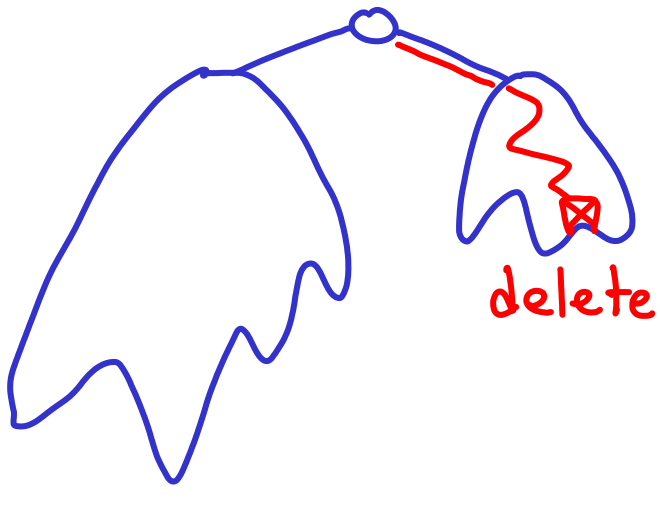
actually, 2 types of rotation
↳ standard & "split"



... depends on amount of imbalance

Insertion / Deletion in BB(α) weight-balanced tree



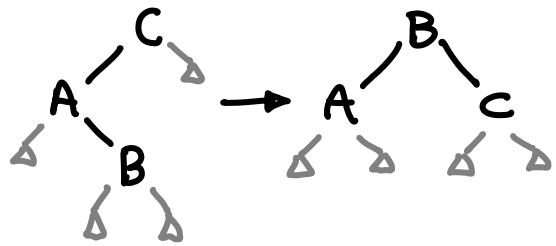


Any ancestor could now have:

$W_R < \alpha W$ (wlog)

Solution: rotate

actually, 2 types of rotation
↳ standard & "split"

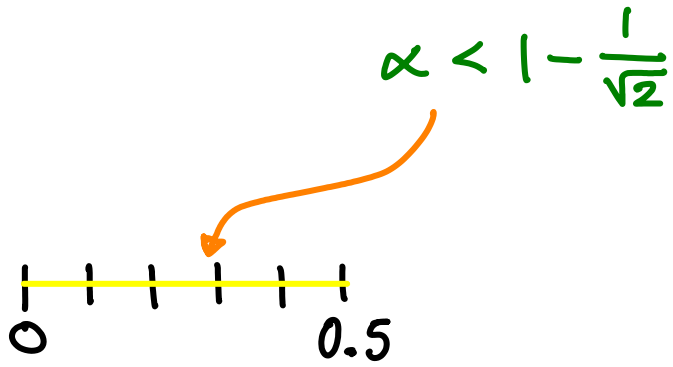


... depends on amount of imbalance

Works for $\alpha < 1 - \frac{1}{\sqrt{2}}$ ~ 0.3 // Proof involves some annoying counting

Can walk up & rebalance ancestors : $O(\log n)$

Insertion / Deletion in BB(α) weight-balanced tree

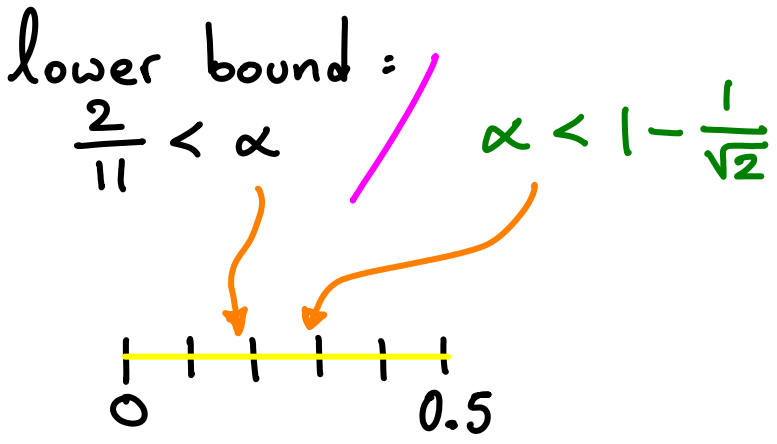


Insertion / Deletion in BB(α) weight-balanced tree

lower bound : $\frac{2}{11} < \alpha$

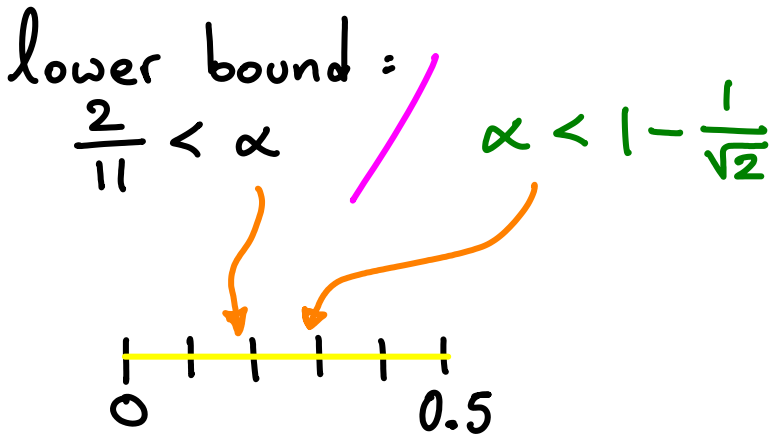
$\alpha < 1 - \frac{1}{\sqrt{2}}$

Insertion / Deletion in BB(α) weight-balanced tree



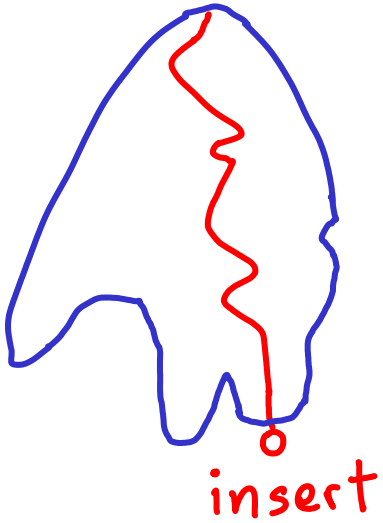
Claim:
can rebalance for α outside this range
with more complicated rotations

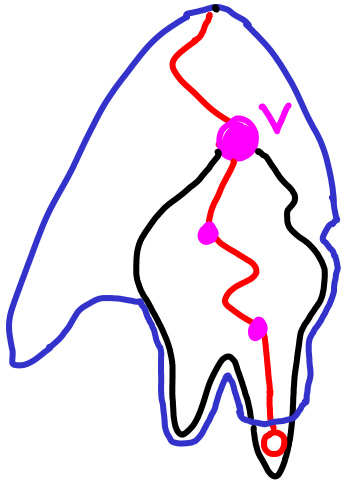
Insertion / Deletion in BB(α) weight-balanced tree



Claim:
can rebalance for α outside this range
with more complicated rotations

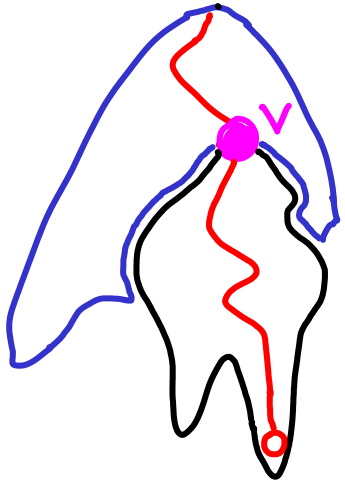
Instead, we will avoid rotations


insert



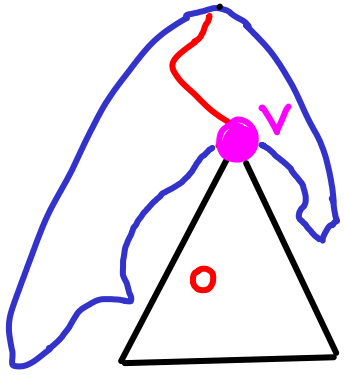
α-violation may occur for any ancestor

Let v be highest violation



α-violation may occur for any ancestor

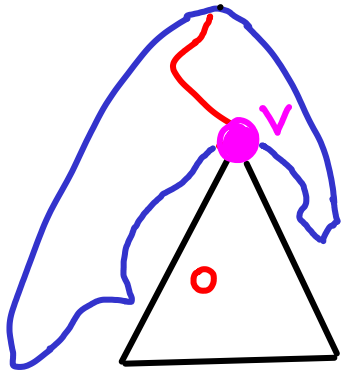
Let v be highest violation



α-violation may occur for any ancestor

Let v be highest violation

Rebuild subtree(v) cost ?

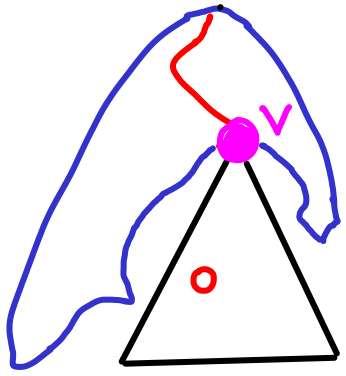


α-violation may occur for any ancestor

Let v be highest violation

Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

AMORTIZE ... ?



α-violation may occur for any ancestor

Let v be highest violation

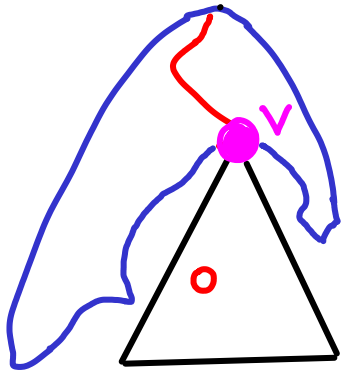
Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R|$$

→ technically, only if diff ≥ 2

(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$



α-violation may occur for any ancestor

Let v be highest violation

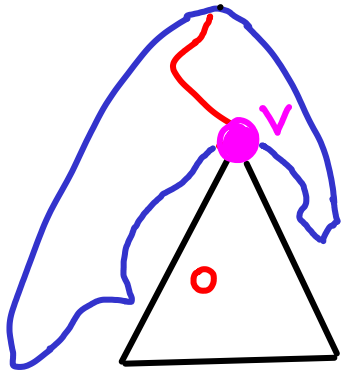
Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R|$$ → technically, only if diff ≥ 2

(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

Regular insert → $\Delta\Phi = 3 \cdot \sum_{\substack{\text{all} \\ \text{ancestors}}} \Delta |W_L - W_R|$



α-violation may occur for any ancestor

Let v be highest violation

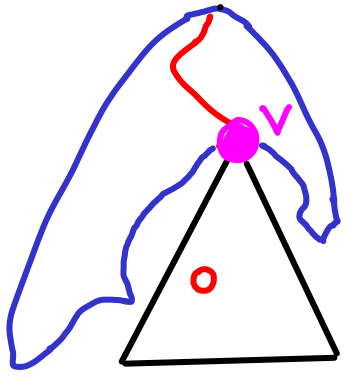
Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R|$$ → technically, only if diff ≥ 2

(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

Regular insert → $\Delta\Phi = 3 \cdot \sum_{\substack{\text{all} \\ \text{ancestors}}} \Delta |W_L - W_R| \leq 3 \log n$



α-violation may occur for any ancestor

Let v be highest violation

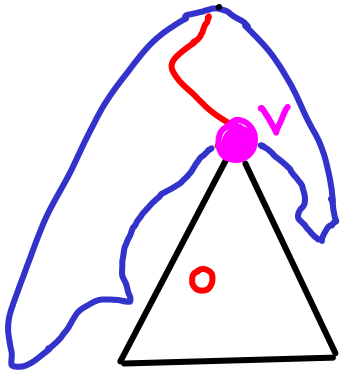
Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R|$$ → technically, only if diff ≥ 2

(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

Regular insert → $\Delta\Phi = 3 \cdot \sum_{\substack{\text{all} \\ \text{ancestors}}} \Delta |W_L - W_R| \leq 3 \log n$ $\quad \hat{c} \leq 4 \cdot \log n$



α-violation may occur for any ancestor

Let v be highest violation

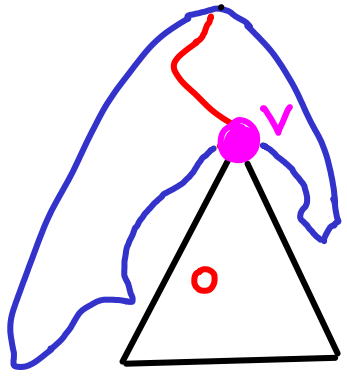
Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R|$$ → technically, only if diff ≥ 2

(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

Rebuild(v) → $\Delta\Phi = 3 \cdot \sum_{\substack{\text{all } x \\ \text{in tree(v)}}} \Delta |W_L - W_R|$



α-violation may occur for any ancestor

Let v be highest violation

Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

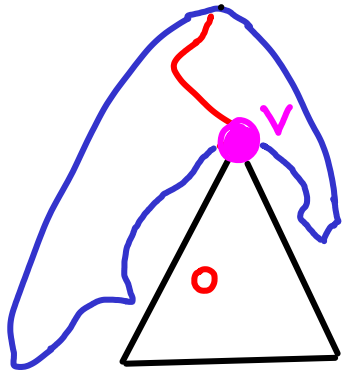
$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R|$$ → technically, only if diff ≥ 2

(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

Rebuild(v) → $\Delta\Phi = 3 \cdot \sum_{\substack{\text{all } x \\ \text{in tree(v)}}} \Delta |W_L - W_R|$

↳ every $\Delta\Phi$ term ≤ 0



α-violation may occur for any ancestor

Let v be highest violation

Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

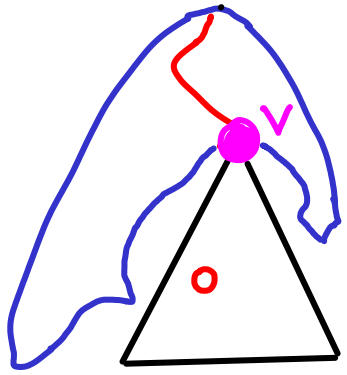
$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R| \rightarrow$$ technically, only if diff ≥ 2

(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

Rebuild(v) $\rightarrow \Delta\Phi = 3 \cdot \sum_{\substack{\text{all } x \\ \text{in tree(v)}}} \Delta|W_L - W_R| \leq 3 \cdot \Delta|W_L^v - W_R^v|$

↳ every $\Delta\Phi$ term ≤ 0



α-violation may occur for any ancestor

Let v be highest violation

Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R| \rightarrow$$ technically, only if diff ≥ 2

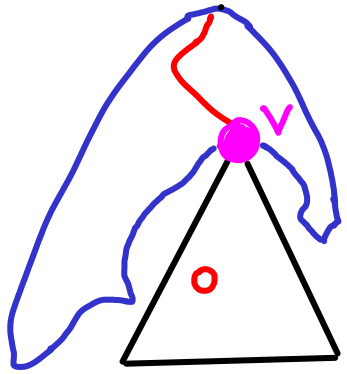
(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

Rebuild(v) $\rightarrow \Delta\Phi = 3 \cdot \sum_{\substack{\text{all } x \\ \text{in tree(v)}}} \Delta|W_L - W_R| \leq 3 \cdot \Delta|W_L^v - W_R^v|$

↳ every $\Delta\Phi$ term ≤ 0

violation: wlog $W_R < \frac{1}{3}W$



α-violation may occur for any ancestor

Let v be highest violation

Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R| \rightarrow$$ technically, only if diff ≥ 2

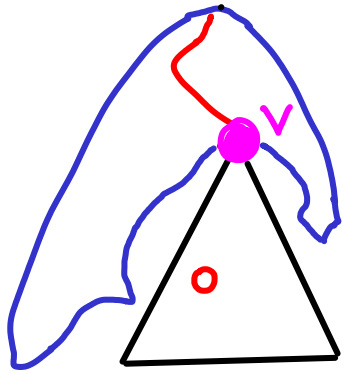
(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

Rebuild(v) $\rightarrow \Delta\Phi = 3 \cdot \sum_{\substack{\text{all } x \\ \text{in tree(v)}}} \Delta|W_L - W_R| \leq 3 \cdot \Delta|\underbrace{W_L^v - W_R^v}_{?}|$

↳ every $\Delta\Phi$ term ≤ 0

violation: wlog $W_R < \frac{1}{3}W$ $\frac{2}{3}W < W_L$



α-violation may occur for any ancestor

Let v be highest violation

Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R| \rightarrow$$ technically, only if diff ≥ 2

(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

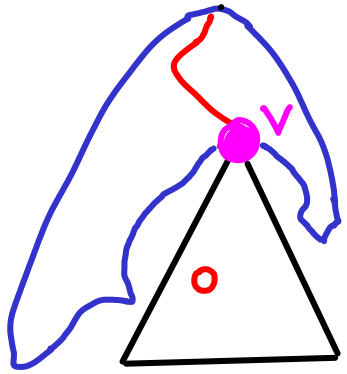
Rebuild(v) $\rightarrow \Delta\Phi = 3 \cdot \sum_{\substack{\text{all } x \\ \text{in tree(v)}}} \Delta|W_L - W_R| \leq 3 \cdot \Delta|W_L^v - W_R^v|$

↳ every $\Delta\Phi$ term ≤ 0

$\frac{1}{3}W$

Φ_{i-1}

violation: wlog $W_R < \frac{1}{3}W < \frac{2}{3}W < W_L$



α-violation may occur for any ancestor

Let v be highest violation

Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R| \rightarrow$$ technically, only if diff ≥ 2

(diff = 1 unavoidable)

e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

Rebuild(v) $\rightarrow \Delta\Phi = 3 \cdot \sum_{\substack{\text{all } x \\ \text{in tree(v)}}} \Delta|W_L - W_R| \leq 3 \cdot \Delta|W_L^v - W_R^v|$

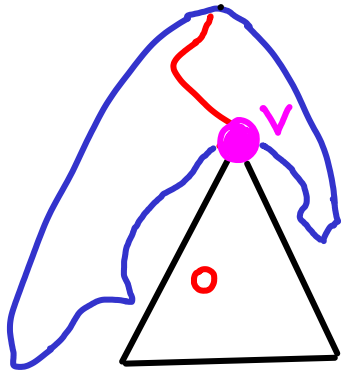
↳ every $\Delta\Phi$ term ≤ 0

$\leq 3 \cdot [0 - \frac{1}{3}W] \sim -\text{size}(v)$

$\Phi_i - \Phi_{i-1}$

violation: wlog $W_R < \frac{1}{3}W < \frac{2}{3}W < W_L$

□



α-violation may occur for any ancestor

Let v be highest violation

Rebuild subtree(v) $O(\log n) + \Theta(\text{size}(v)) = O(n)$

$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R| \rightarrow$$ technically, only if diff ≥ 2

(diff = 1 unavoidable)

Rebuild(v) $\rightarrow \Delta\Phi = \frac{1}{1-2\alpha} \cdot \sum_{\substack{\text{all } x \\ \text{in tree(v)}}} \Delta|W_L - W_R| \leq \frac{1}{1-2\alpha} \cdot \Delta|W_L^v - W_R^v|$

↳ every $\Delta\Phi$ term ≤ 0

$\leq \frac{1}{1-2\alpha} \cdot [0 - (1-2\alpha)W] \sim -\text{size}(v)$

($0 \rightarrow \Phi_i$, $(1-2\alpha)W \rightarrow \Phi_{i-1}$)

violation (wlog): $W_R < \alpha W < (1-\alpha) W < W_L$

□

An application of BB(α)

Dynamic high-dimensional multilayered range trees

↳ not easy to update efficiently with rotations

see Michiel Smid's notes, re: Blum & Mehlhorn

SCAPEGOAT TREES : amortized balanced dynamic BST

SCAPEGOAT TREES : amortized balanced dynamic BST

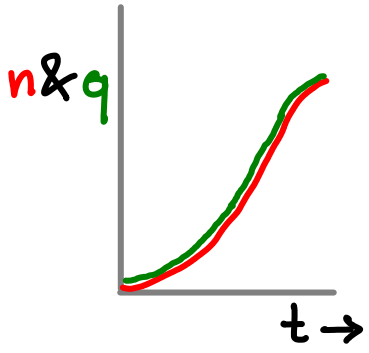
n = # keys

q = variable s.t. $q/2 \leq n \leq q$

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \leq n \leq q$

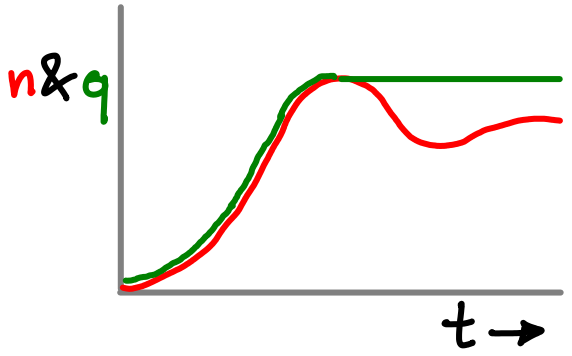


- if $n=q$ and n is incremented, then increment q

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \leq n \leq q$

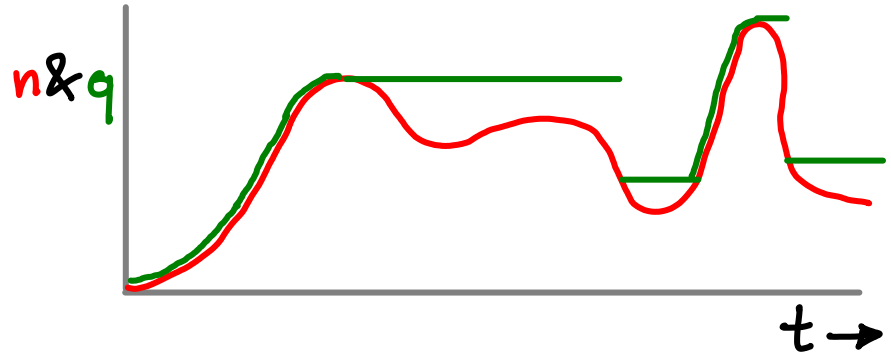


- if $n=q$ and n is incremented, then increment q

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$

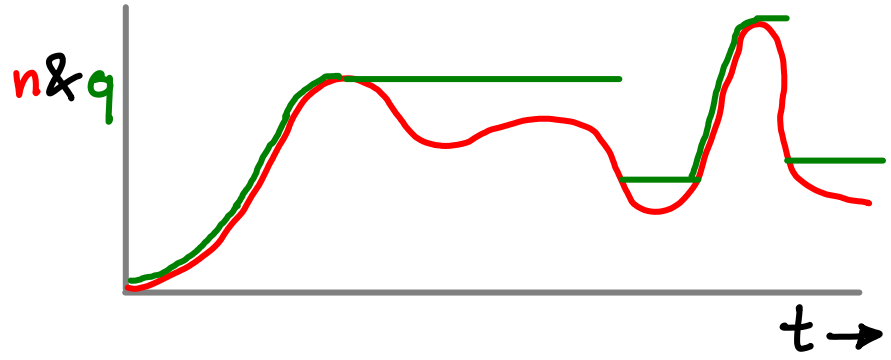


- if $n = q$ and n is incremented, then increment q
- if $n = q/2$ and n is decremented, then set $q = n$

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$



α = balance factor,

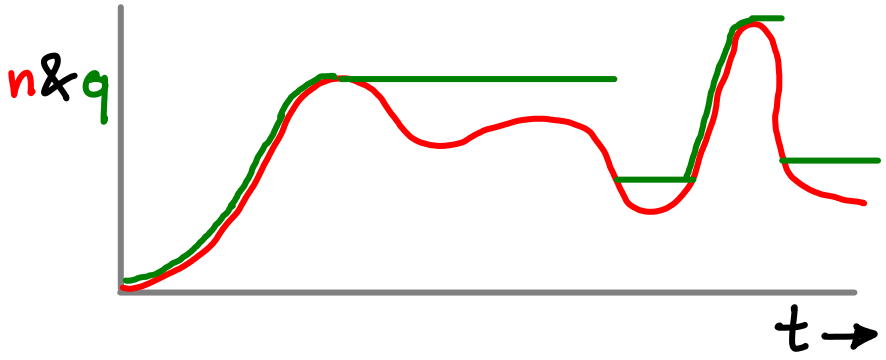
$\frac{1}{2} < \alpha < 1$

- if $n = q$ and n is incremented, then increment q
- if $n = q/2$ and n is decremented, then set $q = n$

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$



α = balance factor,

$\frac{1}{2} < \alpha < 1$

determines max allowed height.

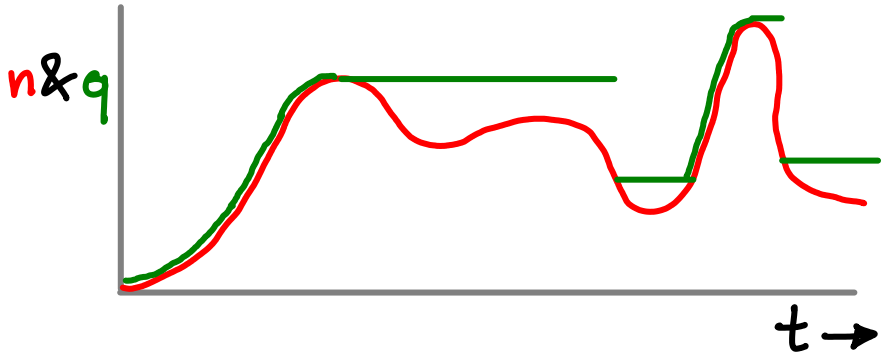
$h \le \log_{1/\alpha} q$

- if $n = q$ and n is incremented, then increment q
- if $n = q/2$ and n is decremented, then set $q = n$

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$



α = balance factor,

$\frac{1}{2} < \alpha < 1$

determines max allowed height.

$h \le \log_{1/\alpha} q$

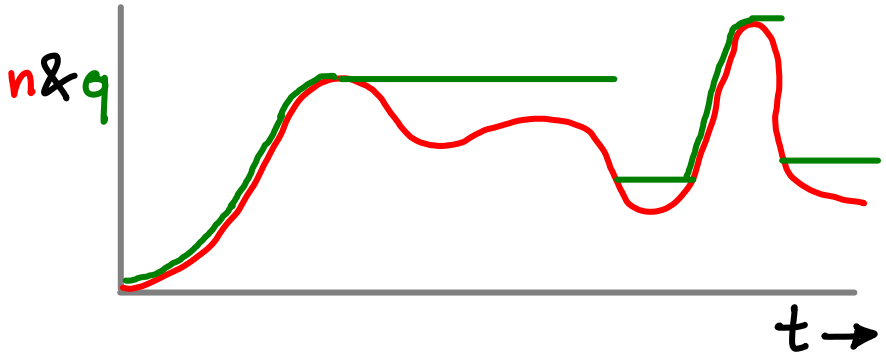
$\le \log_{1/\alpha} 2n$

- if $n = q$ and n is incremented, then increment q
- if $n = q/2$ and n is decremented, then set $q = n$

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$



α = balance factor,

$\frac{1}{2} < \alpha < 1$

determines max allowed height.

$h \le \log_{1/\alpha} q$

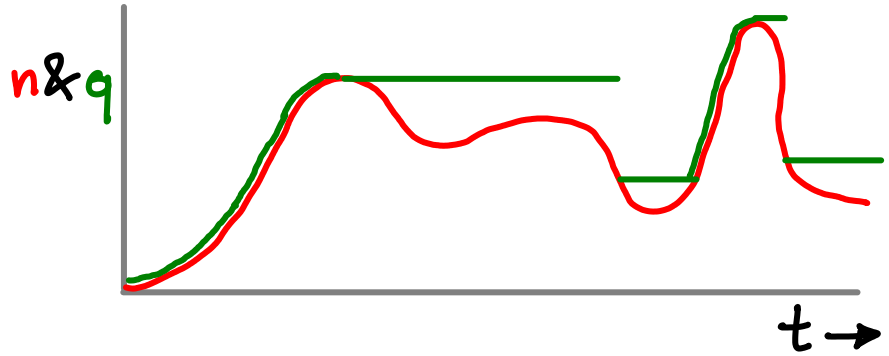
$\le \log_{1/\alpha} 2n = \log_{1/\alpha} n + O(1)$

- if $n = q$ and n is incremented, then increment q
- if $n = q/2$ and n is decremented, then set $q = n$

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$ →



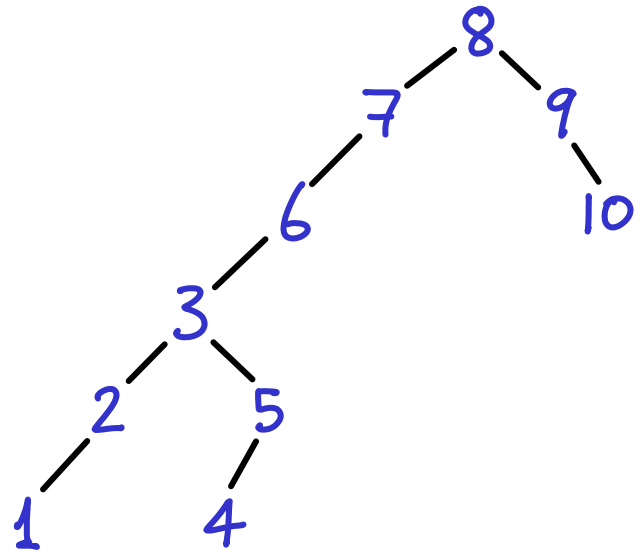
α = balance factor,

$\frac{1}{2} < \alpha < 1$

determines max allowed height.

$h \le \log_{1/\alpha} q$

$\le \log_{1/\alpha} 2n = \log_{1/\alpha} n + O(1)$



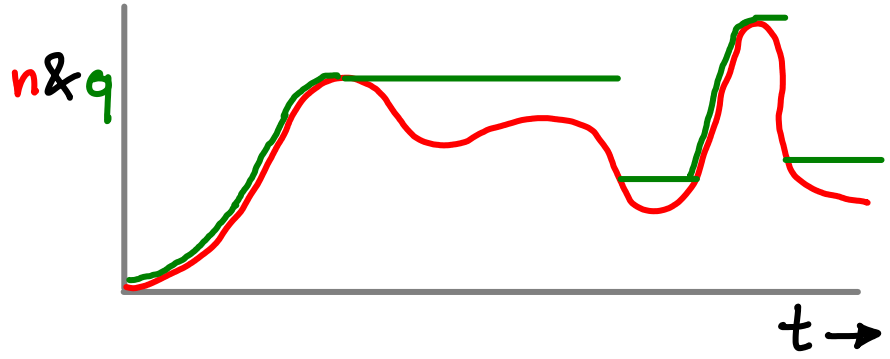
$n = 10$

$q = ?$

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$ →



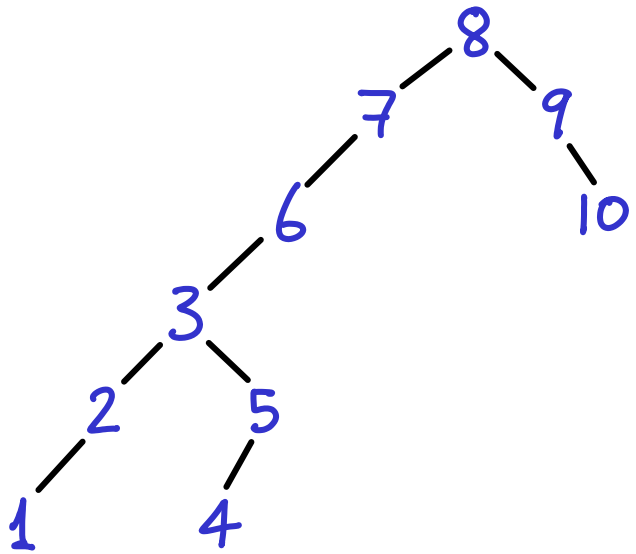
α = balance factor,

$\frac{1}{2} < \alpha < 1$

determines max allowed height.

$h \le \log_{1/\alpha} q$

$\le \log_{1/\alpha} 2n = \log_{1/\alpha} n + O(1)$



$n = 10$

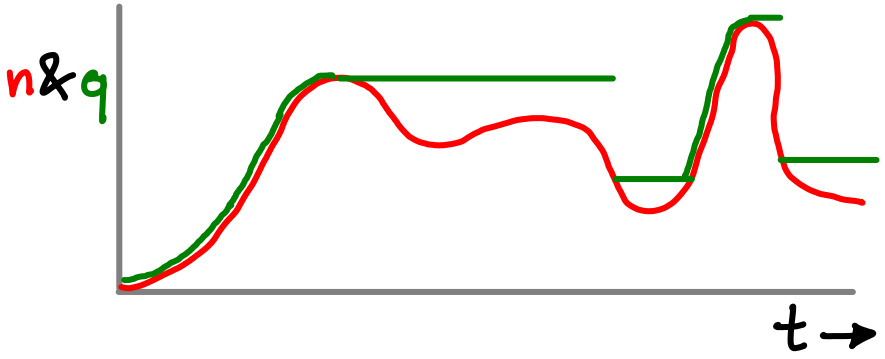
$10 \le q \le 20$

$\alpha = \frac{2}{3}$ (our choice)

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$ →



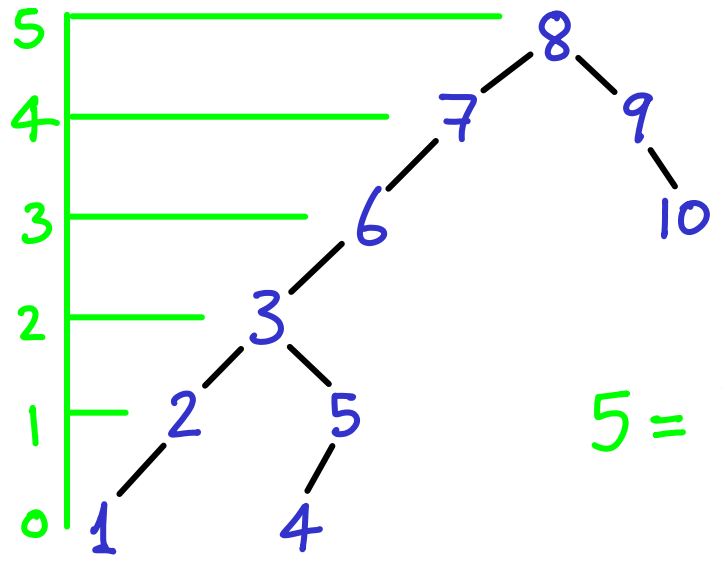
α = balance factor,

$\frac{1}{2} < \alpha < 1$

determines max allowed height.

$h \le \log_{1/\alpha} q$

$\le \log_{1/\alpha} 2n = \log_{1/\alpha} n + O(1)$



$n = 10$

$10 \le q \le 20$

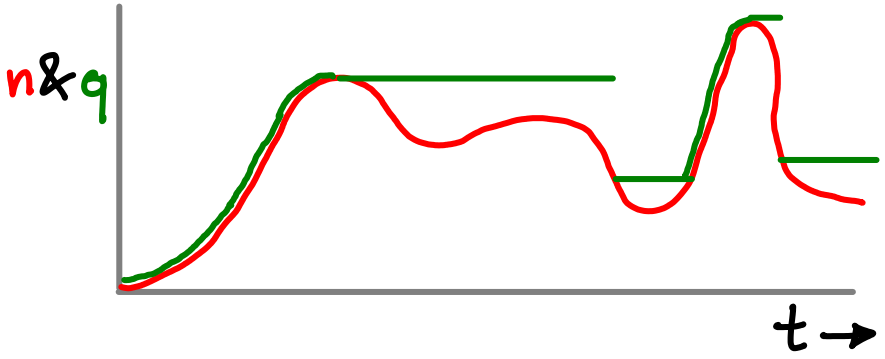
$\alpha = \frac{2}{3}$

$5 = h$

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$ →



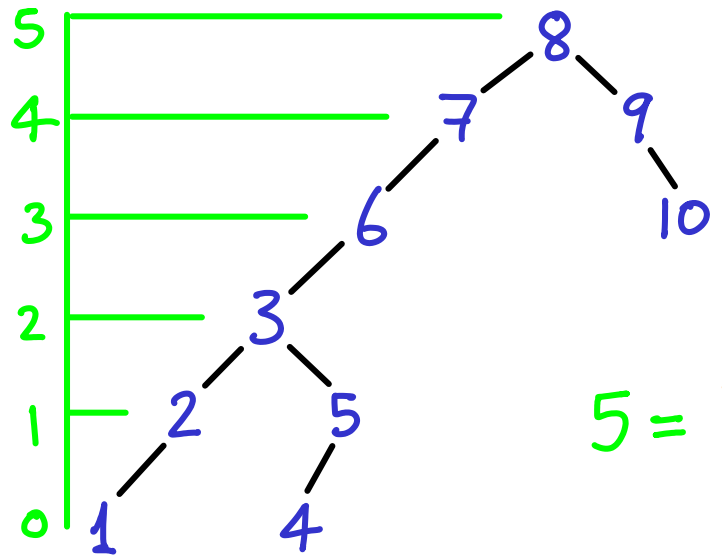
α = balance factor,

$\frac{1}{2} < \alpha < 1$

determines max allowed height.

$h \le \log_{1/\alpha} q$

$\le \log_{1/\alpha} 2n = \log_{1/\alpha} n + O(1)$



$n = 10$

$10 \le q \le 20$

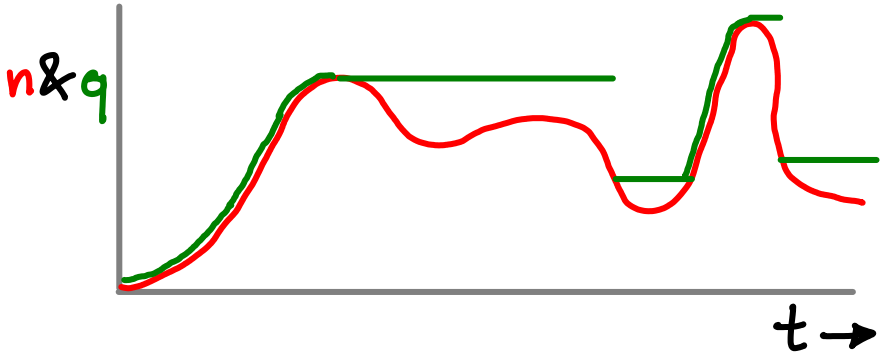
$\alpha = \frac{2}{3}$

$5 = h \le \log_{1.5} q$

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$ →



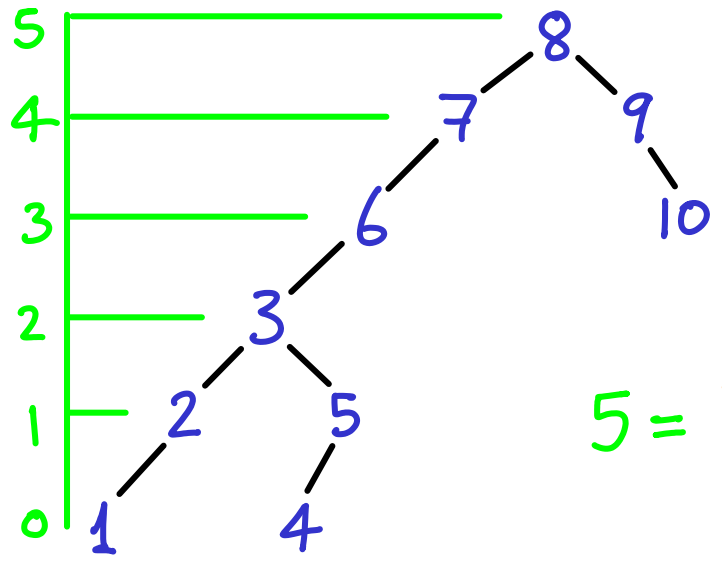
α = balance factor,

$\frac{1}{2} < \alpha < 1$

determines max allowed height.

$h \le \log_{1/\alpha} q$

$\le \log_{1/\alpha} 2n = \log_{1/\alpha} n + O(1)$



$n = 10$

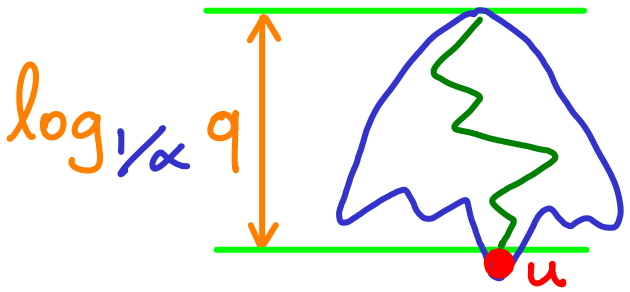
$10 \le q \le 20$

$\alpha = \frac{2}{3}$

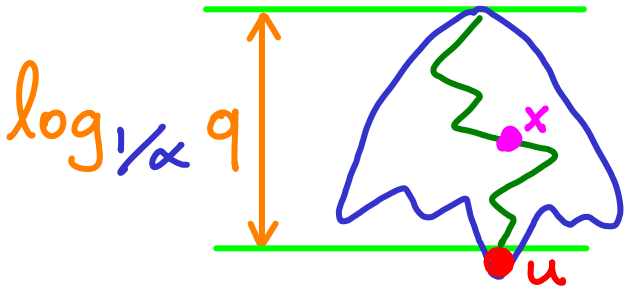
$5 = h \le \log_{1.5} q \sim 5.68$ for $q=10$

OK for $q > 10$

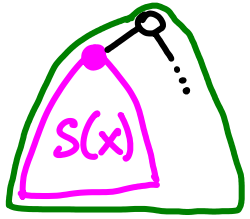
Suppose height condition fails → $h > \log_{1/\alpha} q$ → Node u is deeper



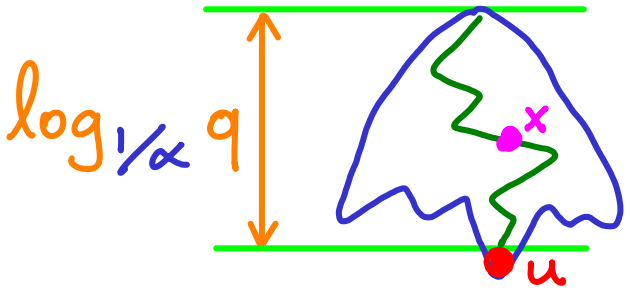
Suppose height condition fails → $h > \log_{1/\alpha} q$ → Node u is deeper



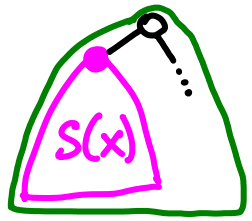
Then an ancestor x of u with parent p must have subtree size$(x) > \alpha \cdot$ (size(p))



Suppose height condition fails → $h > \log_{1/\alpha} q$ → Node u is deeper



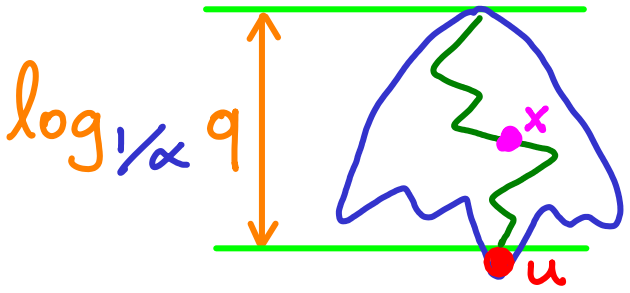
Then an ancestor x of u with parent p must have subtree size$(x) > \alpha \cdot$ (size(p))



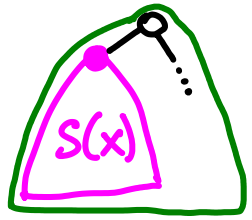
By contradiction: if all ancestors of x had $< \alpha$ ratio,

we would run out of nodes before reaching depth $\log_{1/\alpha} q$

Suppose height condition fails → $h > \log_{1/\alpha} q$ → Node u is deeper



Then an ancestor x of u with parent p must have subtree size(x) > $\alpha \cdot$(size(p))



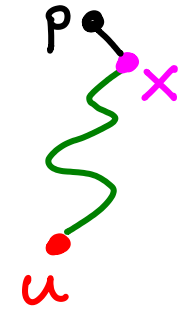
By contradiction: if all ancestors of x had $< \alpha$ ratio,

we would run out of nodes before reaching depth $\log_{1/\alpha} q$

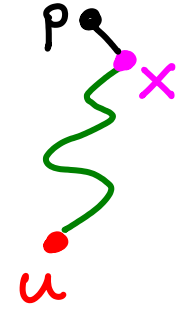
$$H_n \leq H\left(\frac{n}{1/\alpha}\right) + 1$$

geometric series

If u has depth $> \log_{1/\alpha} q$ then $\text{size}(x) > \alpha \cdot (\text{size}(p))$

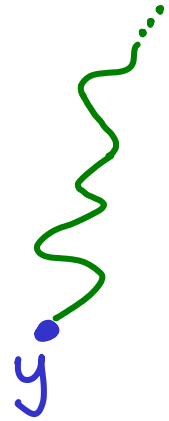


If u has depth $> \log_{1/\alpha} q$ then $\text{size}(x) > \alpha \cdot (\text{size}(p))$

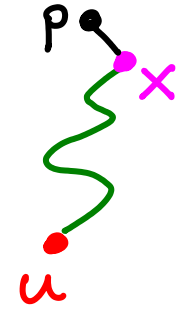


Insertion of node y:

- compare $\text{depth}(y)$ with $\log_{1/\alpha} q$

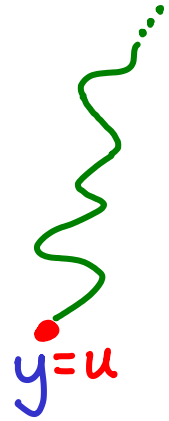


If u has depth $> \log_{1/\alpha} q$ then $\text{size}(x) > \alpha \cdot (\text{size}(p))$

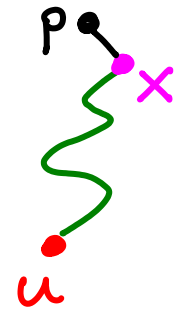


Insertion of node y:

- compare $\text{depth}(y)$ with $\log_{1/\alpha} q$
- if violation,

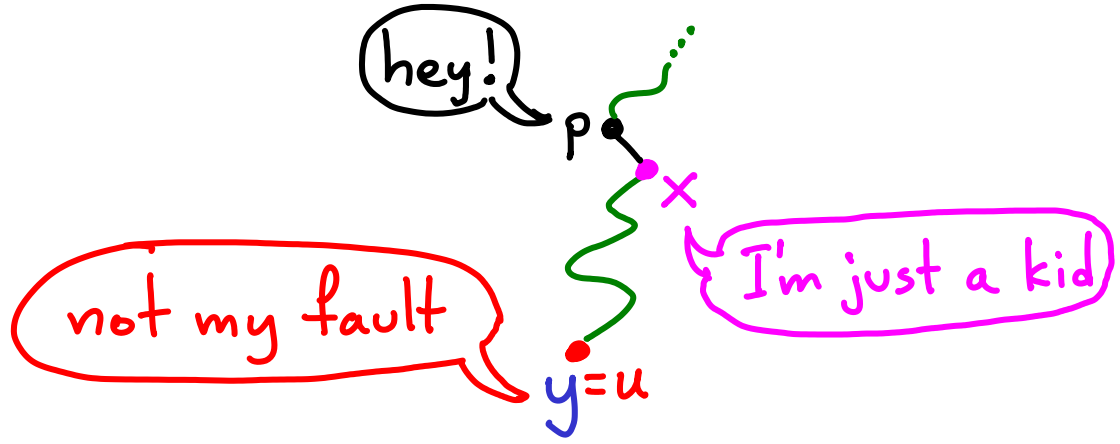


If u has depth $> \log_{1/\alpha} q$ then $\text{size}(x) > \alpha \cdot (\text{size}(p))$

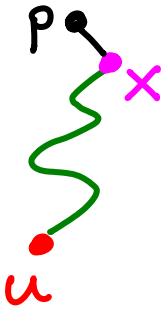


Insertion of node y:

- compare $\text{depth}(y)$ with $\log_{1/\alpha} q$
- if violation, $\exists p$ = scapegoat

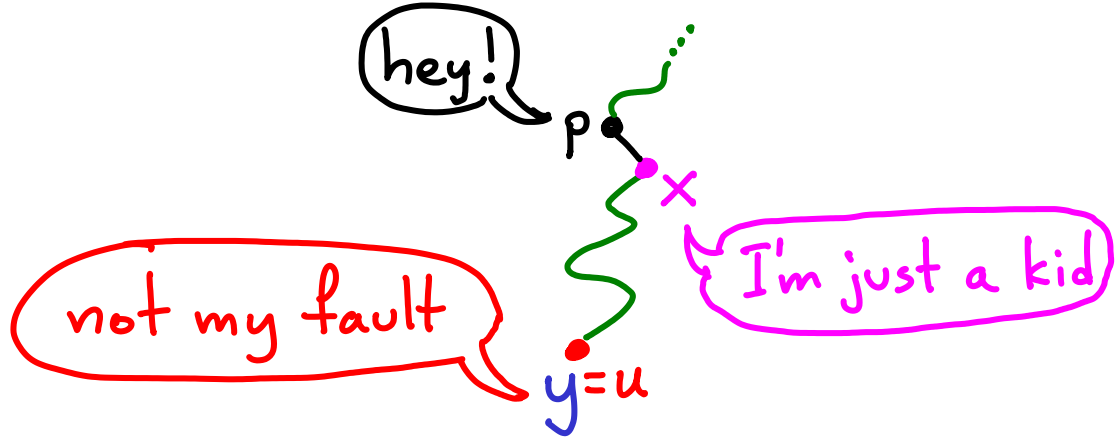


If u has depth $> \log_{1/\alpha} q$ then $\text{size}(x) > \alpha \cdot (\text{size}(p))$

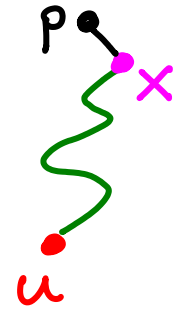


Insertion of node y:

- compare depth(y) with $\log_{1/\alpha} q$
- if violation, $\exists p$ = scapegoat
- find p
- rebuild subtree(p)

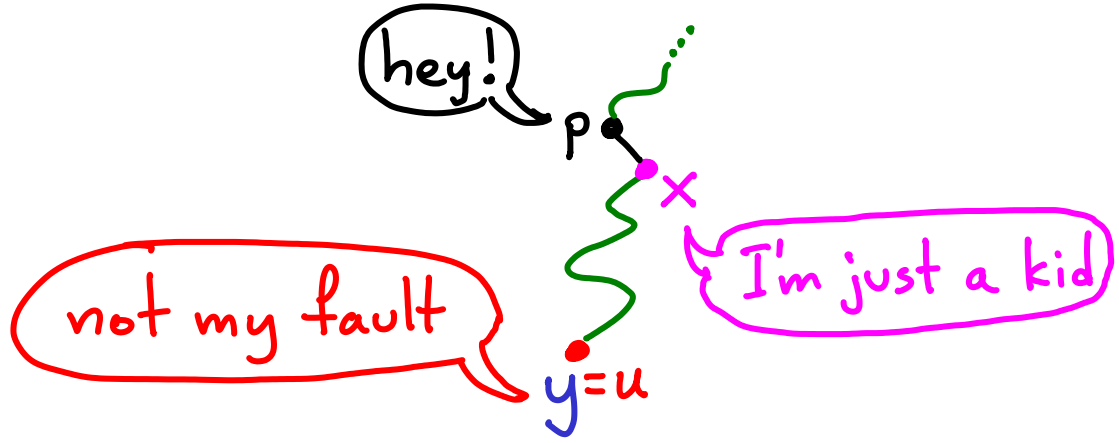


If u has depth $> \log_{1/\alpha} q$ then $\text{size}(x) > \alpha \cdot (\text{size}(p))$

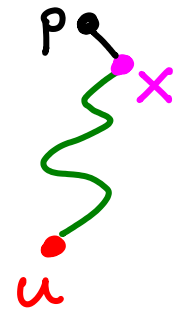


Insertion of node y:

- compare $\text{depth}(y)$ with $\log_{1/\alpha} q$
- if violation, $\exists p$ = scapegoat
- find p
- rebuild $\text{subtree}(p)$ $= \Theta(\text{size}(p))$

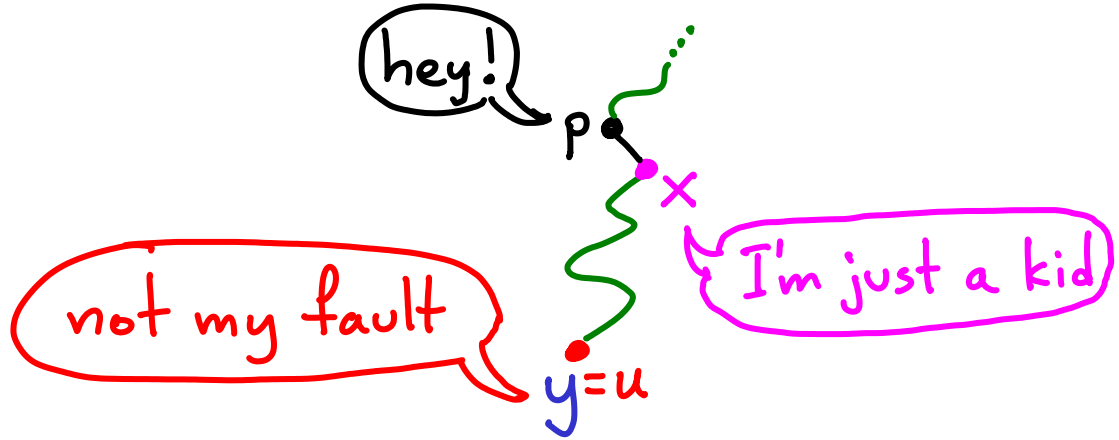


If u has depth $> \log_{1/\alpha} q$ then $\text{size}(x) > \alpha \cdot (\text{size}(p))$



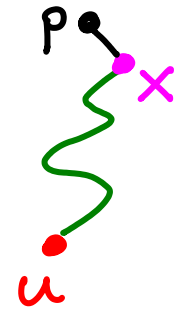
Insertion of node y:

- compare $\text{depth}(y)$ with $\log_{1/\alpha} q$
- if violation, $\exists p$ = scapegoat
- find p ... how?
- rebuild $\text{subtree}(p)$ $= \Theta(\text{size}(p))$



↳ why does that fix things?

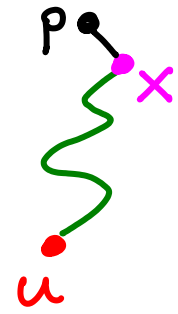
If u has depth $> \log_{1/\alpha} q$ then $\text{size}(x) > \alpha \cdot (\text{size}(p))$



Insertion of node y:

- compare depth(y) with $\log_{1/\alpha} q$
- if violation, $\exists p$ = scapegoat
- find p ... how? → store size( ) at each node → walk up & compare
- rebuild subtree(p) $= \Theta(\text{size}(p))$
  - → why does that fix things?

If u has depth $> \log_{1/\alpha} q$ then $\text{size}(x) > \alpha \cdot (\text{size}(p))$



Insertion of node y:

- compare depth(y) with $\log_{1/\alpha} q$
- if violation, $\exists p$ = scapegoat
- find p ... how? OR
  - store size( ) at each node → walk up & compare
  - brute force, $\Theta(\text{size}(p))$ → no need to store size( )
- rebuild subtree(p) $= \Theta(\text{size}(p))$
  - → why does that fix things?

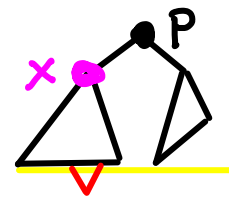
If u has depth $> \log_{1/\alpha} q$ then $\text{size}(x) > \alpha \cdot (\text{size}(p))$

Insertion of node y:

- compare depth(y) with $\log_{1/\alpha} q$
- if violation, $\exists p$ = scapegoat
- find p ... how? OR
  - store size( ) at each node → walk up & compare
  - brute force, $\Theta(\text{size}(p))$ → no need to store size( )
- rebuild subtree(p) $= \Theta(\text{size}(p))$

  → why does that fix things?

  Only y caused a violation.
  p is so unbalanced,
  plenty of space to fill



Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat}} + O(\log_{1/\alpha} q)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat}} + O(\log_{1/\alpha} q)$

Deletion

- doesn't cause a height problem

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat}} + O(\log_{1/\alpha} q)$

Deletion

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat}} + O(\log_{1/\alpha} q)$

Deletion

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat}} + O(\log_{1/\alpha} q)$

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat}} + O(\log_{1/\alpha} q)$

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

AMORTIZE ...

What changes a lot if cost is expensive?

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat}} + O(\log_{1/\alpha} q)$

$|\text{size}(L) - \text{size}(R)|$

← for all nodes in tree(scapegoat)

but also depends on α

AMORTIZE ...

What changes a lot if cost is expensive?

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat}} + O(\log_{1/\alpha} q)$

$|\text{size}(L) - \text{size}(R)|$

for all nodes in tree(scapegoat)

but also depends on α

AMORTIZE ...

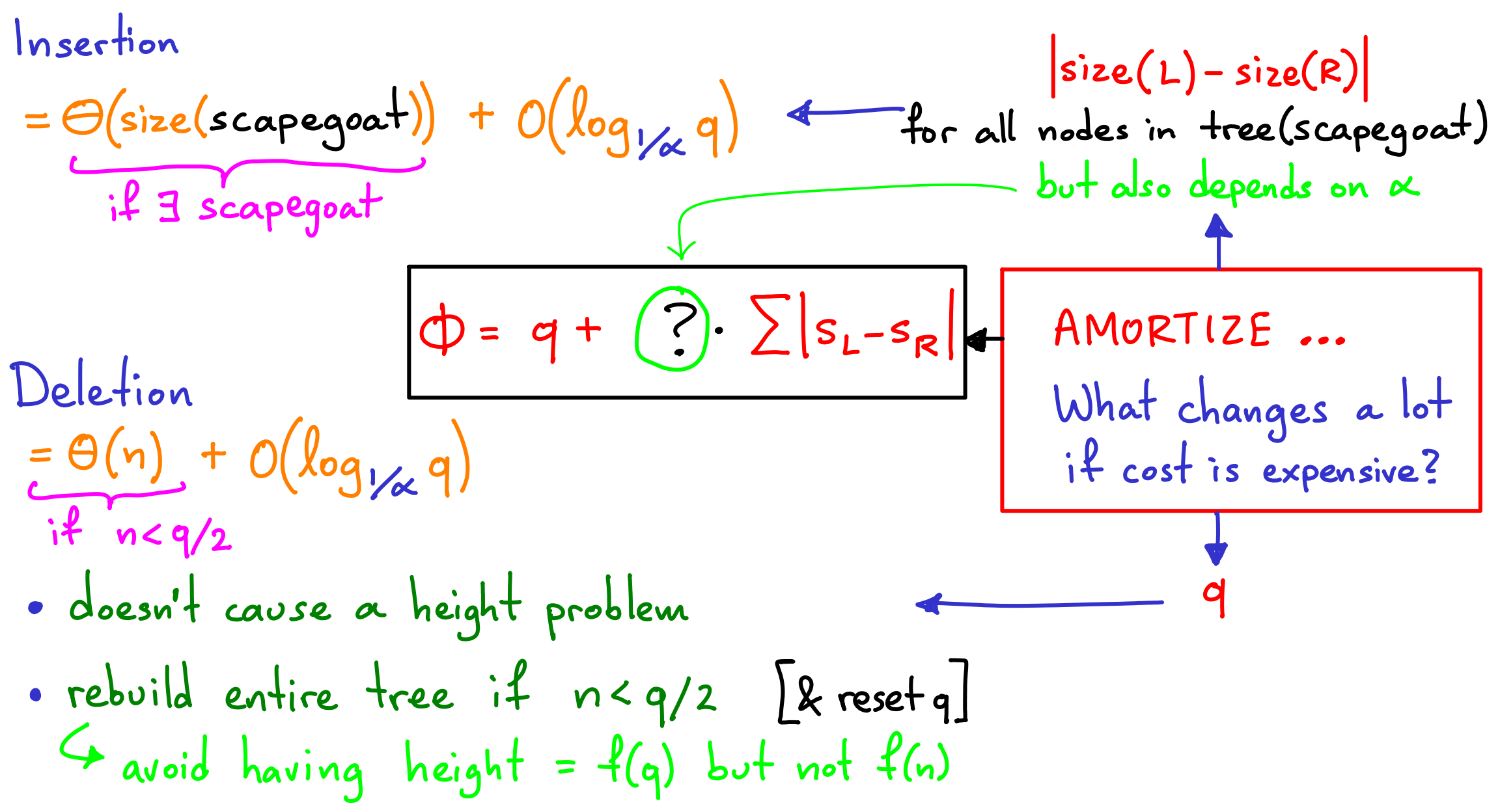
What changes a lot if cost is expensive?

q

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  - ↳ avoid having height = $f(q)$ but not $f(n)$


Insertion
$= \Theta(\text{size}(\text{scapegoat})) + O(\log_{1/\alpha} q)$
if ∃ scapegoat
$|\text{size}(L) - \text{size}(R)|$
for all nodes in tree(scapegoat)
but also depends on α
$\Phi = q + ? \cdot \sum |s_L - s_R|$
AMORTIZE ...
What changes a lot if cost is expensive?
q
Deletion
$= \Theta(n) + O(\log_{1/\alpha} q)$
if $n < q/2$
• doesn't cause a height problem
• rebuild entire tree if $n < q/2$ [& reset q]
↳ avoid having height = f(q) but not f(n)

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat } x} + O(\log_{1/\alpha} q)$

$$\Phi = q + \boxed{?} \cdot \sum |s_L - s_R|$$

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

Ignore $\Delta\Phi$ for all nodes except x
↳ all ≤ 0

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat } x} + O(\log_{1/\alpha} q)$

$$\Phi = q + \boxed{?} \cdot \sum |s_L - s_R|$$

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat } x} + O(\log_{1/\alpha} q)$

Ignore $\Delta\Phi$ for all nodes except x
↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = 0$ or 1 $+q$

$$\Phi = q + (?) \cdot \sum |s_L - s_R|$$

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat } x} + O(\log_{1/\alpha} q)$

Ignore $\Delta\Phi$ for all nodes except x
↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = 0$ or 1 $+q$

Φ_{i-1} ...

$$\Phi = q + \boxed{?} \cdot \sum |s_L - s_R|$$

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat } x} + O(\log_{1/\alpha} q)$

Ignore $\Delta\Phi$ for all nodes except x
↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = 0$ or 1 $+q$

Φ_{i-1}

$$\Phi = q + \boxed{?} \cdot \sum |s_L - s_R|$$

$s_L > \alpha s_x$

WLOG

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat } x} + O(\log_{1/\alpha} q)$

Ignore $\Delta\Phi$ for all nodes except x
↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = 0$ or 1 $+q$

Φ_{i-1}

$$\Phi = q + \boxed{?} \cdot \sum |s_L - s_R|$$

$s_L > \alpha s_x$

$s_R < (1-\alpha) s_x$

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat } x} + O(\log_{1/\alpha} q)$

Ignore $\Delta\Phi$ for all nodes except x
↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = 0$ or 1 $+q$

Φ_{i-1}

$$\Phi = q + \boxed{?} \cdot \sum |s_L - s_R|$$

$s_L > \alpha s_x$

$s_R < (1-\alpha) s_x$

$s_L - s_R > [\alpha - (1-\alpha)] s_x$

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \Theta(\text{size}(\text{scapegoat})) + O(\log_{1/\alpha} q)$

$\Theta(\text{size}(\text{scapegoat}))$: if $\exists$ scapegoat x

Ignore $\Delta\Phi$ for all nodes except x
↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = 0$ or 1 $+q$

Φ_{i-1}

$$\Phi = q + \boxed{?} \cdot \sum |s_L - s_R|$$

$s_L > \alpha s_x$

$s_R < (1-\alpha) s_x$

$s_L - s_R > [\alpha - (1-\alpha)] s_x$

$= (2\alpha - 1) s_x$

Deletion

$= \Theta(n) + O(\log_{1/\alpha} q)$

$\Theta(n)$: if $n < q/2$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat } x} + O(\log_{1/\alpha} q)$

Ignore $\Delta\Phi$ for all nodes except x
↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = 0$ or 1 $+q$

Φ_{i-1}

$$\Phi = q + \frac{1}{(2\alpha-1)} \sum |s_L - s_R|$$

$s_L > \alpha s_x$

$s_R < (1-\alpha) s_x$

$s_L - s_R > [\alpha - (1-\alpha)] s_x$

$= (2\alpha - 1) s_x$

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion

$= \underbrace{\Theta(\text{size}(\text{scapegoat}))}_{\text{if } \exists \text{ scapegoat } x} + O(\log_{1/\alpha} q)$

$$\Phi = q + \frac{1}{(2\alpha-1)} \sum |s_L - s_R|$$

* if diff > 1

Deletion

$= \underbrace{\Theta(n)}_{\text{if } n < q/2} + O(\log_{1/\alpha} q)$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Ignore $\Delta\Phi$ for all nodes except x
↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = 0$ or 1 (+q) ← $O(1)$, or O^*

$\Phi_{i-1} > s_x$ (+q)

$s_L > \alpha s_x$

$s_R < (1-\alpha) s_x$

$s_L - s_R > [\alpha - (1-\alpha)] s_x$

$= (2\alpha - 1) s_x$

Insertion $\Delta\Phi \le -\text{size}(\text{scapegoat})$

$= \Theta(\text{size}(\text{scapegoat})) + O(\log_{1/\alpha} q)$

if $\exists$ scapegoat x

Ignore $\Delta\Phi$ for all nodes except x

↳ all ≤ 0

$$\Phi = q + \frac{1}{(2\alpha-1)} \sum |s_L - s_R|$$

$\Delta\Phi^x$: $\Phi_i = O(1)$ $+q$

$\Phi_{i-1} > s_x$ $+q$

$s_L > \alpha s_x$

$s_R < (1-\alpha) s_x$

$s_L - s_R > [\alpha - (1-\alpha)] s_x$

$= (2\alpha - 1) s_x$

Deletion

$= \Theta(n) + O(\log_{1/\alpha} q)$

if $n < q/2$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]

↳ avoid having height = $f(q)$ but not $f(n)$

Insertion $\Delta\Phi < -\text{size}(\text{scapegoat})$

$= \Theta(\text{size}(\text{scapegoat})) + O(\log_{1/\alpha} q)$

if $\exists$ scapegoat x

Ignore $\Delta\Phi$ for all nodes except x

↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = O(1) + q$

$\Phi_{i-1} > S_x + q$

$S_L > \alpha S_x$

$S_R < (1-\alpha) S_x$

$S_L - S_R > [\alpha - (1-\alpha)] S_x$

$= (2\alpha - 1) S_x$

$$\Phi = q + \frac{1}{(2\alpha-1)} \sum |S_L - S_R|$$

Deletion

$= \Theta(n) + O(\log_{1/\alpha} q)$

if $n < q/2$

?

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion $\Delta\Phi < -\text{size}(\text{scapegoat})$

$= \Theta(\text{size}(\text{scapegoat})) + O(\log_{1/\alpha} q)$

if $\exists$ scapegoat x

Ignore $\Delta\Phi$ for all nodes except x

↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = O(1) + q$

$\Phi_{i-1} > S_x + q$

$S_L > \alpha S_x$

$S_R < (1-\alpha) S_x$

$S_L - S_R > [\alpha - (1-\alpha)] S_x$

$= (2\alpha - 1) S_x$

$$\Phi = q + \frac{1}{(2\alpha-1)} \sum |S_L - S_R|$$

Ignore $\Delta\Phi$ of Σ

↳ all ≤ 0

Deletion

$= \Theta(n) + O(\log_{1/\alpha} q)$

if $n < q/2$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]

↳ avoid having height = $f(q)$ but not $f(n)$

Insertion $\Delta\Phi < -\text{size}(\text{scapegoat})$

$= \Theta(\text{size}(\text{scapegoat})) + O(\log_{1/\alpha} q)$

if $\exists$ scapegoat x

Ignore $\Delta\Phi$ for all nodes except x

↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = O(1)$ $+q$

$\Phi_{i-1} > S_x$ $+q$

$S_L > \alpha S_x$

$S_R < (1-\alpha) S_x$

$S_L - S_R > [\alpha - (1-\alpha)] S_x$

$= (2\alpha - 1) S_x$

$$\Phi = q + \frac{1}{(2\alpha-1)} \sum |S_L - S_R|$$

Ignore $\Delta\Phi$ of Σ

↳ all ≤ 0

Deletion

$= \Theta(n) + O(\log_{1/\alpha} q)$

if $n < q/2$

$\Phi_{i-1} \sim 2n$

$\Phi_i = n$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

Insertion $\Delta\Phi < -\text{size}(\text{scapegoat})$

$= \Theta(\text{size}(\text{scapegoat})) + O(\log_{1/\alpha} q)$

if $\exists$ scapegoat x

Ignore $\Delta\Phi$ for all nodes except x

↳ all ≤ 0

$\Delta\Phi^x$: $\Phi_i = O(1) + q$

$\Phi_{i-1} > S_x + q$

$S_L > \alpha S_x$

$S_R < (1-\alpha) S_x$

$S_L - S_R > [\alpha - (1-\alpha)] S_x$

$= (2\alpha - 1) S_x$

$$\Phi = q + \frac{1}{(2\alpha-1)} \sum |S_L - S_R|$$

Ignore $\Delta\Phi$ of Σ

↳ all ≤ 0

Deletion $\Delta\Phi < -n$

$= \Theta(n) + O(\log_{1/\alpha} q)$

if $n < q/2$

$\Phi_{i-1} \sim 2n$

$\Phi_i = n$

- doesn't cause a height problem
- rebuild entire tree if $n < q/2$ [& reset q]
  ↳ avoid having height = $f(q)$ but not $f(n)$

SCAPEGOAT TREES : amortized balanced dynamic BST

Insert, Delete : $O(\log_{1/\alpha} n)$ amortized