Applications of amortized analysis

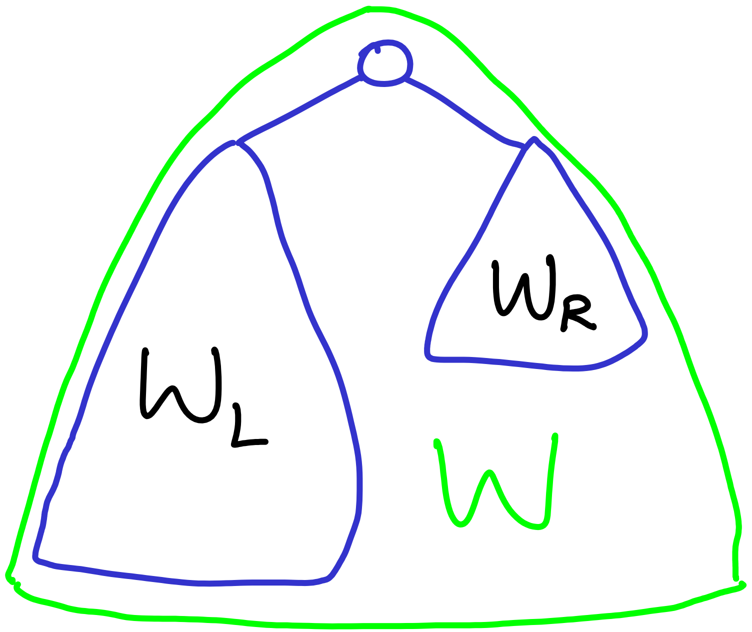
Binary search trees of bounded balance α $\rightarrow$ BB(α)

(weight-balanced trees)

Binary search trees of bounded balance α $\rightarrow$ BB(α)

(weight-balanced trees)

node with subtree weight $\underline{W}$ and — left subtree weight W_L
— right subtree weight W_R

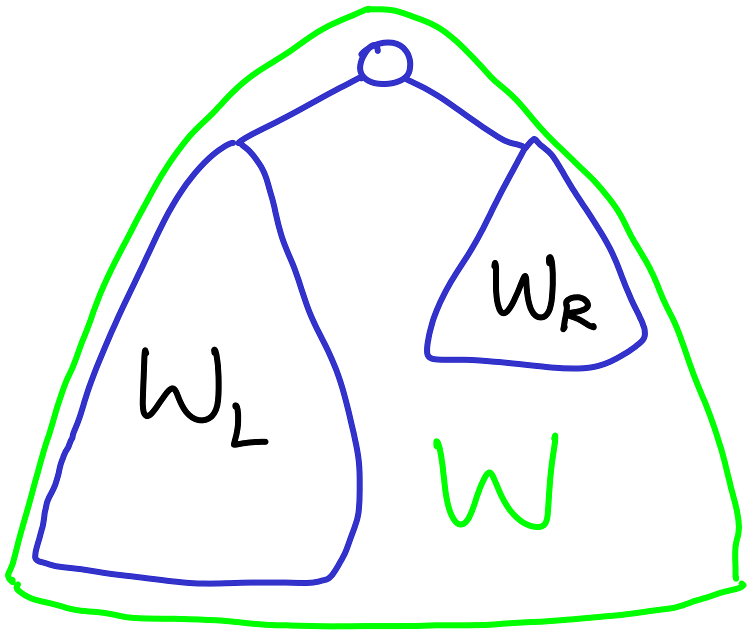


Binary search trees of bounded balance α $\rightarrow$ BB(α)

(weight-balanced trees)

For every node with subtree weight $\underline{W}$ and
- left subtree weight W_L
- right subtree weight W_R

$$\boxed{\begin{array}{l} W_L \geq \alpha W \\ W_R \geq \alpha W \end{array} \quad 0 < \alpha \leq \frac{1}{2}}$$



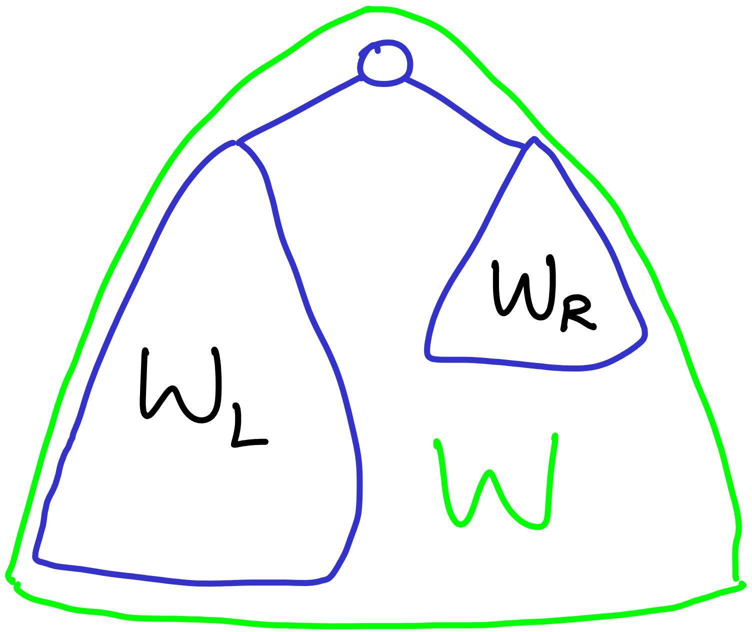
Binary search trees of bounded balance α → BB(α)

(weight-balanced trees)

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

$$\boxed{\begin{aligned} W_L &\geq \alpha W \\ W_R &\geq \alpha W \end{aligned} \quad 0 < \alpha \leq \tfrac{1}{2}}$$

Height: ?



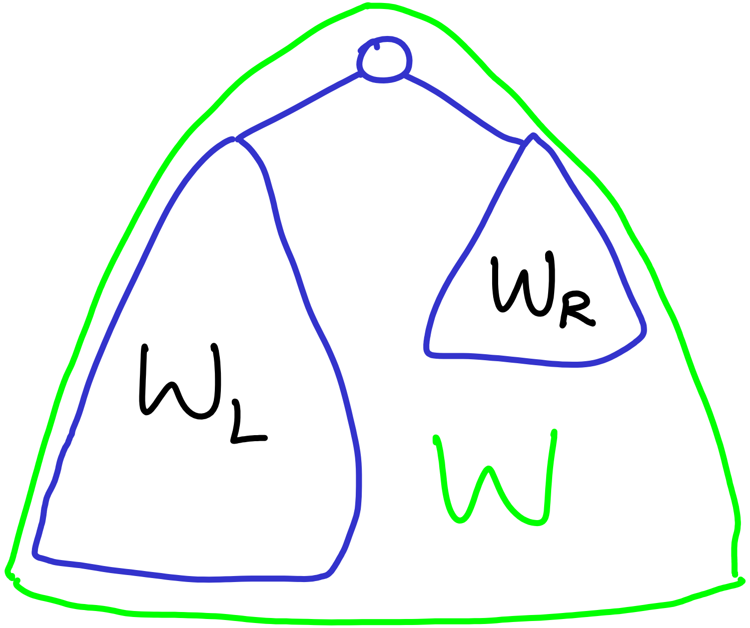
Binary search trees of bounded balance α $\rightarrow$ BB(α)

(weight-balanced trees)

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

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

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

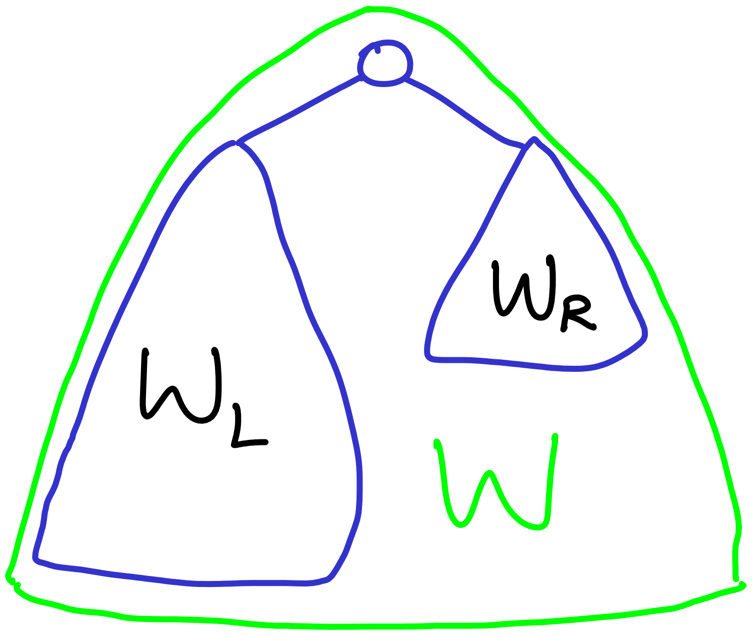


Binary search trees of bounded balance α $\rightarrow$ $BB(\alpha)$

(weight-balanced trees)

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

$$\boxed{\begin{aligned} W_L &\geq \alpha W \\ W_R &\geq \alpha W \end{aligned} \qquad 0 < \alpha \leq \tfrac{1}{2}}$$



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

standard geometric series:

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

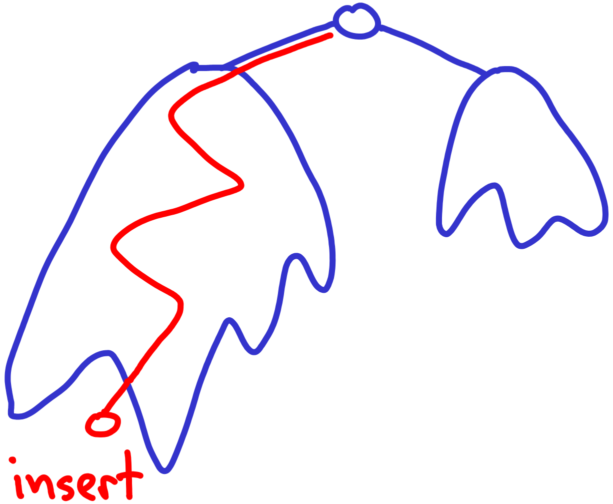
Insertion / Deletion in BB(α) weight-balanced tree

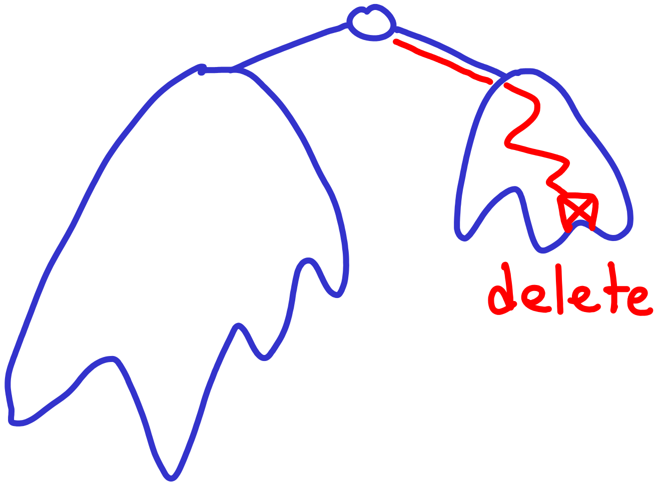
Insertion / Deletion in BB(α) weight-balanced tree

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

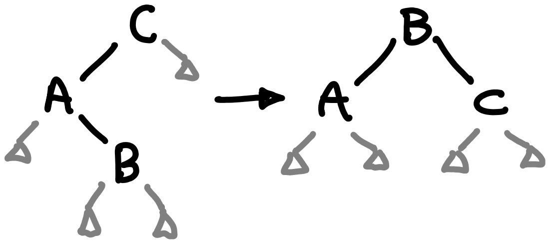
Any ancestor could now have:

$W_R < \alpha W$ (wlog)

Solution: rotate

FYI

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



... depends on amount of imbalance

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

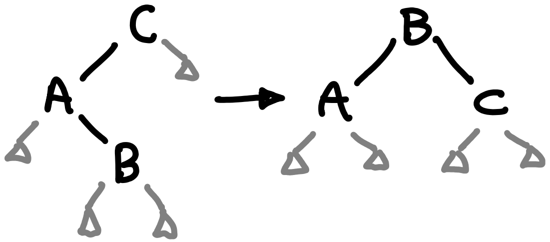
Any ancestor could now have:

$W_R < \alpha W$ (wlog)

Solution: rotate

FYI

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



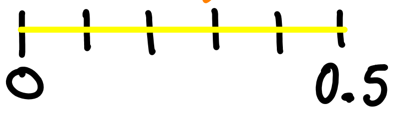
... depends on amount of imbalance

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

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



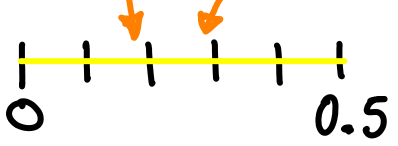
FYI

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

lower bound:

$\frac{2}{11} < \alpha$

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

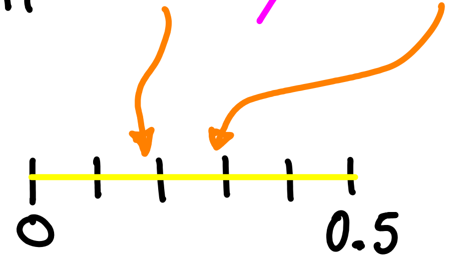


FYI

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

lower bound:

$\frac{2}{11} < \alpha$ $\quad$ $\alpha < 1 - \frac{1}{\sqrt{2}}$



FYI

Claim:

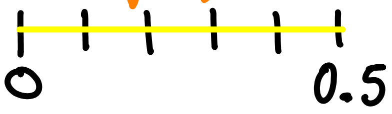
can rebalance for α outside this range with more complicated rotations

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

lower bound:

$\frac{2}{11} < \alpha$

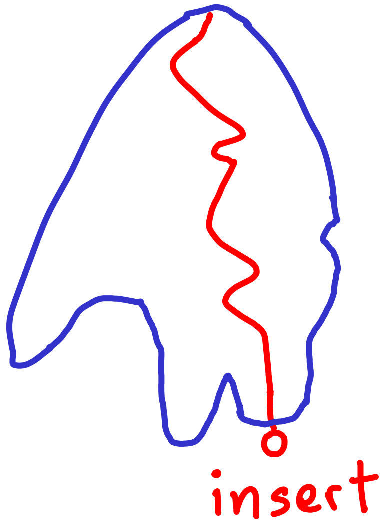
$\alpha < 1 - \frac{1}{\sqrt{2}}$

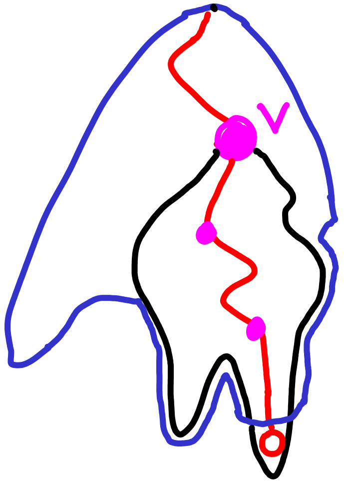


Claim:

can rebalance for α outside this range with more complicated rotations

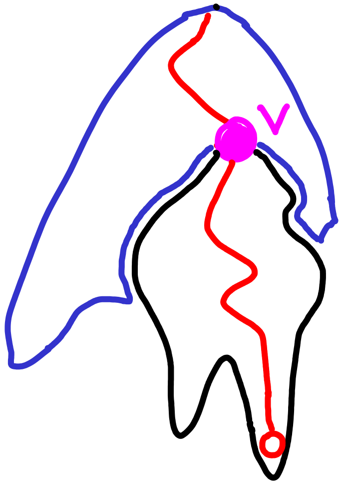
Instead, we will avoid rotations


insert



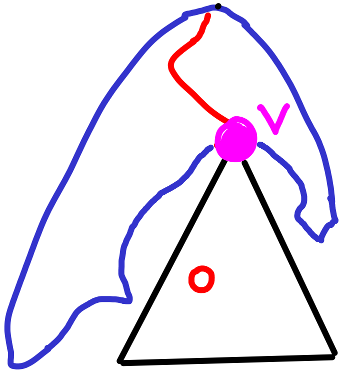
α-violation may occur for any ancestor

Let v be highest violation



α-violation may occur for any ancestor

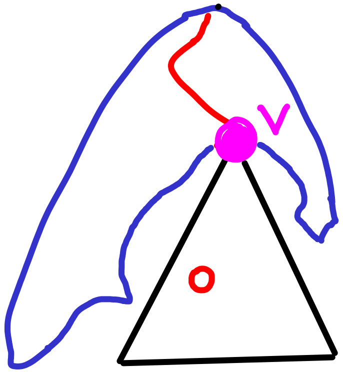
Let v be highest violation



α- violation may occur for any ancestor

Let v be highest violation

Rebuild subtree(v)   cost ?

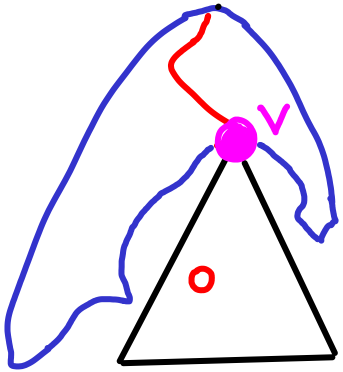


α-violation may occur for any ancestor

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

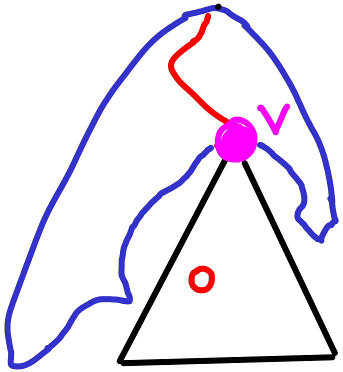
Let v be highest violation

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

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

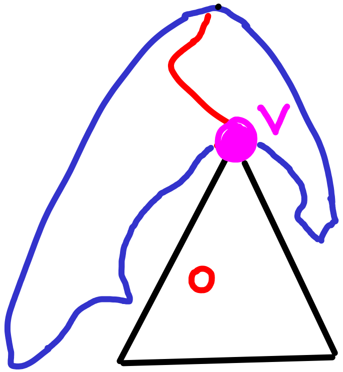
Rebuild subtree(v) $\quad O(\log n) + \Theta(\text{size}(v)) = O(n)$

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

(diff = 1 unavoidable)

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

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



α-violation may occur for any ancestor

Let v be highest violation

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

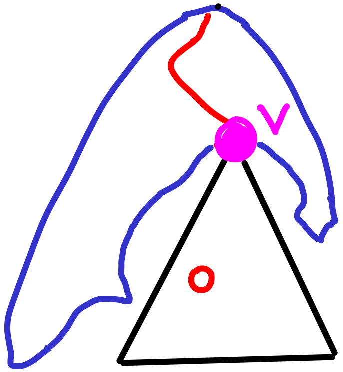
$$\Phi = \sum_{\substack{\text{all}\\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R|$$

$\rightarrow$ technically, only if diff ≥ 2

(diff = 1 unavoidable)

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

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



α-violation may occur for any ancestor

Let v be highest violation

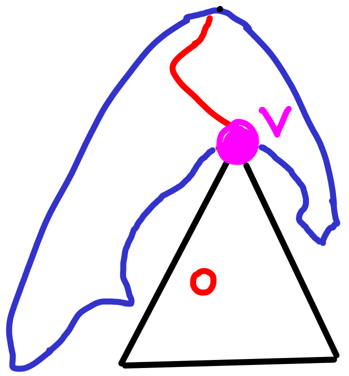
Rebuild subtree(v) $\quad O(\log n) + \Theta(\text{size}(v)) = O(n)$

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

(diff = 1 unavoidable)

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

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



α-violation may occur for any ancestor

Let v be highest violation

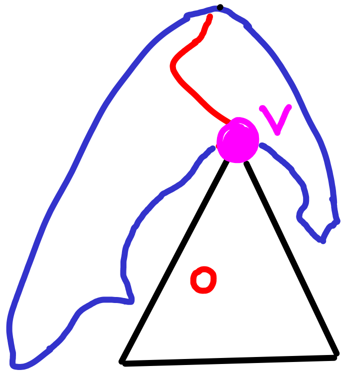
Rebuild subtree(v) $\quad O(\log n) + \Theta(\text{size}(v)) = O(n)$

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

(diff = 1 unavoidable)

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

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



α-violation may occur for any ancestor

Let v be highest violation

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

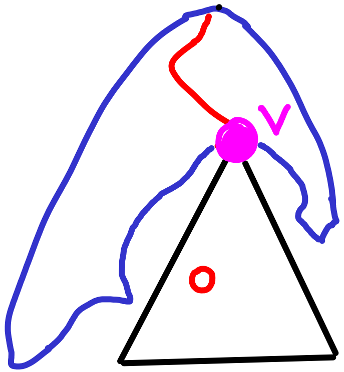
$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R|$$ $\rightarrow$ technically, only if diff ≥ 2

(diff = 1 unavoidable)

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

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

every $\Delta\Phi$ term ≤ 0



α-violation may occur for any ancestor

Let v be highest violation

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

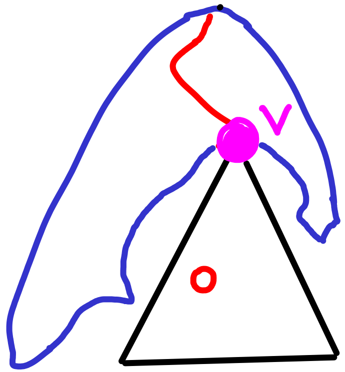
$$\Phi = \sum_{\substack{\text{all} \\ \text{nodes}}} \frac{1}{1-2\alpha} |W_L - W_R| \rightarrow$$ technically, only if diff ≥ 2

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

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

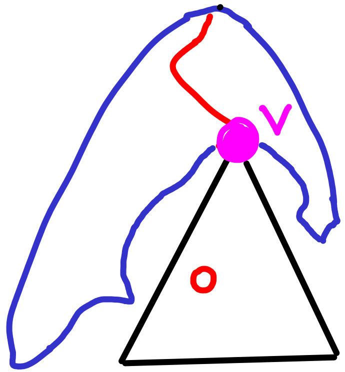
(diff = 1 unavoidable)

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

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

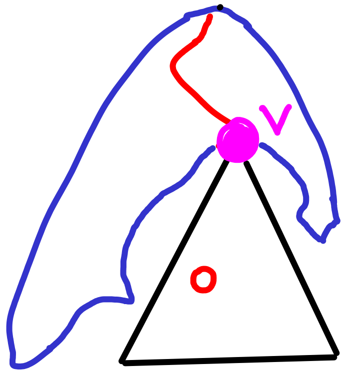
(diff = 1 unavoidable)

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

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

every $\Delta\Phi$ term ≤ 0

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



α- violation may occur for any ancestor

Let v be highest violation

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

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

(diff = 1 unavoidable)

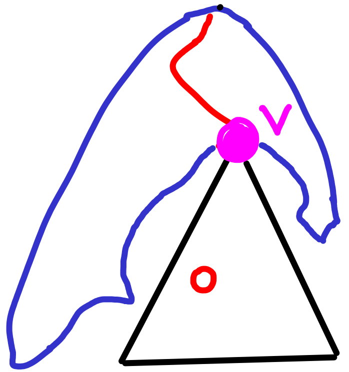
e.g. $\alpha = \frac{1}{3}$: $\Phi = 3 \cdot \sum |W_L - W_R|$

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

$$\underbrace{\tfrac{1}{3}W}_{\Phi_{i-1}}$$

violation: $\boxed{\text{wlog } W_R < \tfrac{1}{3}W < \tfrac{2}{3}W < W_L}$



α- violation may occur for any ancestor

Let v be highest violation

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

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

$\rightarrow$ technically, only if diff ≥ 2

(diff = 1 unavoidable)

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

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

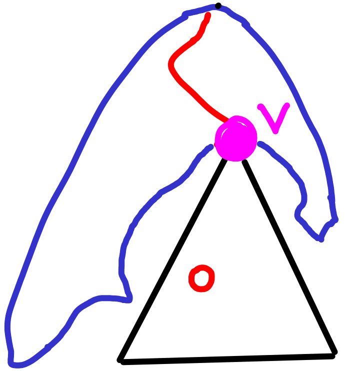
every $\Delta\Phi$ term ≤ 0

$$\leq 3 \cdot [\underset{\downarrow}{0} - \underbrace{\tfrac{1}{3} W}] \sim -\text{size}(v)$$

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

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

$\square$



α- violation may occur for any ancestor

Let v be highest violation

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

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

(diff = 1 unavoidable)

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

every $\Delta\Phi$ term ≤ 0

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

$\downarrow$ Φ_i

violation: $\boxed{W_R < \alpha W < (1-\alpha) W < W_L}$

wlog

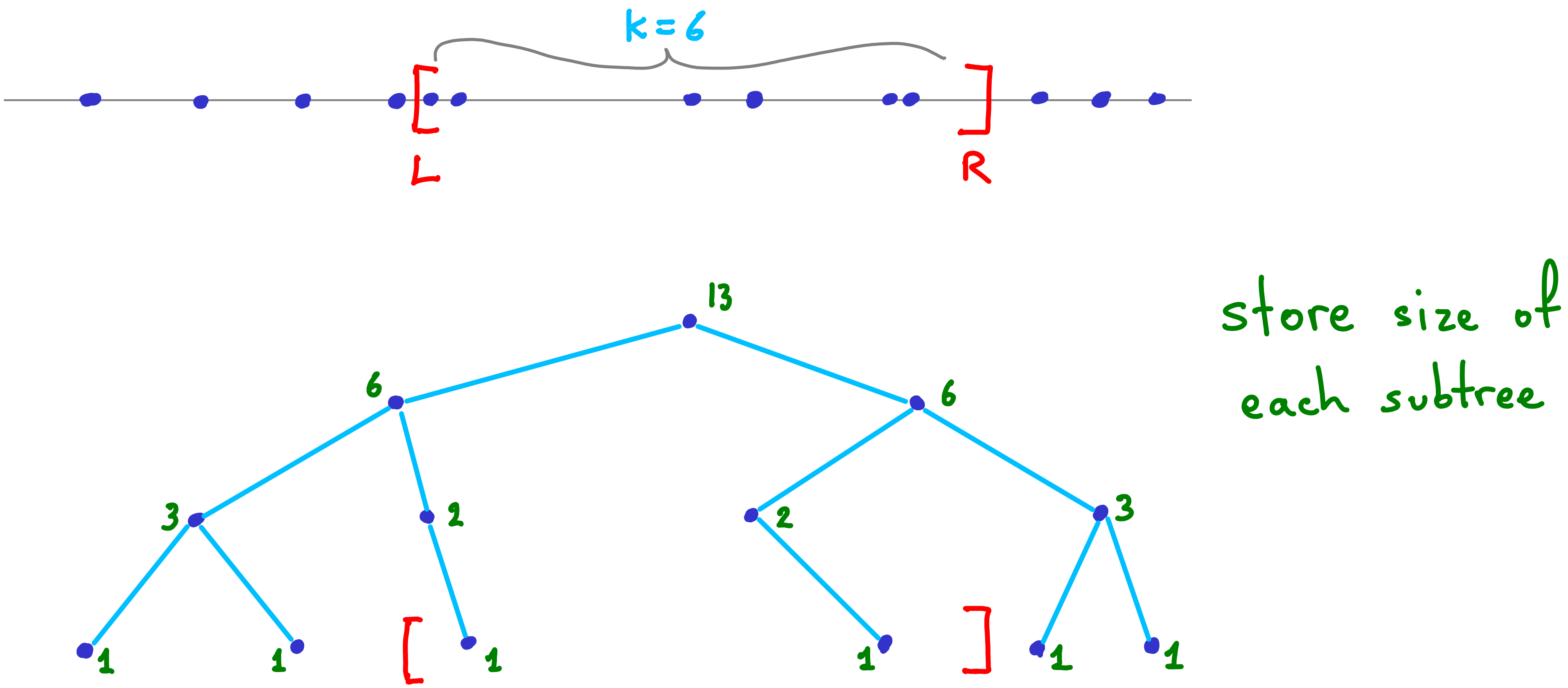
$\square$

An application of BB(α)

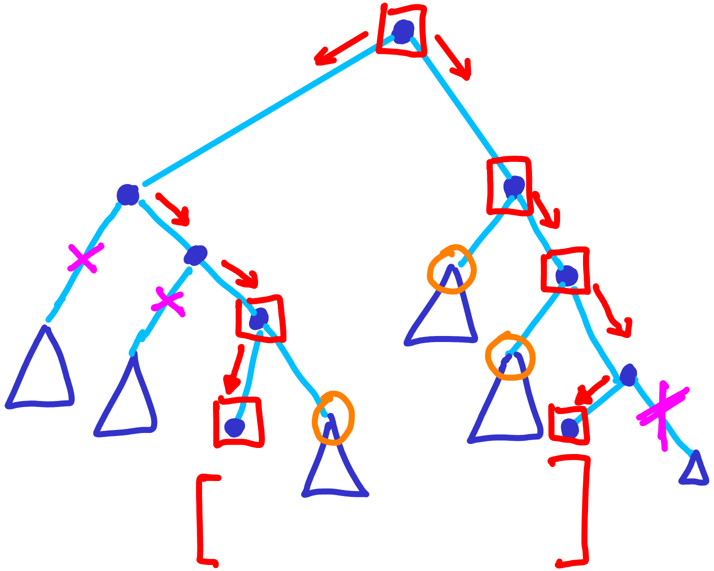
Dynamic high-dimensional multilayered range trees

↪ not easy to update efficiently with rotations

FYI

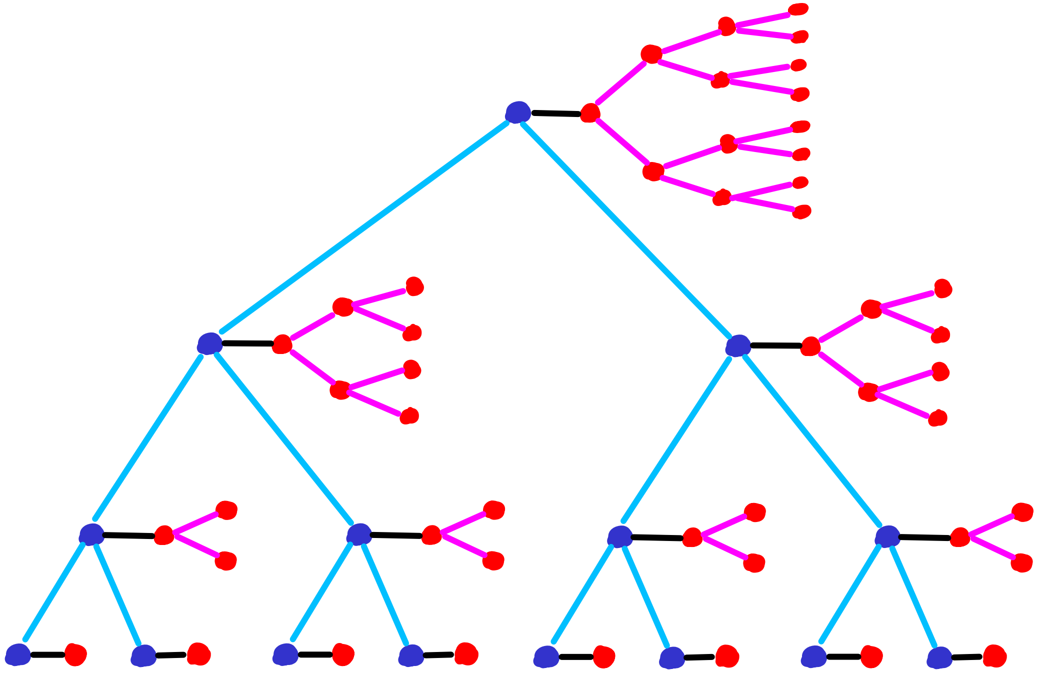


1D range counting



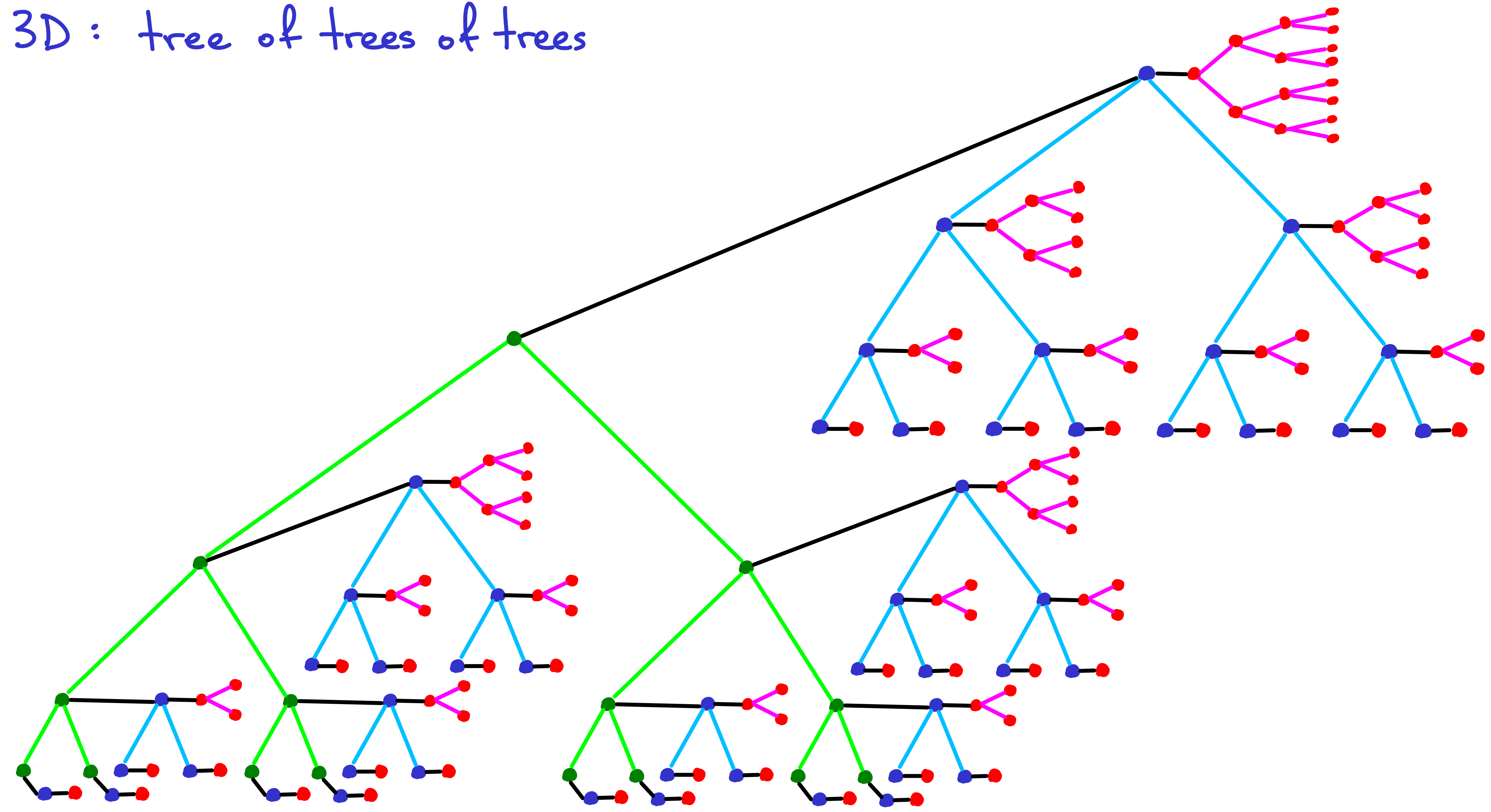
Every X-range is represented by O(logn) nodes

For each node, create a new (aux.) tree containing all nodes of subtree, sorted by Y.



2D range counting

3D : tree of trees of trees

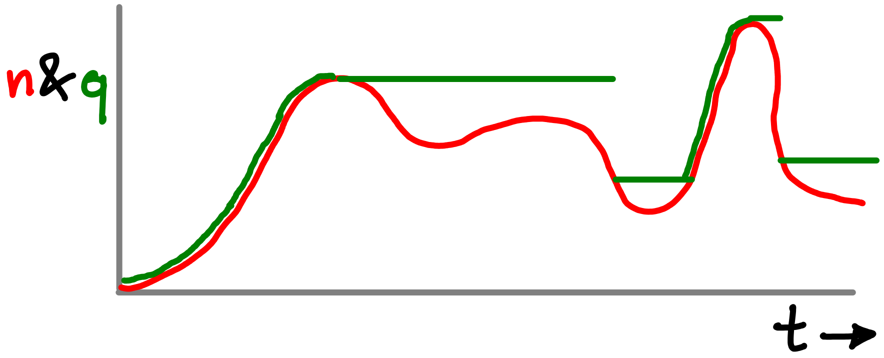


More results that use amortized analysis

SCAPEGOAT TREES : amortized balanced dynamic BST

n = # keys

q = variable s.t. $q/2 \le n \le q$ →



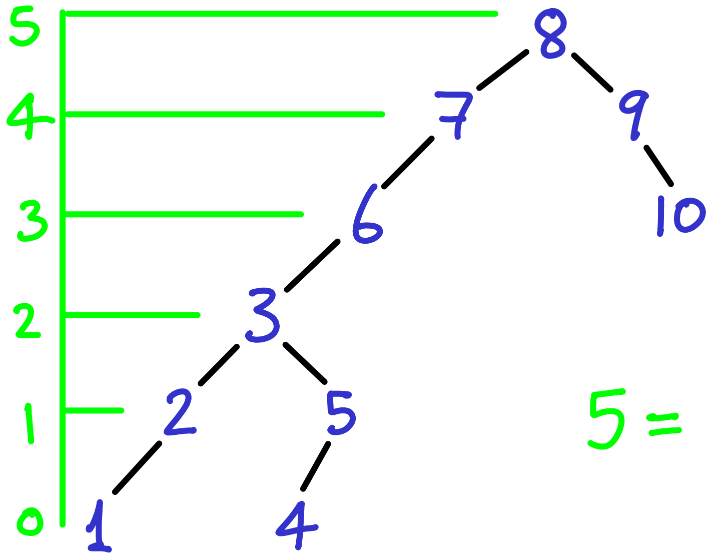
α = balance factor,

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

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 $q=10$

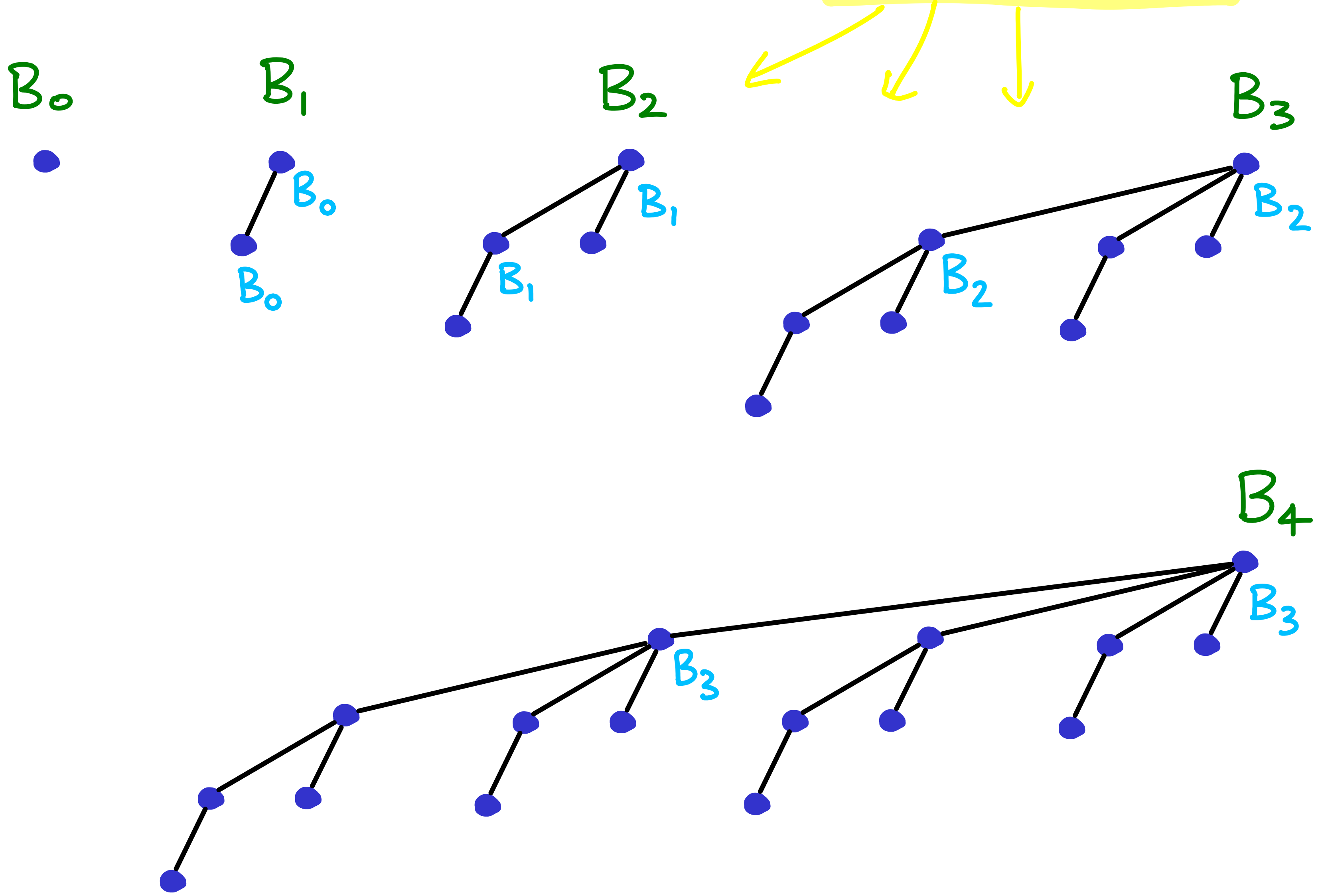
OK for $q > 10$

Heap building: we have seen $O(n)$ instead of $O(n\log n)$

Heap building: we have seen $O(n)$ instead of $O(n\log n)$

Binomial heaps: collections of binomial trees

BINOMIAL TREES



Heap building: we have seen $O(n)$ instead of $O(n\log n)$

Binomial heaps: collections of binomial trees

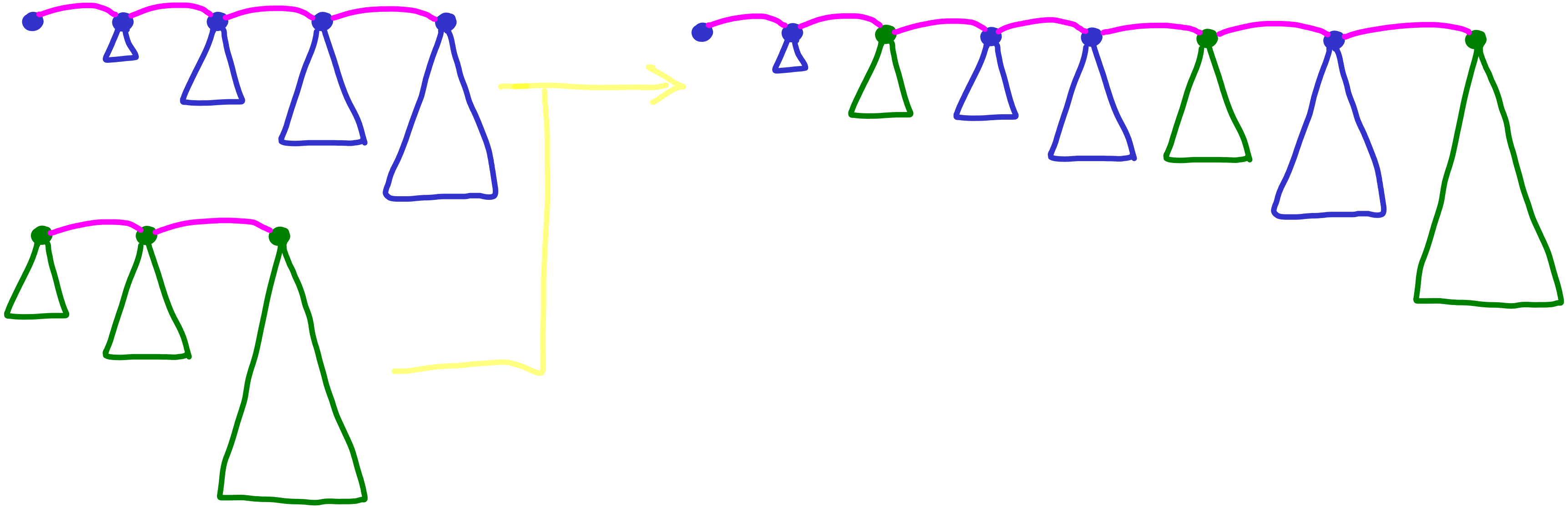
size = power of 2
only one of each size
trivially mergeable

Heap building: we have seen O(n) instead of O(nlogn)

Binomial heaps: collections of binomial trees

size = power of 2
only one of each size
trivially mergeable

merging two b.heaps: like binary addition … O(logn)



Heap building: we have seen O(n) instead of O(nlogn)

Binomial heaps: collections of binomial trees

size = power of 2
only one of each size
trivially mergeable

merging two b.heaps: like binary addition … O(logn)

insert into b.heap: O(logn) worst case

Heap building: we have seen O(n) instead of O(nlogn)

Binomial heaps: collections of binomial trees

size = power of 2
only one of each size
trivially mergeable

merging two b.heaps: like binary addition … O(logn)

insert into b.heap: O(logn) worst case

but O(1) amortized

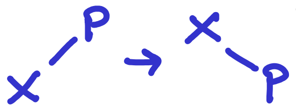
HEAPS:	REGULAR	BINOMIAL	FIBONACCI
REPORT MIN:	$O(1)$	←	←
EXTRACT MIN:	$O(\log n)$	←	$O(n)$ $O(\log n)$ amortized
INSERT:	$O(\log n)$ $O(1)$ amortized	←	$O(1)$
DECREASE KEY:	$O(\log n)$	←	$O(n)$ * $O(1)$ amortized
DELETE:	$O(\log n)$	←	$O(n)$ $O(\log n)$ amortized
MERGE/UNION:	$O(n)$	$O(\log n)$	* $O(1)$

HEAPS:

	REGULAR	BINOMIAL	QUAKE
REPORT MIN:	$O(1)$	←	←
EXTRACT MIN:	$O(\log n)$	←	$O(n)$ $O(\log n)$ amortized
INSERT:	$O(\log n)$ $O(1)$ amortized	←	$O(1)$
DECREASE KEY:	$O(\log n)$	←	$O(1)$
DELETE:	$O(\log n)$	←	$O(n)$ $O(\log n)$ amortized
MERGE/UNION:	$O(n)$	$O(\log n)$	$O(1)$

Splay trees: whenever you access a node, bring it to the top

Amortized cost O(logn)

(if no G then )