Splay Trees

Sleator & Tarjan
1985

SPLAY TREES

Sleator & Tarjan
1985

- BST with no extra info
- Tries to stay balanced via rotations

SPLAY TREES

Sleator & Tarjan
1985

- BST with no extra info
- Tries to stay balanced via rotations

Idea: move recently accessed keys to the top

↳ not a guarantee for rebalancing, but over many operations it works

SPLAY TREES

Sleator & Tarjan
1985

- BST with no extra info
- Tries to stay balanced via rotations

Idea: move recently accessed keys to the top

↳ not a guarantee for rebalancing, but over many operations it works

↳ nice feature: frequently accessed items can take $o(\log n)$

SPLAY TREES

Sleator & Tarjan
1985

- BST with no extra info
- Tries to stay balanced via rotations

Idea: move recently accessed* keys to the top

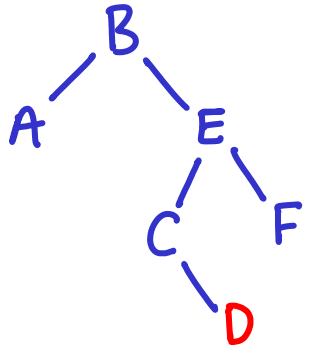
↳ not a guarantee for rebalancing, but over many operations it works

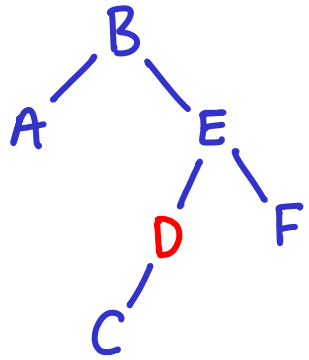
↳ nice feature: frequently accessed items can take $o(\log n)$

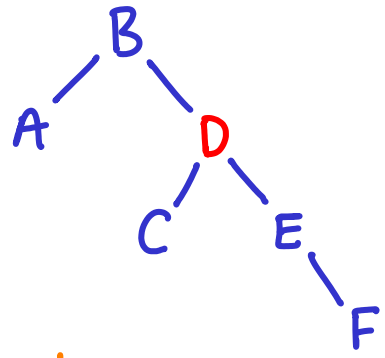
*More specifically, if already taking time t to search,
(even for delete and unsuccessful search)

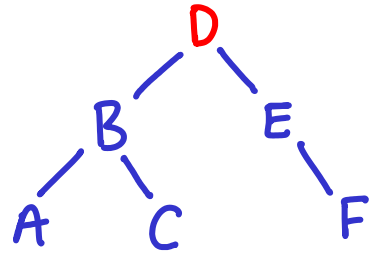
use $O(t)$ time to do rotations

Easy to move to top with rotations



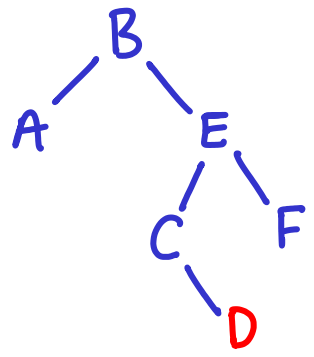


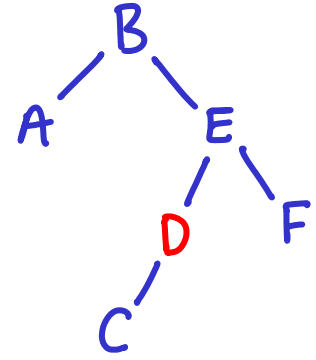


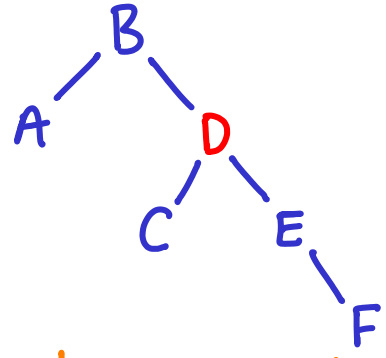


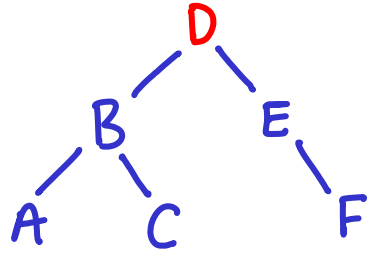
But that's another method, not splay tree

Easy to move to top with rotations





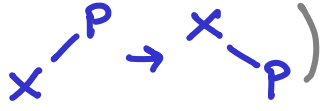




But that's another method, not splay tree

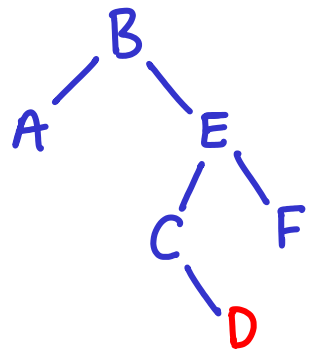
Splaying is subtly different

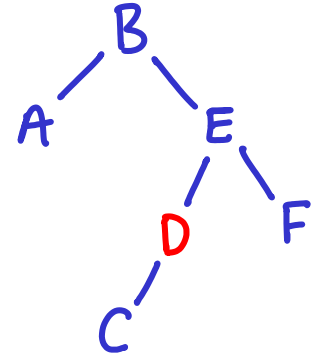
Look at current node, x, its parent and grandparent (if ∃).

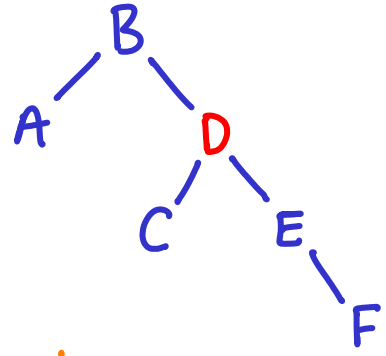
(if no G then 

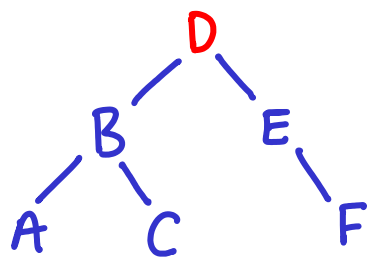
)

Easy to move to top with rotations





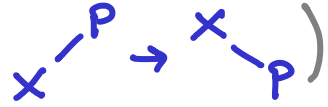




But that's another method, not splay tree

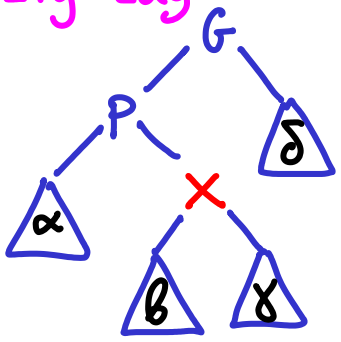
Splaying is subtly different

Look at current node, x, its parent and grandparent (if ∃).
Rotate twice, but rotation order depends on geometry.

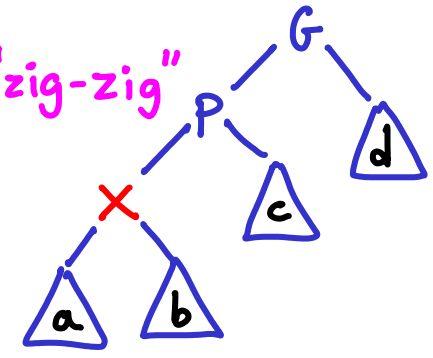
(if no G then 

)

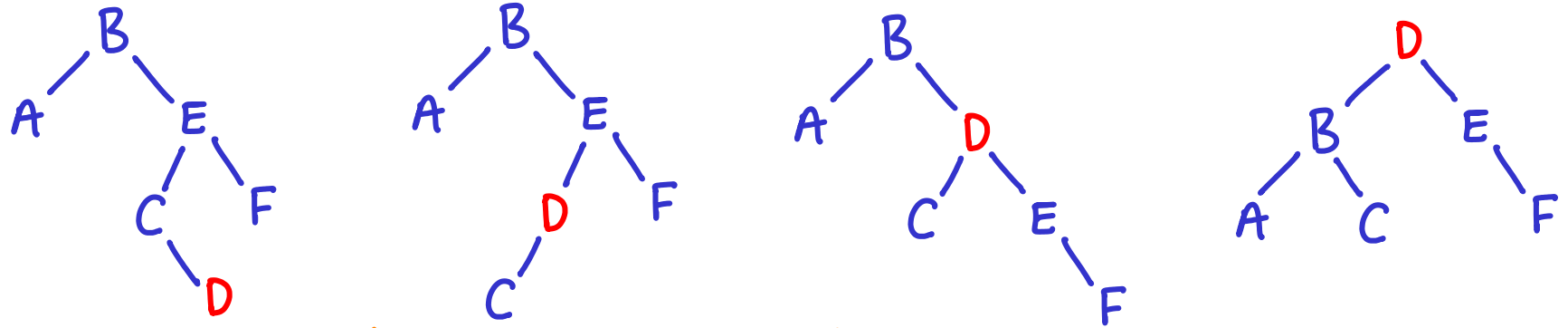
"zig-zag"



"zig-zig"



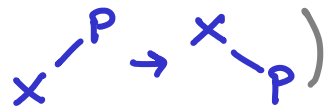
Easy to move to top with rotations



But that's another method, not splay tree

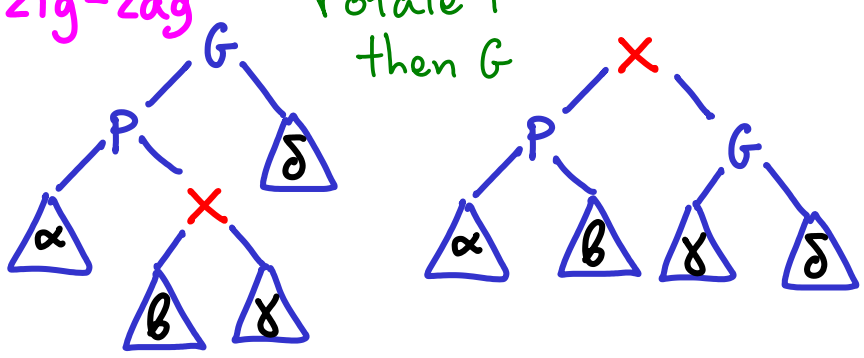
Splaying is subtly different

Look at current node, x, its parent and grandparent (if ∃).
Rotate twice, but rotation order depends on geometry.

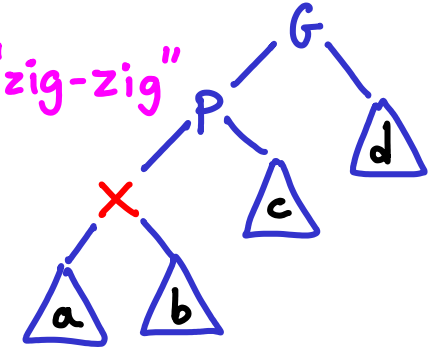
(if no G then 

)

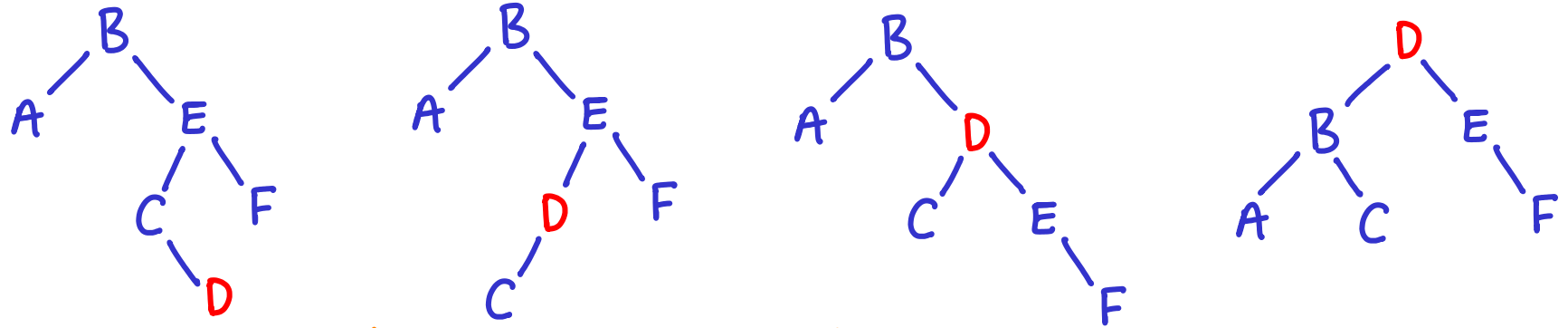
"zig-zag" rotate P then G



"zig-zig"



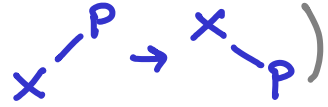
Easy to move to top with rotations



But that's another method, not splay tree

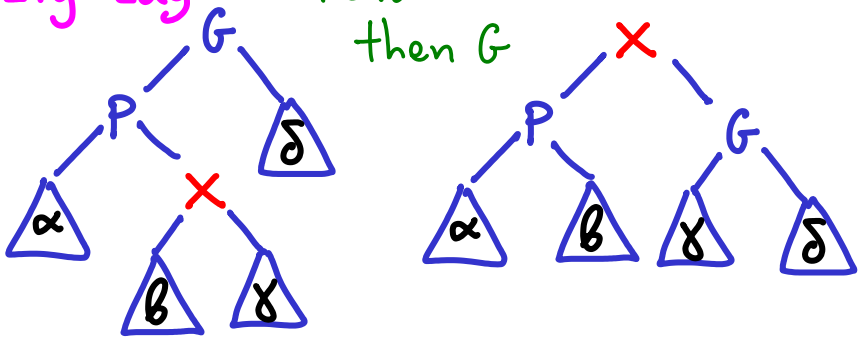
Splaying is subtly different

Look at current node, x, its parent and grandparent (if ∃).
Rotate twice, but rotation order depends on geometry.

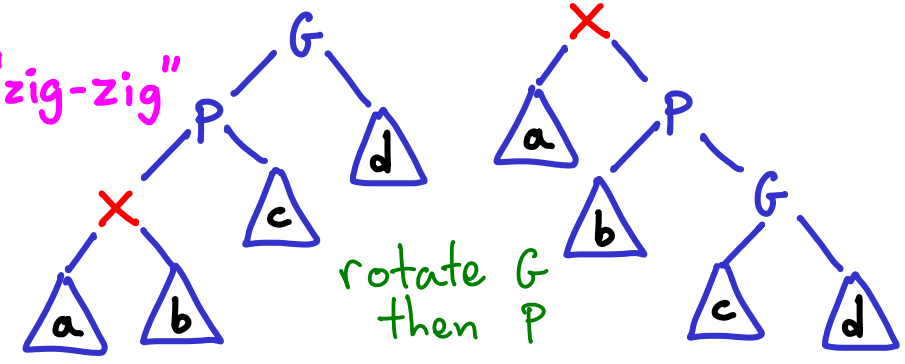
(if no G then 

)

"zig-zag" rotate P then G



"zig-zig"

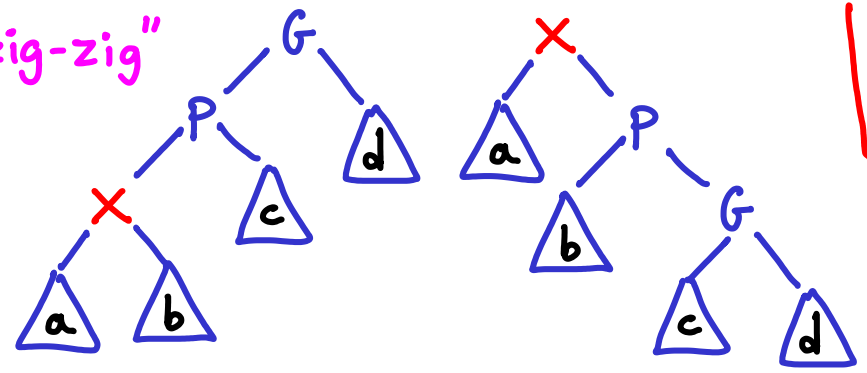


- Splay node x if we access it (search or insert)
- Splay parent of any deleted node (possibly parent of successor)
- Splay last visited node if a search is unsuccessful

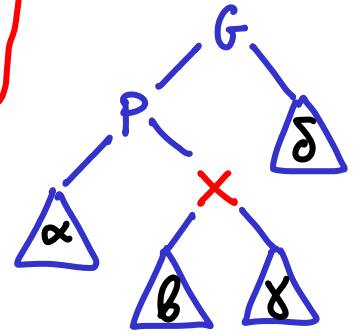
Bottom line: splay up (to root) from deepest position of search
(the search buys us splaying time)

There are other variants, e.g. splay-before-delete or top-down splaying

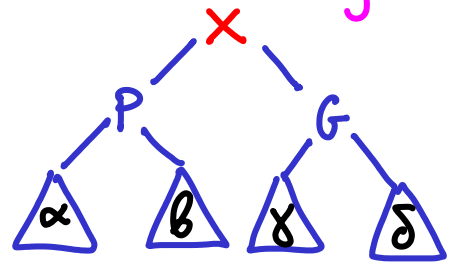
"zig-zig"



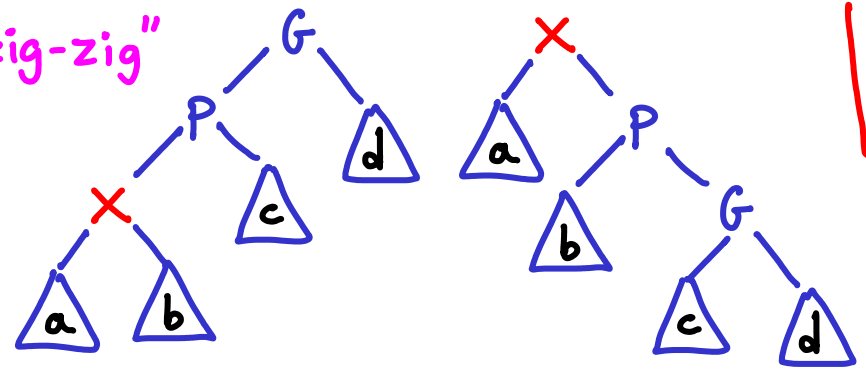
What does one step accomplish?



"zig-zag"



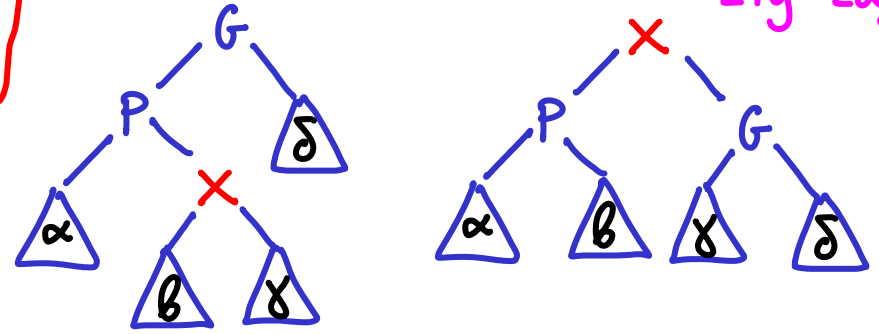
"zig-zig"



looks the same

What does one step accomplish?

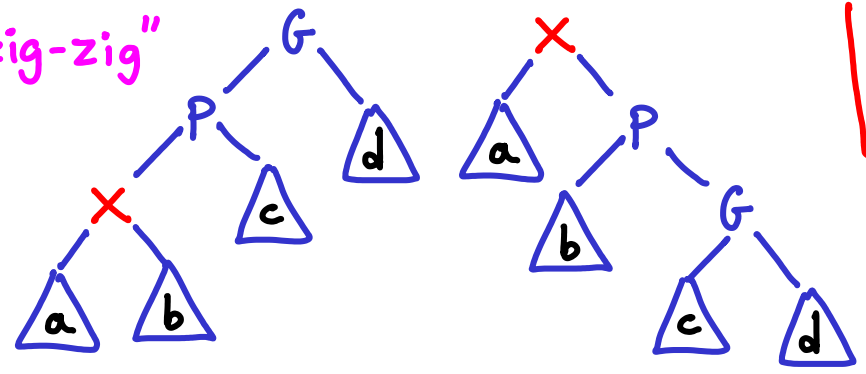
"zig-zag"



sort of has a tendency to decrease depth

?

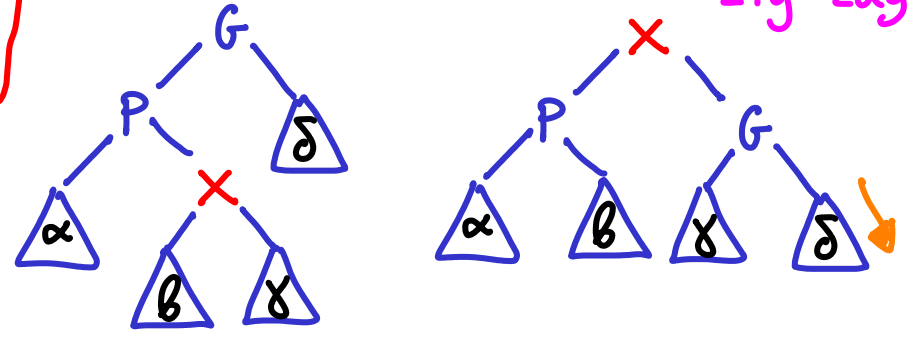
"zig-zig"



looks the same

What does one step accomplish?

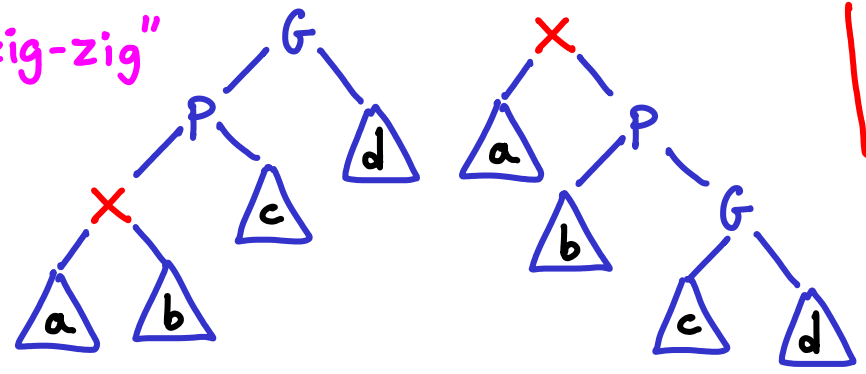
"zig-zag"



sort of has a tendency to decrease depth

but δ could be large and/or cause greater max-depth

"zig-zig"

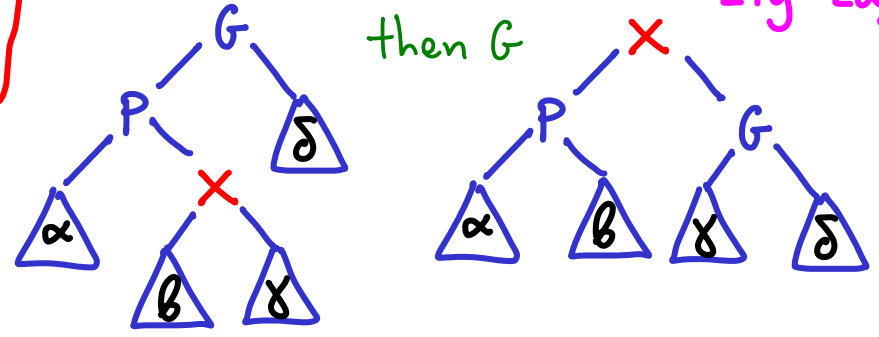


looks the same

What does one step accomplish?

rotate P then G

"zig-zag"



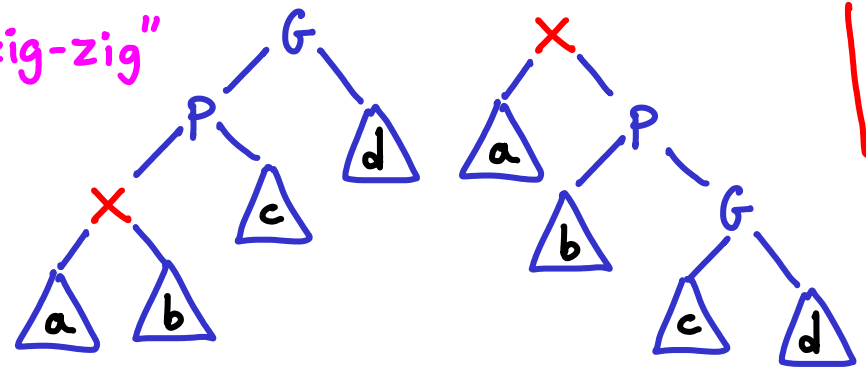
sort of has a tendency to decrease depth

but δ could be large and/or cause greater max-depth

insert keys alphabetically

A

"zig-zig"

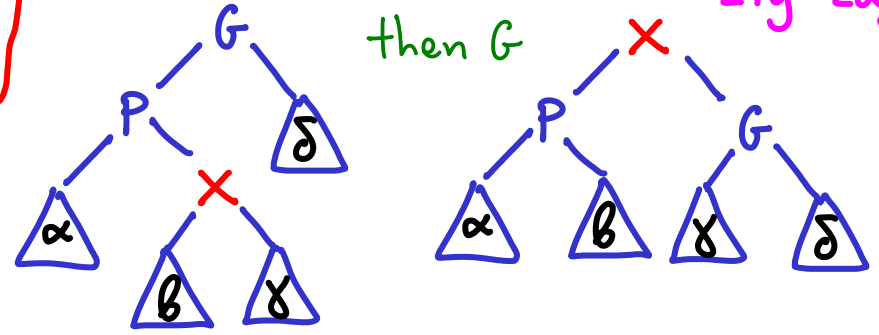


looks the same

What does one step accomplish?

rotate P then G

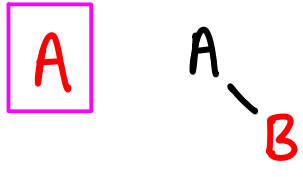
"zig-zag"



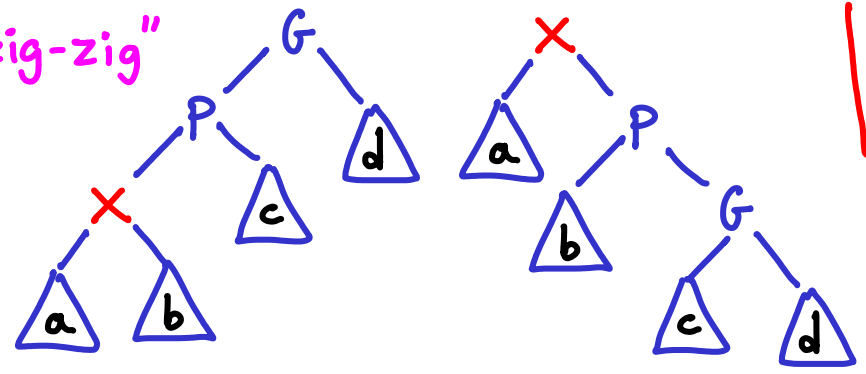
sort of has a tendency to decrease depth

but δ could be large and/or cause greater max-depth

insert keys alphabetically



"zig-zig"

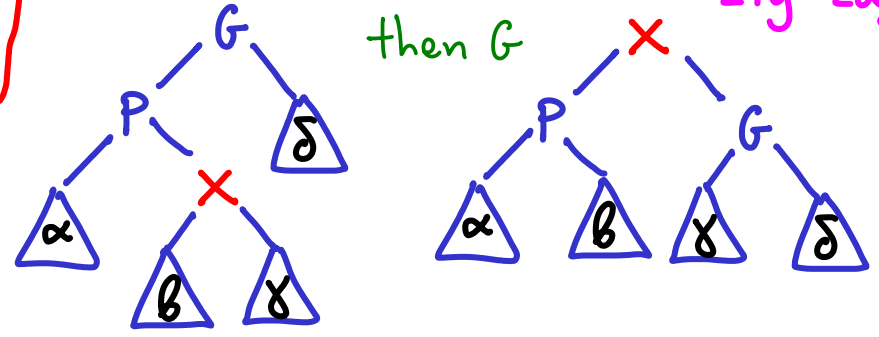


looks the same

What does one step accomplish?

rotate P then G

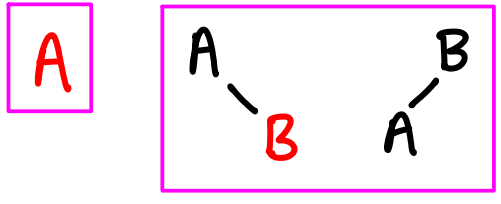
"zig-zag"



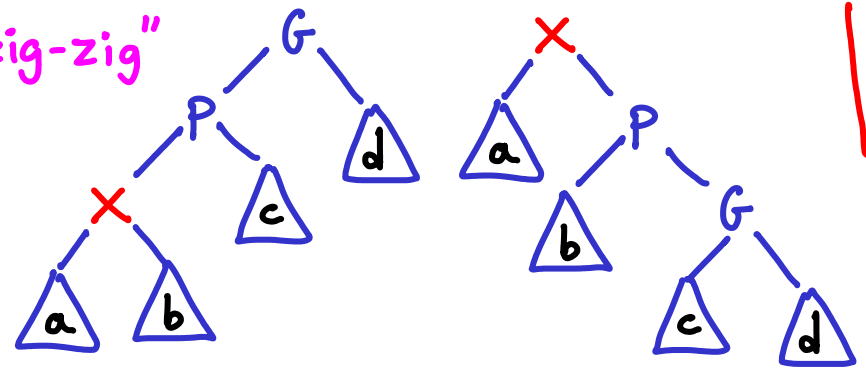
sort of has a tendency to decrease depth

but δ could be large and/or cause greater max-depth

insert keys alphabetically



"zig-zig"

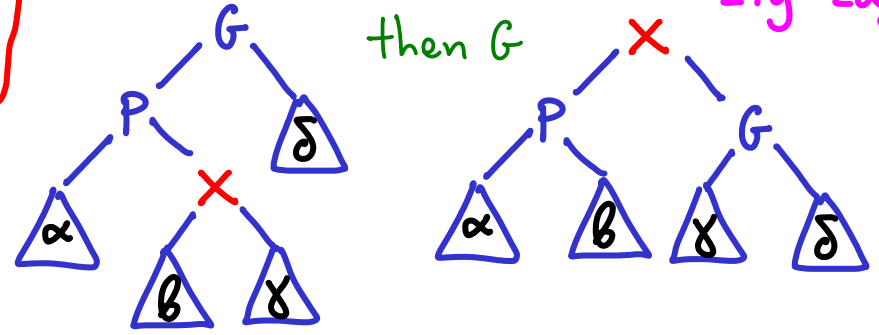


looks the same

What does one step accomplish?

rotate P then G

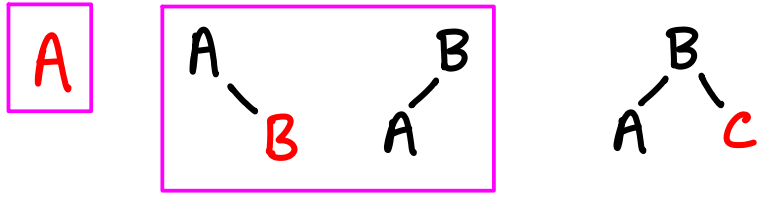
"zig-zag"



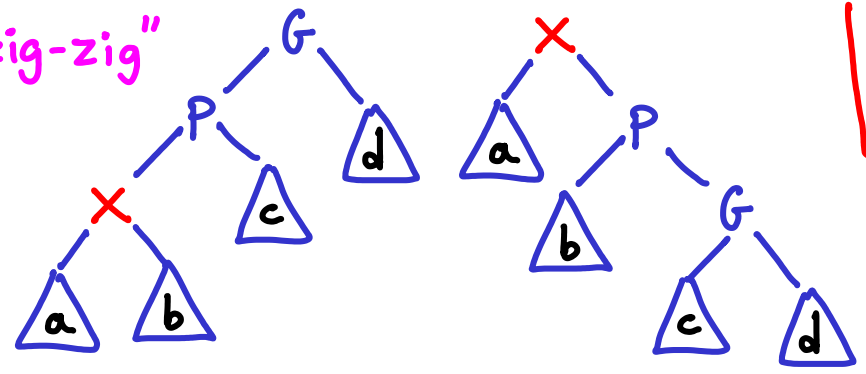
sort of has a tendency to decrease depth

but δ could be large and/or cause greater max-depth

insert keys alphabetically



"zig-zig"

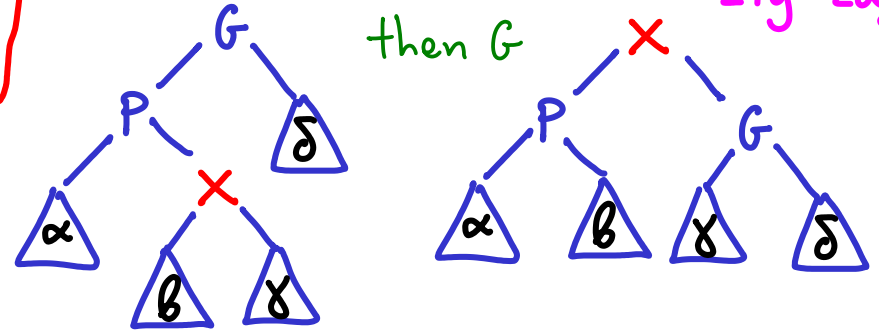


looks the same

What does one step accomplish?

rotate P then G

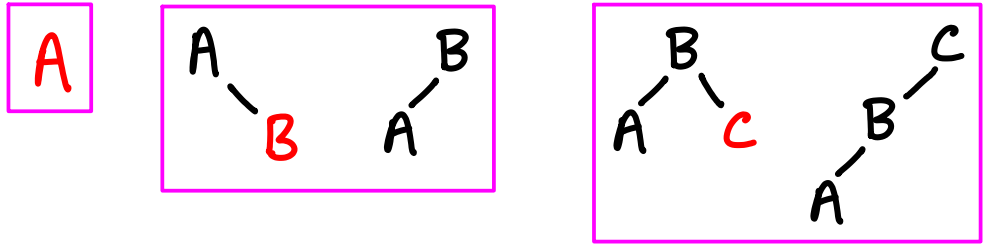
"zig-zag"



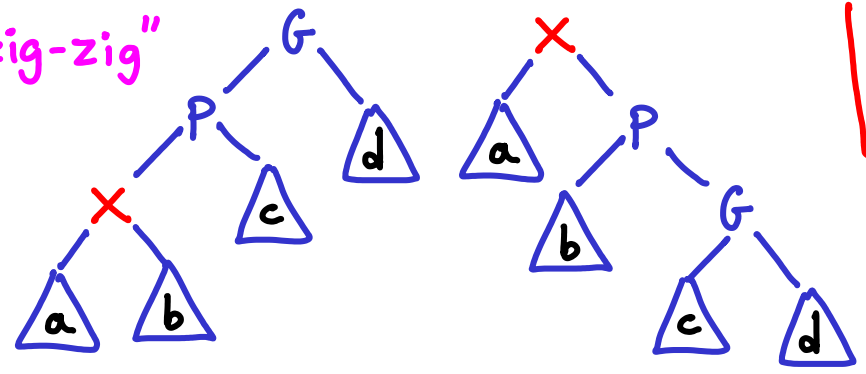
sort of has a tendency to decrease depth

but δ could be large and/or cause greater max-depth

insert keys alphabetically



"zig-zig"

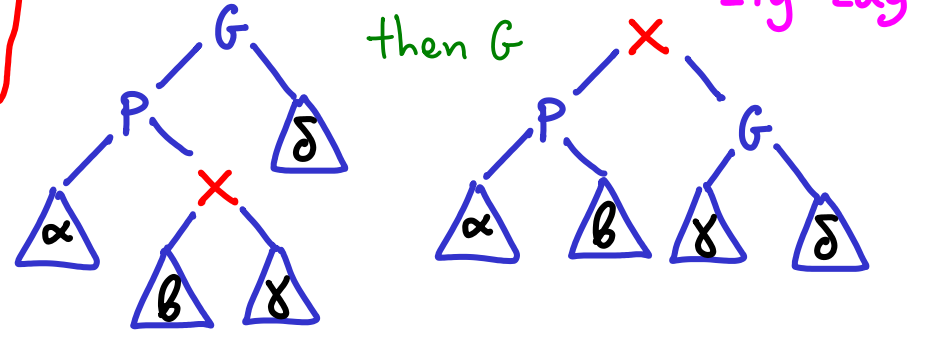


looks the same

What does one step accomplish?

rotate P then G

"zig-zag"

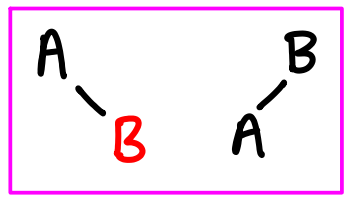


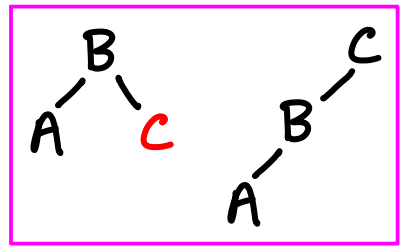
sort of has a tendency to decrease depth

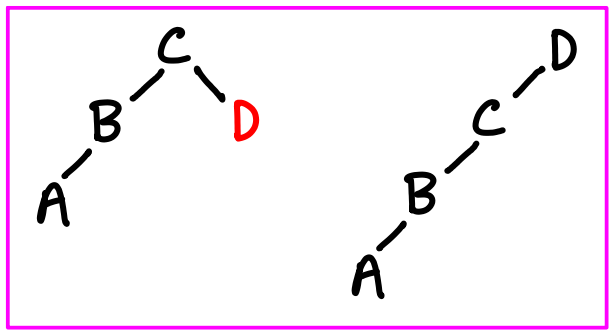
but δ could be large and/or cause greater max-depth

insert keys alphabetically

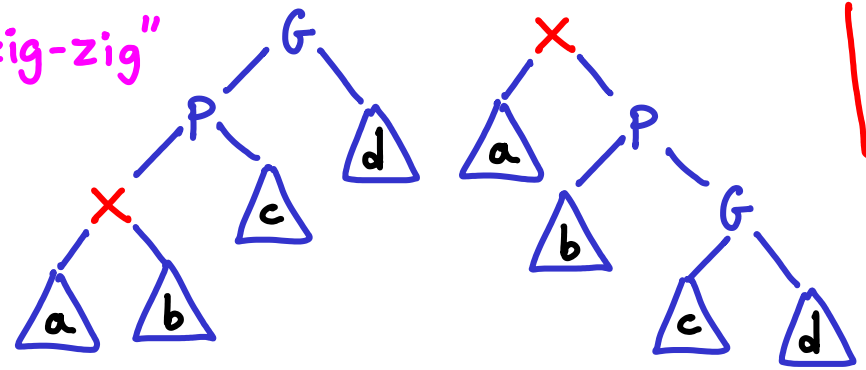
↳ depth = n







"zig-zig"

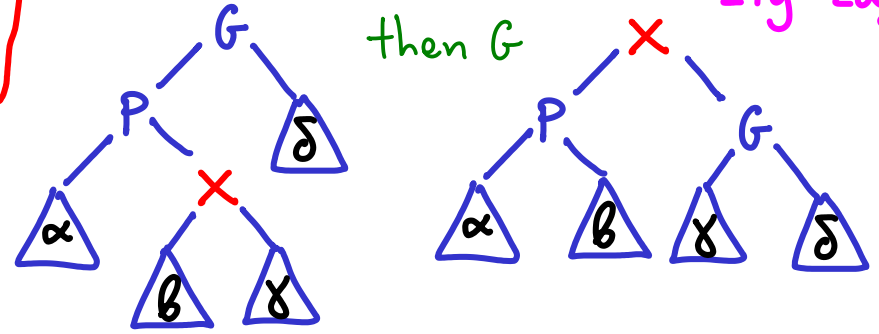


looks the same

What does one step accomplish?

rotate P then G

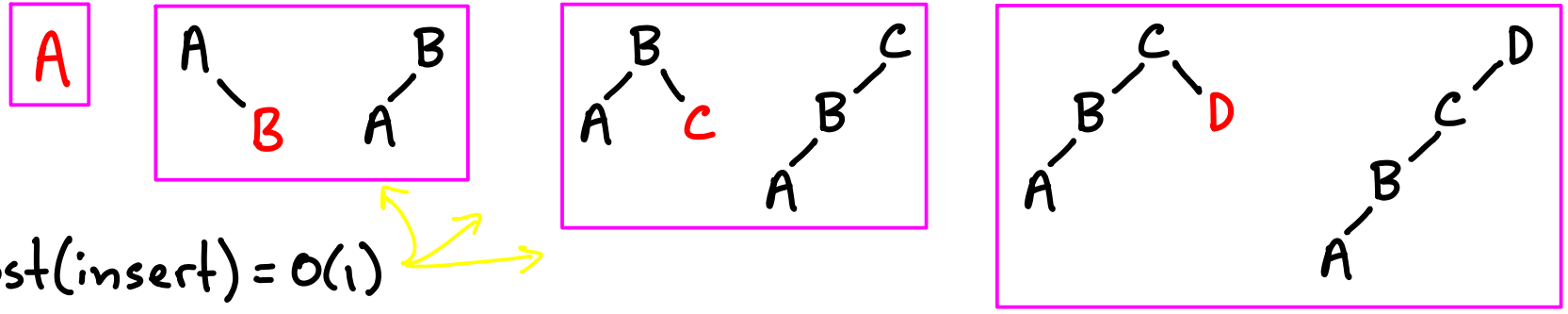
"zig-zag"



sort of has a tendency to decrease depth

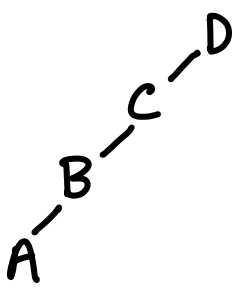
but δ could be large and/or cause greater max-depth

insert keys alphabetically

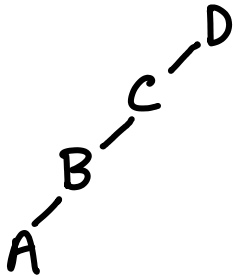


↳ depth = n but cost(insert) = O(1)

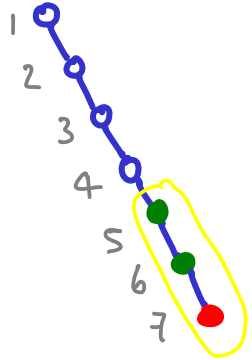
A splay tree can be awfully unbalanced but it will be cheap to create

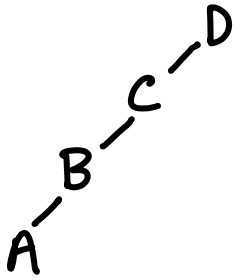


What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit

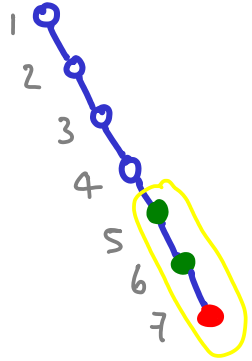


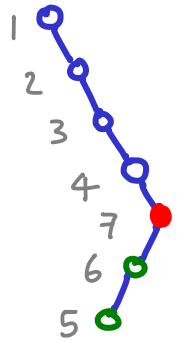
What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit

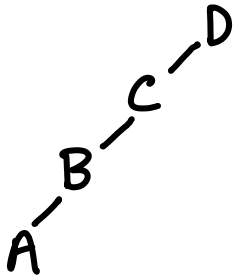




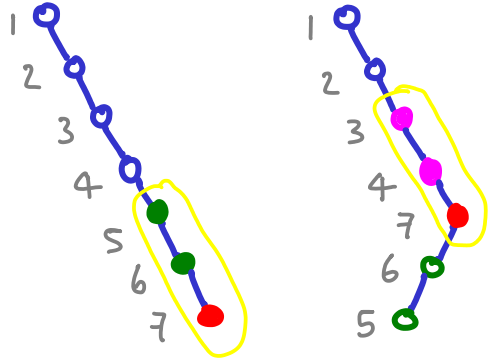
What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit

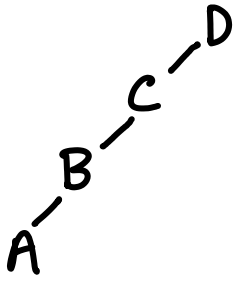




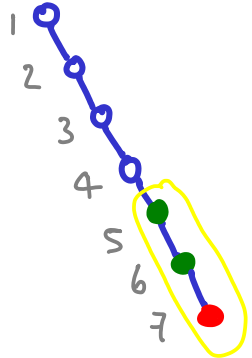


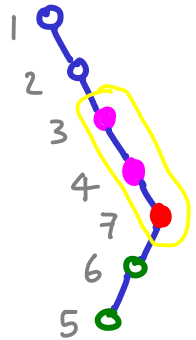
What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit

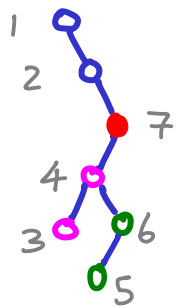


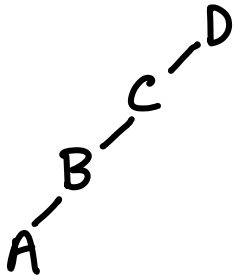


What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit

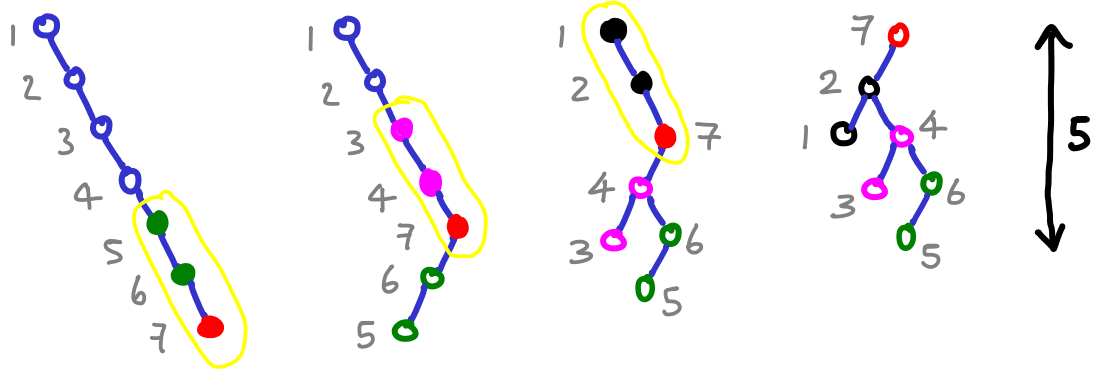


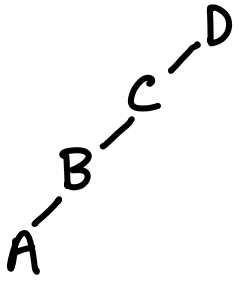




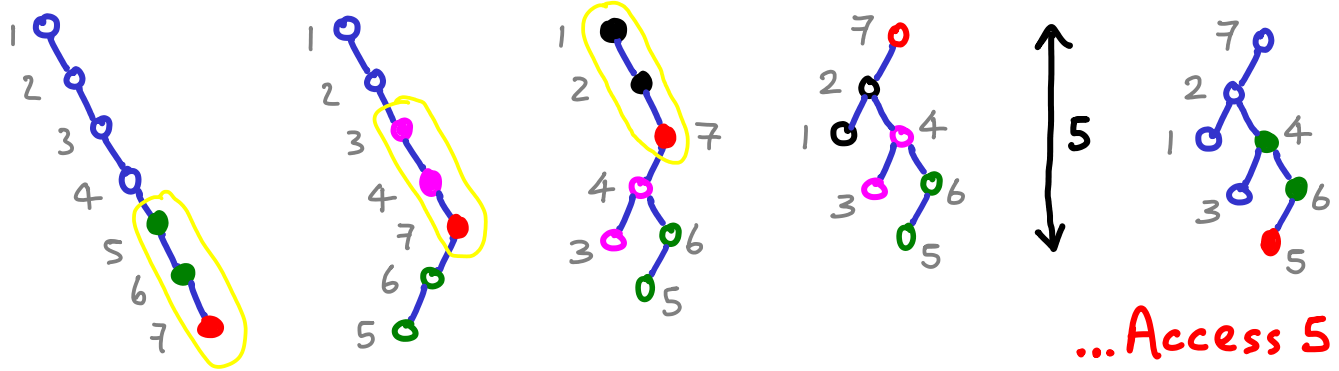


What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit

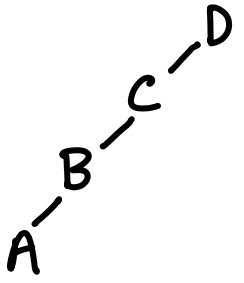




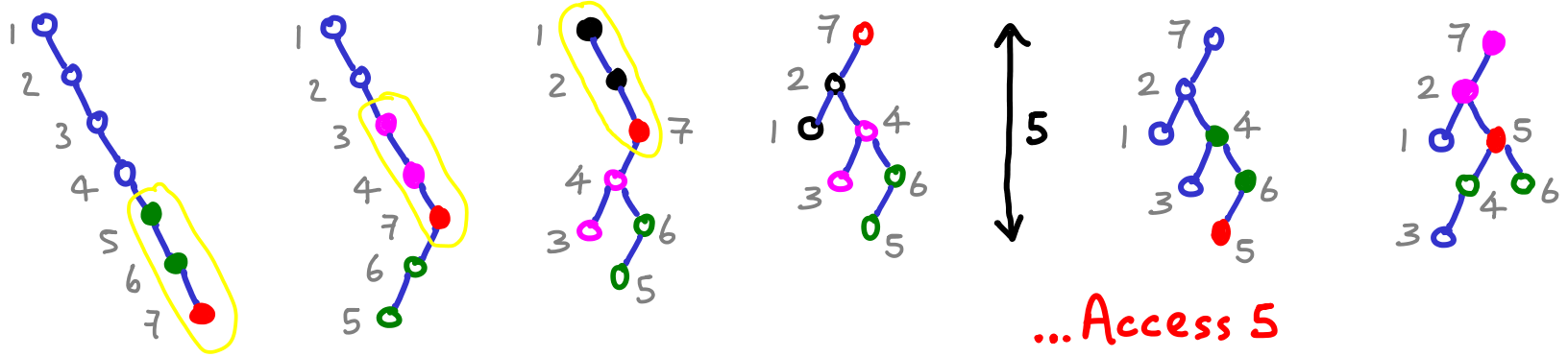
What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit

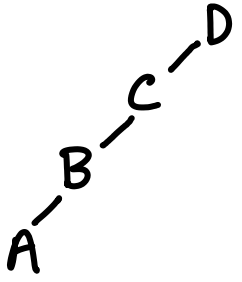


...Access 5

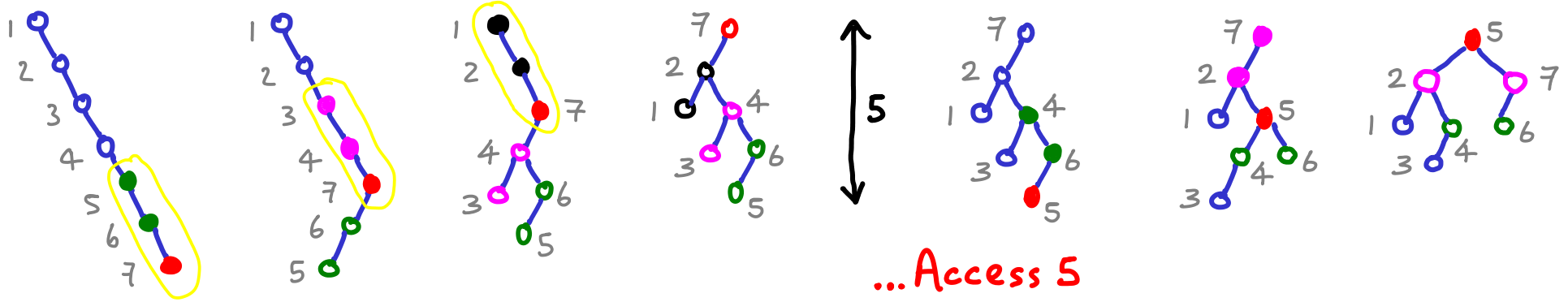


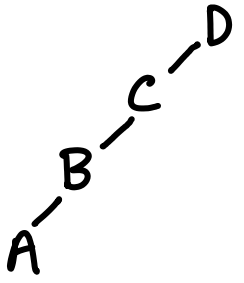
What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit



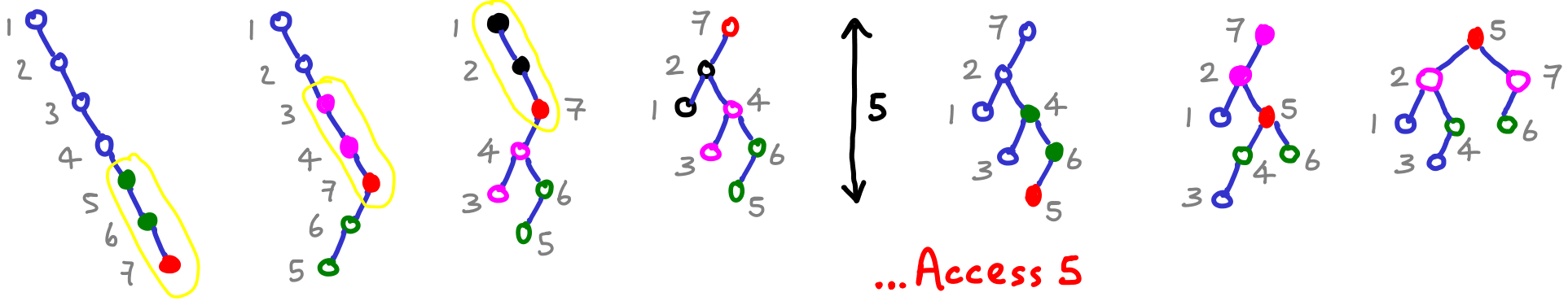


What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit

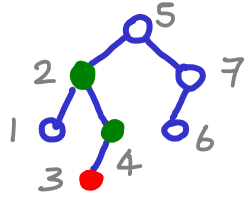




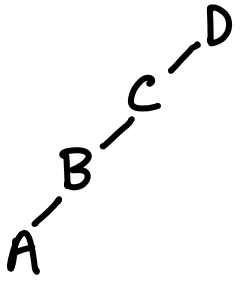
What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit



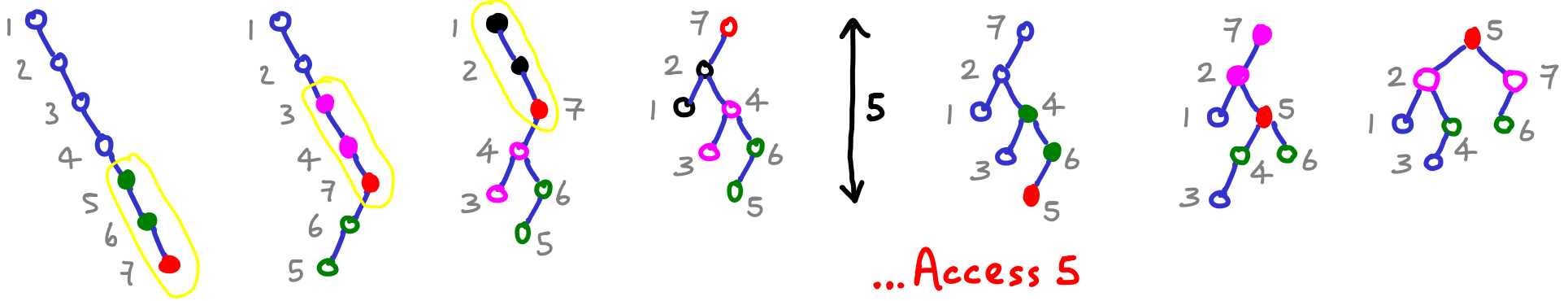
...Access 5



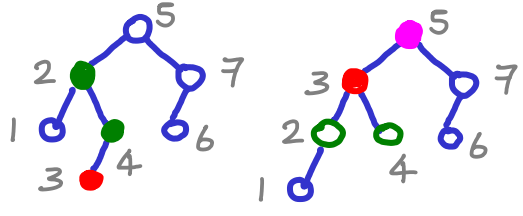
...Access 3



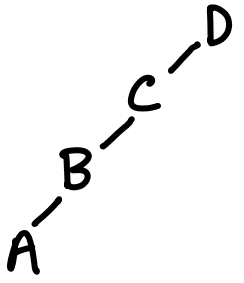
What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit



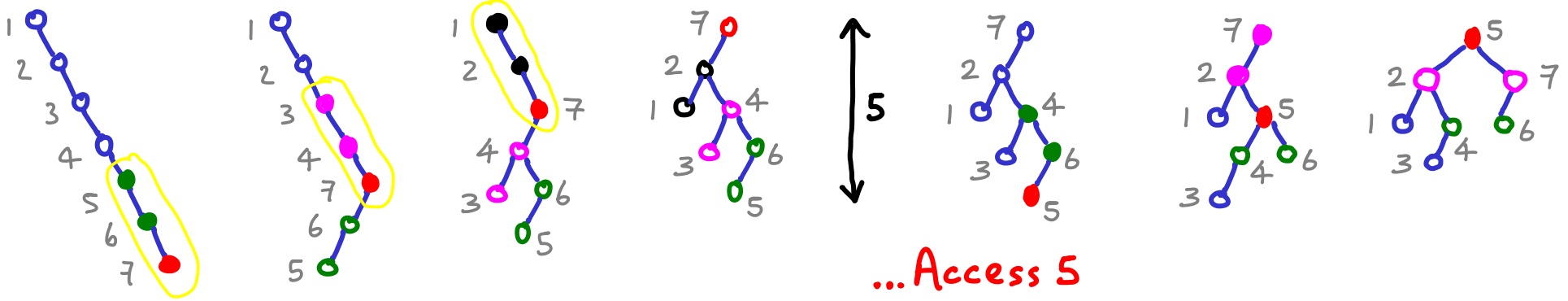
...Access 5



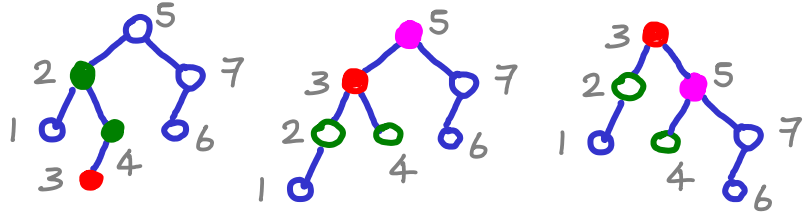
...Access 3



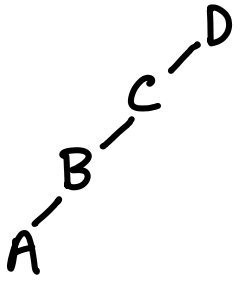
What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit



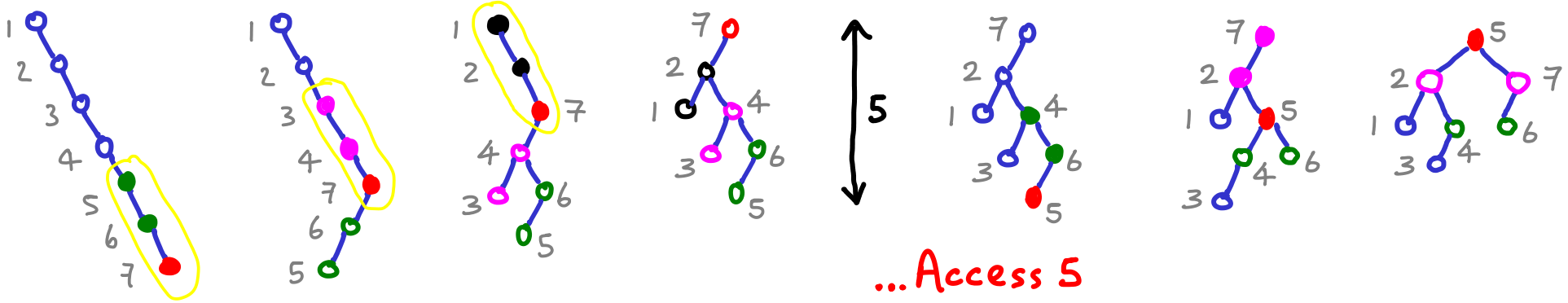
...Access 5



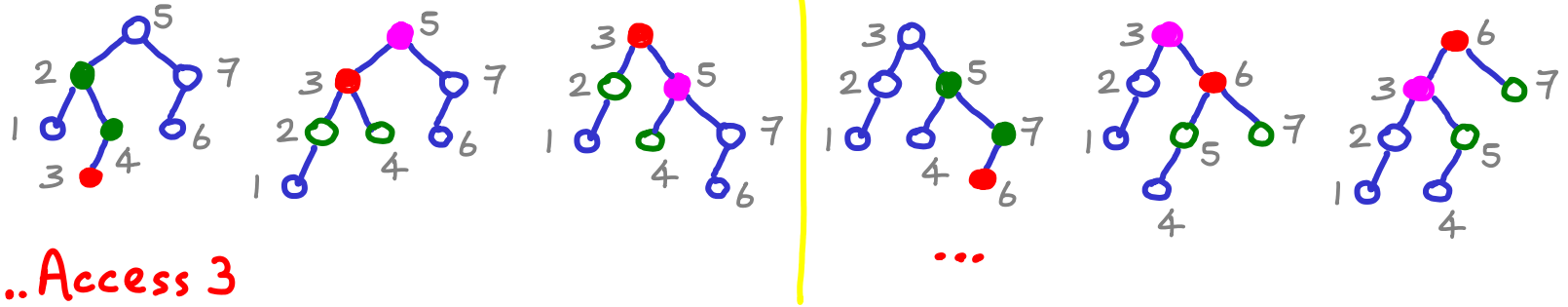
...Access 3



What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit

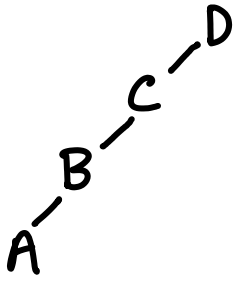


...Access 5

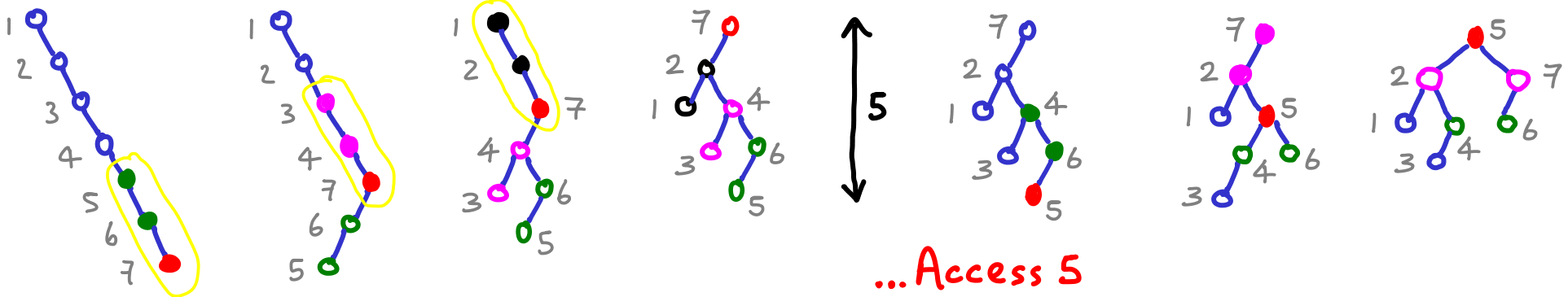


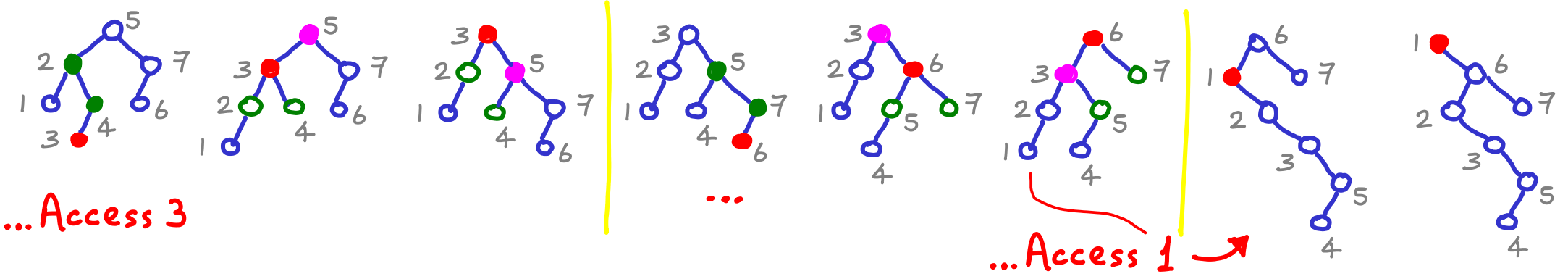
...Access 3

...



What if we access A now? Cost = n, but we will splay to top and as a result the tree will balance a bit





:?

Looks like we can create deep trees

Looks like we can create deep trees, but it takes a while to do so...

...and they get balanced quickly (at least we hope so)

Looks like we can create deep trees, but it takes a while to do so...

...and they get balanced quickly (at least we hope so)

Amortize!

Looks like we can create deep trees, but it takes a while to do so...

...and they get balanced quickly (at least we hope so)

Amortize!

#nodes in subtree rooted at v = size(v)

Looks like we can create deep trees, but it takes a while to do so...

...and they get balanced quickly (at least we hope so)

Amortize!

$s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

Looks like we can create deep trees, but it takes a while to do so...

...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

Looks like we can create deep trees, but it takes a while to do so...

...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↑ to Φ

Looks like we can create deep trees, but it takes a while to do so...

...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \ldots a_h$

Looks like we can create deep trees, but it takes a while to do so...

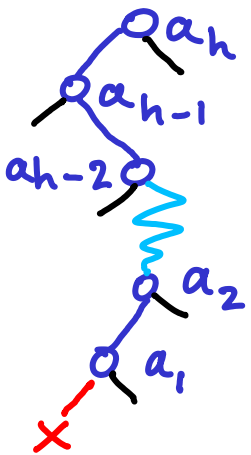
...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \ldots a_h$

$$\Delta\Phi = \sum_{i=1}^{h} \left[\log(\text{new size}(a_i)) - \log(\text{old size}(a_i)) \right]$$



Looks like we can create deep trees, but it takes a while to do so...

...and they get balanced quickly (at least we hope so)

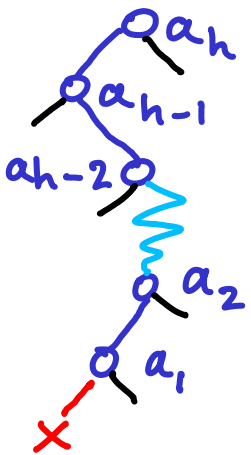
Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \ldots a_h$

$$\Delta\Phi = \sum_{i=1}^{h} \left[\log(\text{new size}(a_i)) - \log(\text{old size}(a_i)) \right] = \sum_{i=1}^{h} \left[\log(1 + \text{size}(a_i)) - \log(\text{size}(a_i)) \right]$$

old old



Looks like we can create deep trees, but it takes a while to do so...

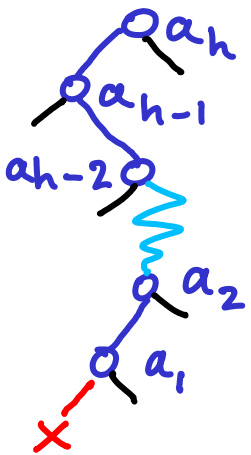
...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \ldots a_h$

$$\Delta\Phi = \sum_{i=1}^{h} \left[\log(\text{new size}(a_i)) - \log(\text{old size}(a_i))\right] = \sum_{i=1}^{h} \left[\log(1 + \text{size}(a_i)) - \log(\text{size}(a_i))\right]$$



$$= \sum_{i=1}^{h} \log \frac{1 + \text{size}(a_i)}{\text{size}(a_i)}$$

Looks like we can create deep trees, but it takes a while to do so...

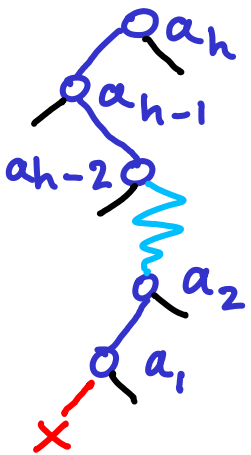
...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \ldots a_h$

$$\Delta\Phi = \sum_{i=1}^{h} \left[\log(\text{new size}(a_i)) - \log(\text{old size}(a_i))\right] = \sum_{i=1}^{h} \left[\log(1 + \text{size}(a_i)) - \log(\text{size}(a_i))\right]$$



$$= \sum_{i=1}^{h} \log \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} = \log \prod_{i=1}^{h} \frac{1 + \text{size}(a_i)}{\text{size}(a_i)}$$

Looks like we can create deep trees, but it takes a while to do so...

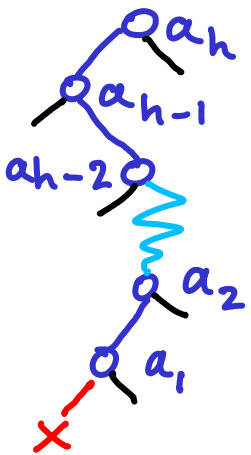
...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \ldots a_h$

$$\Delta\Phi = \sum_{i=1}^{h} \left[\log(\text{new size}(a_i)) - \log(\text{old size}(a_i)) \right] = \sum_{i=1}^{h} \left[\log(1 + \text{size}(a_i)) - \log(\text{size}(a_i)) \right]$$



$$= \sum_{i=1}^{h} \log \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} = \log \prod_{i=1}^{h} \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} = \log \left[\prod_{i=1}^{h-1} \frac{1+\text{size}(a_i)}{\text{size}(a_i)} \cdot \frac{1+\text{size}(a_h)}{\text{size}(a_h)} \right]$$

Looks like we can create deep trees, but it takes a while to do so...

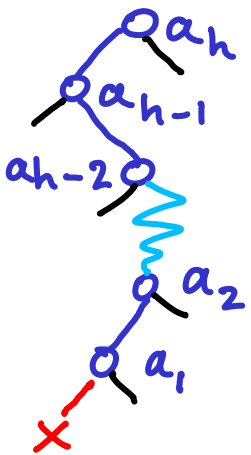
...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \ldots a_h$

$$\Delta\Phi = \sum_{i=1}^{h} \left[\log(\text{new size}(a_i)) - \log(\text{old size}(a_i))\right] = \sum_{i=1}^{h} \left[\log(1 + \text{size}(a_i)) - \log(\text{size}(a_i))\right]$$



$$= \sum_{i=1}^{h} \log \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} = \log \prod_{i=1}^{h} \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} \leq \log \left[\prod_{i=1}^{h-1} \frac{\text{size}(a_{i+1})}{\text{size}(a_i)} \cdot \frac{1 + \text{size}(a_h)}{\text{size}(a_h)} \right]$$

Looks like we can create deep trees, but it takes a while to do so...

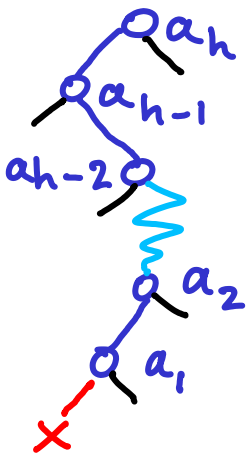
...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \ldots a_h$

$$\Delta\Phi = \sum_{i=1}^{h} \left[\log(\text{new size}(a_i)) - \log(\text{old size}(a_i))\right] = \sum_{i=1}^{h} \left[\log(1 + \text{size}(a_i)) - \log(\text{size}(a_i))\right]$$



$$= \sum_{i=1}^{h} \log \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} = \log \prod_{i=1}^{h} \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} \leq \log \left[\prod_{i=1}^{h-1} \frac{\text{size}(a_{i+1})}{\text{size}(a_i)} \cdot \frac{1 + \text{size}(a_h)}{\text{size}(a_h)} \right]$$

$$= \log \frac{1 + \text{size}(a_h)}{\text{size}(a_1)}$$

// fractions telescope

Looks like we can create deep trees, but it takes a while to do so...

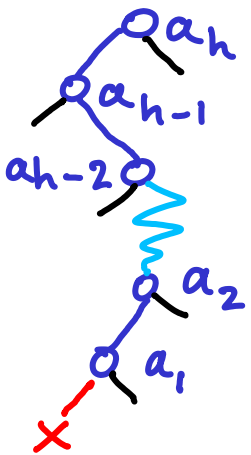
...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \ldots a_h$

$$\Delta\Phi = \sum_{i=1}^{h} \left[\log(\text{new size}(a_i)) - \log(\text{old size}(a_i))\right] = \sum_{i=1}^{h} \left[\log(1 + \text{size}(a_i)) - \log(\text{size}(a_i))\right]$$



$$= \sum_{i=1}^{h} \log \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} = \log \prod_{i=1}^{h} \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} \leq \log \left[\prod_{i=1}^{h-1} \frac{\text{size}(a_{i+1})}{\text{size}(a_i)} \cdot \frac{1 + \text{size}(a_h)}{\text{size}(a_h)} \right]$$

$$= \log \frac{1 + \text{size}(a_h)}{\text{size}(a_1)} \leq \log\left[1 + \text{size}(a_h)\right]$$

Looks like we can create deep trees, but it takes a while to do so...

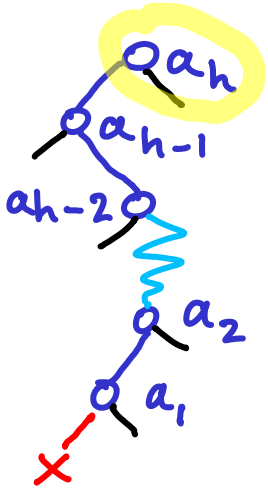
...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \ldots a_h$

$$\Delta\Phi = \sum_{i=1}^{h} \left[\log(\text{new size}(a_i)) - \log(\text{old size}(a_i))\right] = \sum_{i=1}^{h} \left[\log(1 + \text{size}(a_i)) - \log(\text{size}(a_i))\right]$$



$$= \sum_{i=1}^{h} \log \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} = \log \prod_{i=1}^{h} \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} \leq \log \left[\prod_{i=1}^{h-1} \frac{\text{size}(a_{i+1})}{\text{size}(a_i)} \cdot \frac{1 + \text{size}(a_h)}{\text{size}(a_h)} \right]$$

$$= \log \frac{1 + \text{size}(a_h)}{\text{size}(a_1)} \leq \log\left[1 + \text{size}(a_h)\right] = \log\left[1 + \text{size}(\text{root})\right]$$

Looks like we can create deep trees, but it takes a while to do so...

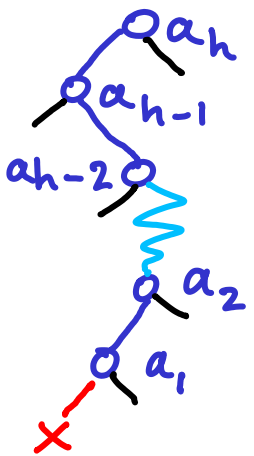
...and they get balanced quickly (at least we hope so)

Amortize! $\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ?

↳ s( ) changes only for h nodes on path: each eventual ancestor $a_1 \dots a_h$

$$\Delta\Phi = \sum_{i=1}^{h} \left[\log(\text{new size}(a_i)) - \log(\text{old size}(a_i))\right] = \sum_{i=1}^{h} \left[\log(1 + \text{size}(a_i)) - \log(\text{size}(a_i))\right]$$



$$= \sum_{i=1}^{h} \log \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} = \log \prod_{i=1}^{h} \frac{1 + \text{size}(a_i)}{\text{size}(a_i)} \leq \log \left[\prod_{i=1}^{h-1} \frac{\text{size}(a_{i+1})}{\text{size}(a_i)} \cdot \frac{1 + \text{size}(a_h)}{\text{size}(a_h)} \right]$$

$$= \log \frac{1 + \text{size}(a_h)}{\text{size}(a_1)} \leq \log\left[1 + \text{size}(a_h)\right] = \log\left[1 + \text{size}(\text{root})\right] = \log n$$

(old)

$\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ? $\quad \Delta\Phi \leq \log n$

$\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ? $\quad \Delta\Phi \leq \log n$

delete x ?

$\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{ nodes in subtree rooted at } v) = \log \text{size}(v)$

What happens when insert x ? $\quad \Delta\Phi \leq \log n$

delete x ? $\quad \leq 0$ good news

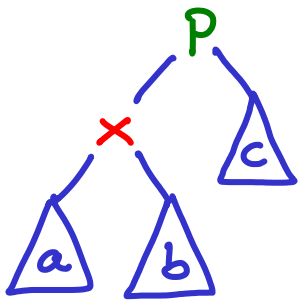
We must still consider $\Delta\Phi$ because of splaying

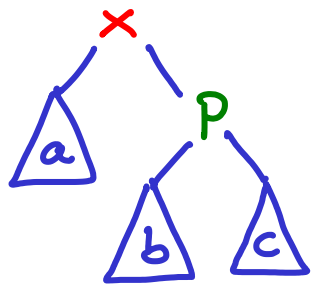
$\Phi = \sum_{\text{all } v} s(v)$

$s(v) = \log_2(\#\text{nodes in subtree rooted at } v)$

What happens when we splay x and there is no grandparent?

↳ s( ) changes only for x and p





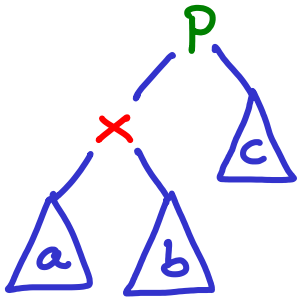
$\Phi = \sum_{\text{all } v} s(v)$

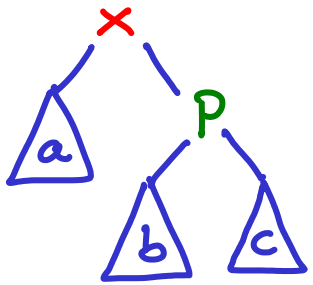
$s(v) = \log_2(\#\text{nodes in subtree rooted at } v)$

What happens when we splay x and there is no grandparent?

↳ s( ) changes only for x and p

$\Delta\Phi = \Delta s(p) + \Delta s(x)$





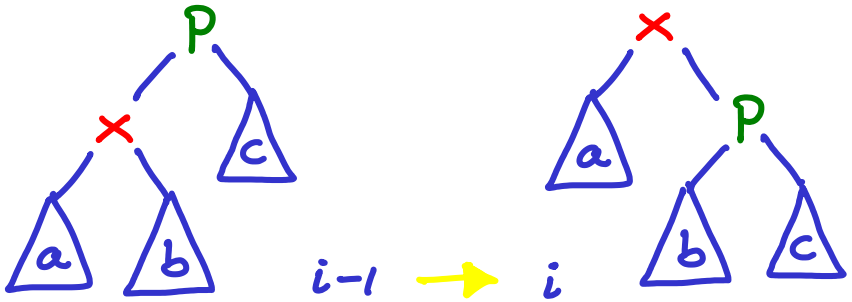
$\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v)$

What happens when we splay x and there is no grandparent?

↳ s( ) changes only for x and p

$$\Delta\Phi = \Delta s(p) + \Delta s(x)$$

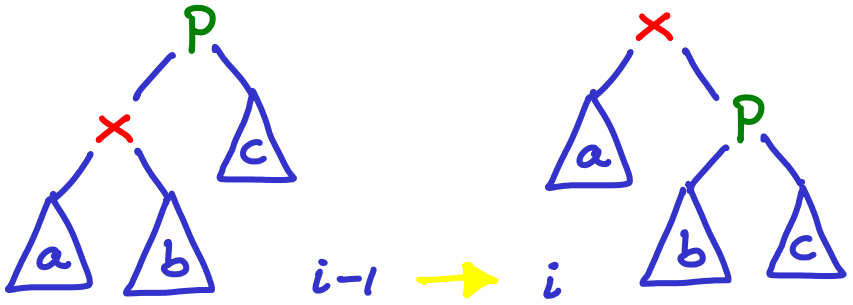
$$= s_i(p) - s_{i-1}(p) + s_i(x) - s_{i-1}(x)$$



$\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v)$

What happens when we splay x and there is no grandparent?

↳ s( ) changes only for x and p



$$\Delta\Phi = \Delta s(p) + \Delta s(x)$$

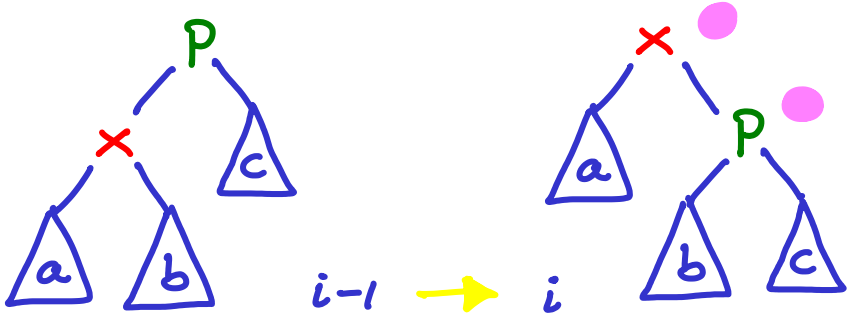
$$= s_i(p) - s_{i-1}(p) + s_i(x) - s_{i-1}(x)$$

$$= s_i(p) - s_{i-1}(x)$$

$\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v)$

What happens when we splay x and there is no grandparent?

↳ s( ) changes only for x and p



$$\Delta\Phi = \Delta s(p) + \Delta s(x)$$

$$= s_i(p) - s_{i-1}(p) + s_i(x) - s_{i-1}(x)$$

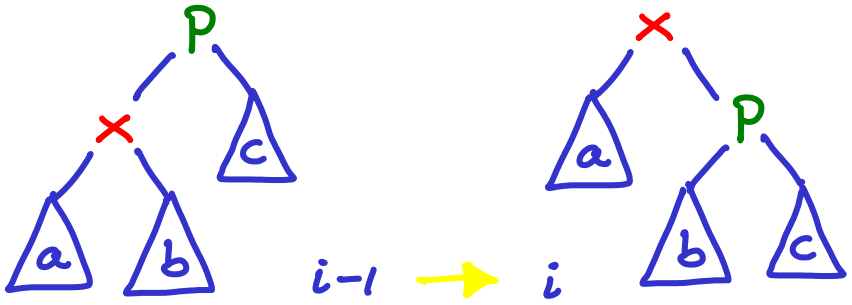
$$= s_i(p) - s_{i-1}(x) \quad // \quad s_i(p) < s_i(x)$$

$$< s_i(x) - s_{i-1}(x)$$

$\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v)$

What happens when we splay x and there is no grandparent?

↳ s( ) changes only for x and p



$$\Delta\Phi = \Delta s(p) + \Delta s(x)$$

$$= s_i(p) - s_{i-1}(p) + s_i(x) - s_{i-1}(x)$$

$$= s_i(p) - s_{i-1}(x) \quad // \; s_i(p) < s_i(x)$$

$$< s_i(x) - s_{i-1}(x)$$

always positive

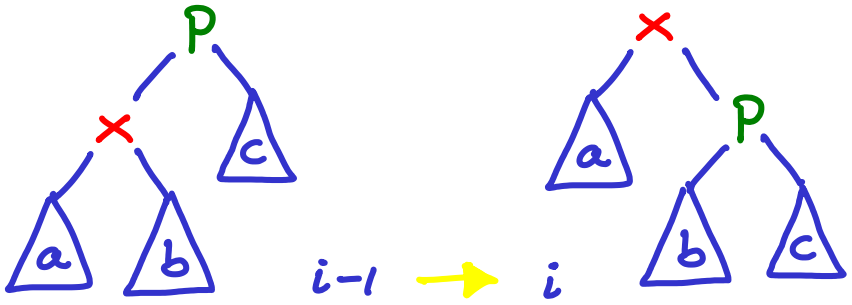
$\Delta\Phi$ could actually be $=0$, <0 or >0

Why are we happy with this?

$\Phi = \sum_{\text{all } v} s(v)$ $\quad s(v) = \log_2(\#\text{nodes in subtree rooted at } v)$

What happens when we splay x and there is no grandparent?

↳ s( ) changes only for x and p



$\Delta\Phi = \Delta s(p) + \Delta s(x)$

$= s_i(p) - s_{i-1}(p) + s_i(x) - s_{i-1}(x)$

$= s_i(p) - s_{i-1}(x)$ // $s_i(p) < s_i(x)$

$< s_i(x) - s_{i-1}(x)$

always positive

We have $\Delta\Phi < s_i(x) = s(\text{root})$

$= O(\log n)$

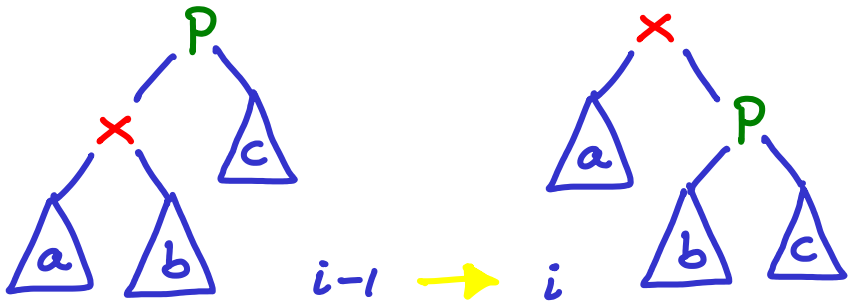
$\Delta\Phi$ could actually be $=0$, <0 or >0

Why are we happy with this?

$\Phi = \sum_{\text{all } v} s(v)$ $\qquad s(v) = \log_2(\#\text{nodes in subtree rooted at } v)$

What happens when we splay x and there is no grandparent?

↳ s( ) changes only for x and p



$$\Delta\Phi = \Delta s(p) + \Delta s(x)$$

$$= s_i(p) - s_{i-1}(p) + s_i(x) - s_{i-1}(x)$$

$$= s_i(p) - s_{i-1}(x) \quad // \; s_i(p) < s_i(x)$$

$$< s_i(x) - s_{i-1}(x)$$

always positive

We have $\Delta\Phi < s_i(x) = s(\text{root})$

$= O(\log n)$

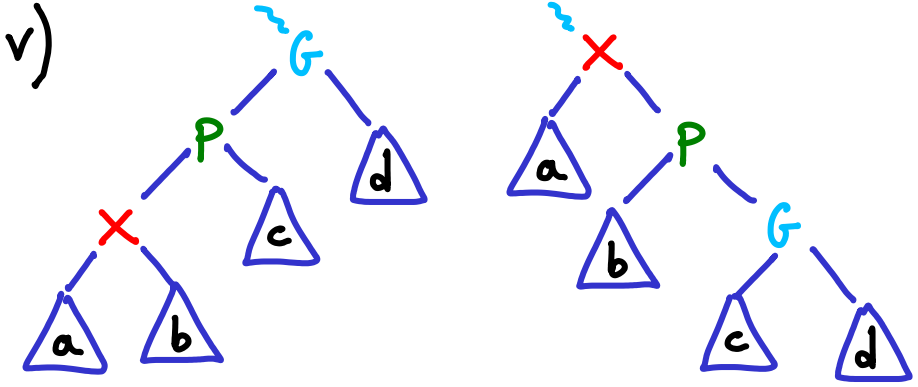
We would be happy even if this iterated (telescope)

$\Delta\Phi$ could actually be $=0$, <0 or >0

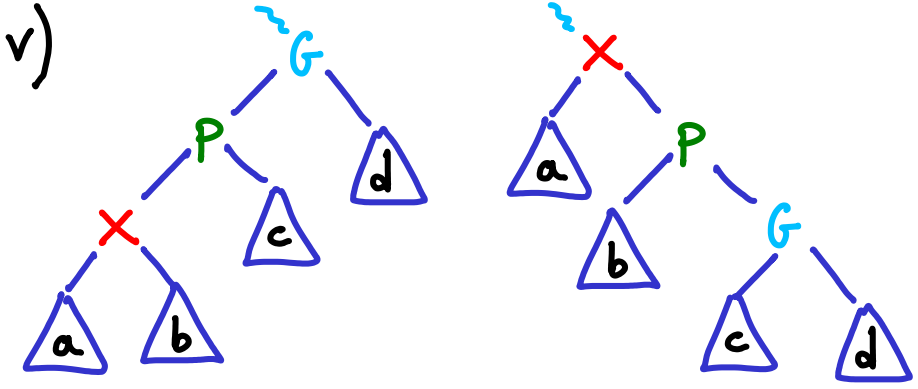
Why are we happy with this?

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

What happens when we zig-zig x?



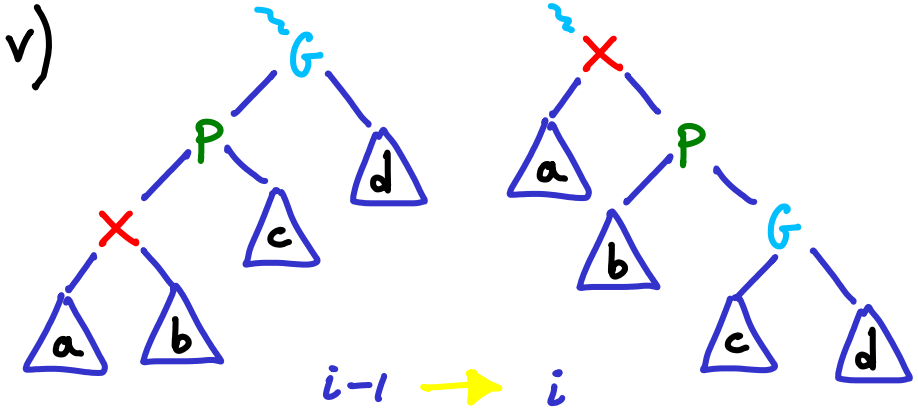
$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$



What happens when we zig-zig x?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

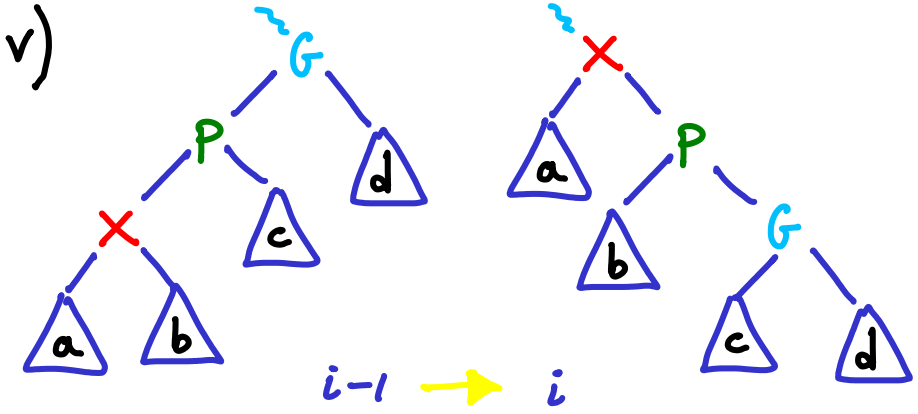


What happens when we zig-zig x ?

$$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$$

$$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$



What happens when we zig-zig x?

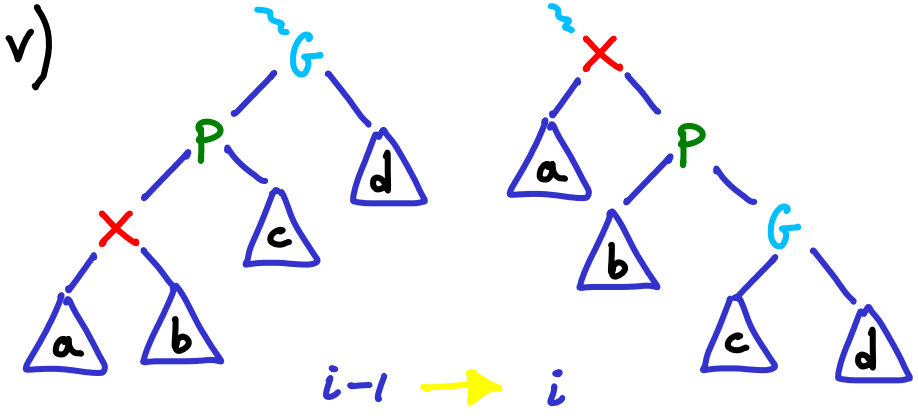
$$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$$

$$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$$

$$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

What happens when we zig-zig x?



$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(X)$

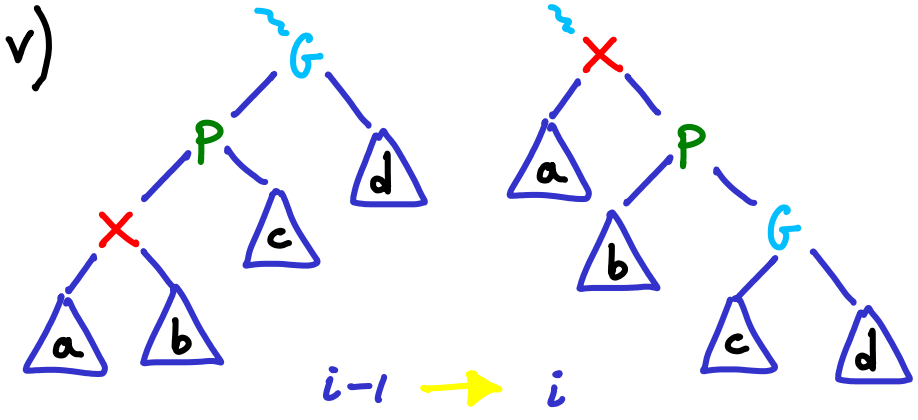
$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(X) - s_{i-1}(X)$

$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(X)$

$s_i(P) < s_i(X)$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

What happens when we zig-zig x?



$$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$$

$$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$$

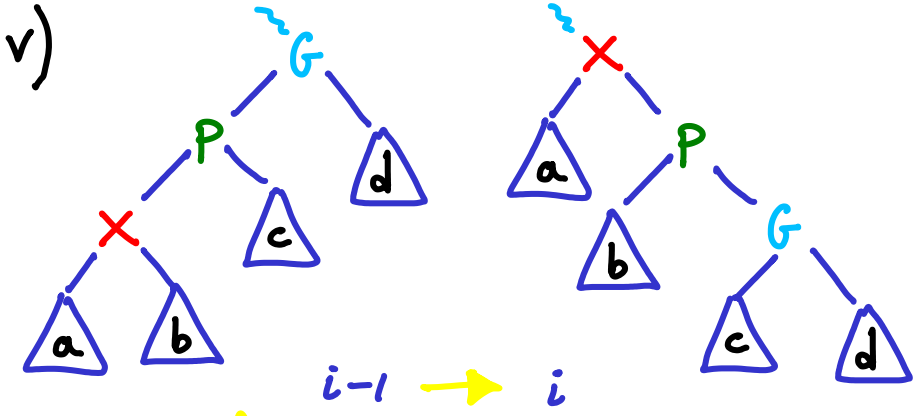
$$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$$

$$< s_i(G) + s_i(x) - s_{i-1}(P) - s_{i-1}(x)$$

$s_i(P) < s_i(x)$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

What happens when we zig-zig x?



$$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$$

$$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$$

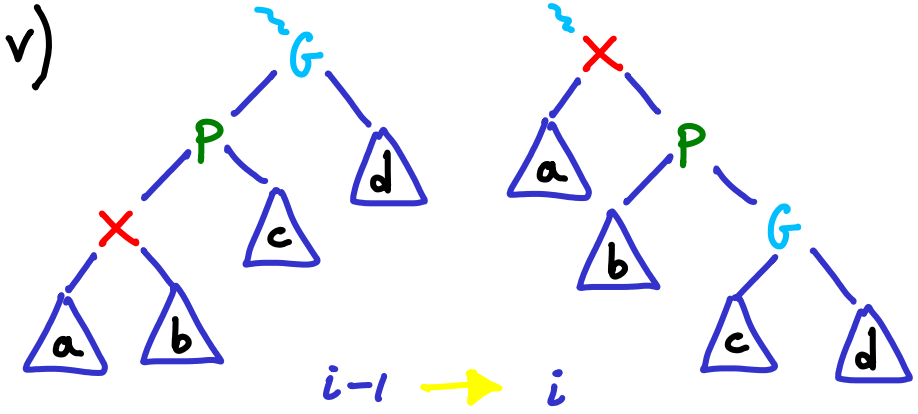
$$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$$

$$< s_i(G) + s_i(x) - s_{i-1}(P) - s_{i-1}(x)$$

$s_i(P) < s_i(x)$

$s_{i-1}(P) > s_{i-1}(x)$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$



What happens when we zig-zig x?

$$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(X)$$

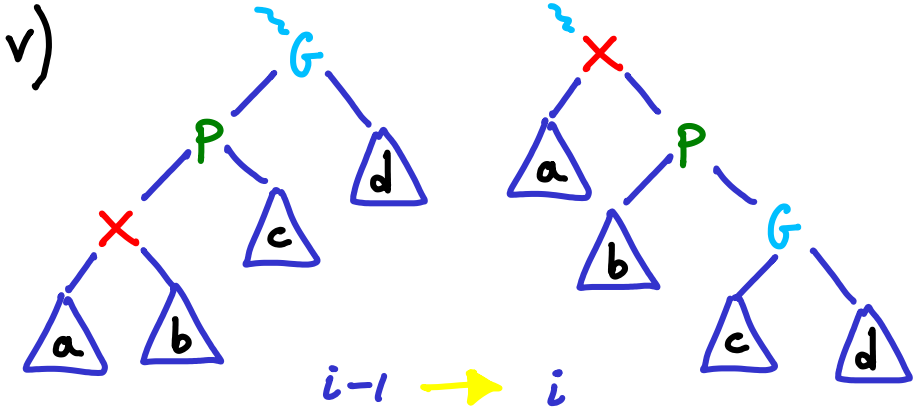
$$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(X) - s_{i-1}(X)$$

$$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(X)$$

$$< s_i(G) + s_i(X) - s_{i-1}(P) - s_{i-1}(X) \qquad s_i(P) < s_i(X)$$

$$< s_i(G) + s_i(X) - s_{i-1}(X) - s_{i-1}(X) \qquad s_{i-1}(P) > s_{i-1}(X)$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$



What happens when we zig-zig x?

$$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(X)$$

$$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(X) - s_{i-1}(X)$$

$$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(X)$$

$$< s_i(G) + s_i(X) - s_{i-1}(P) - s_{i-1}(X) \quad (s_i(P) < s_i(X))$$

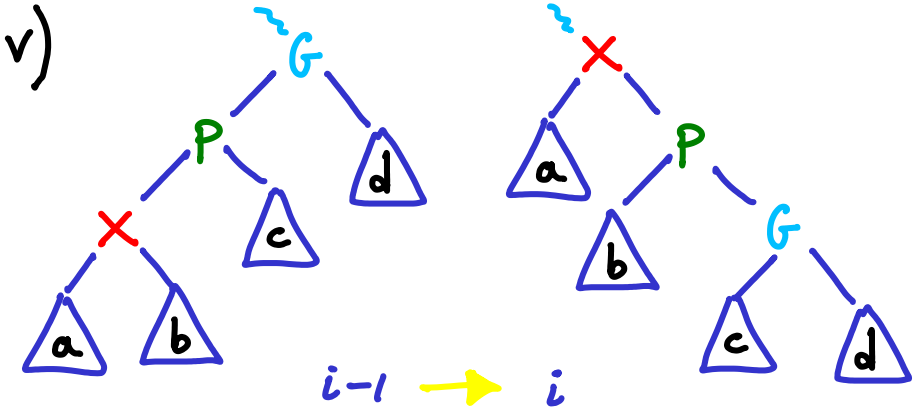
$$< s_i(G) + s_i(X) - s_{i-1}(X) - s_{i-1}(X) \quad (s_{i-1}(P) > s_{i-1}(X))$$

$$< s_i(G) + s_i(X) - 2s_{i-1}(X)$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

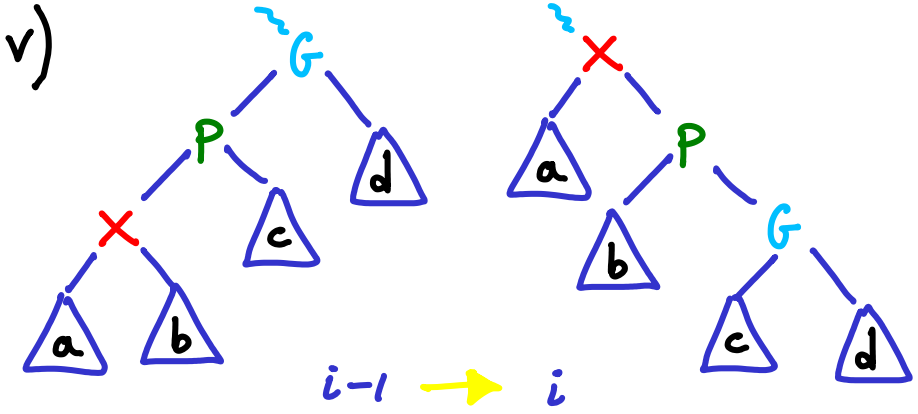
What happens when we zig-zig x?

$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$



$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

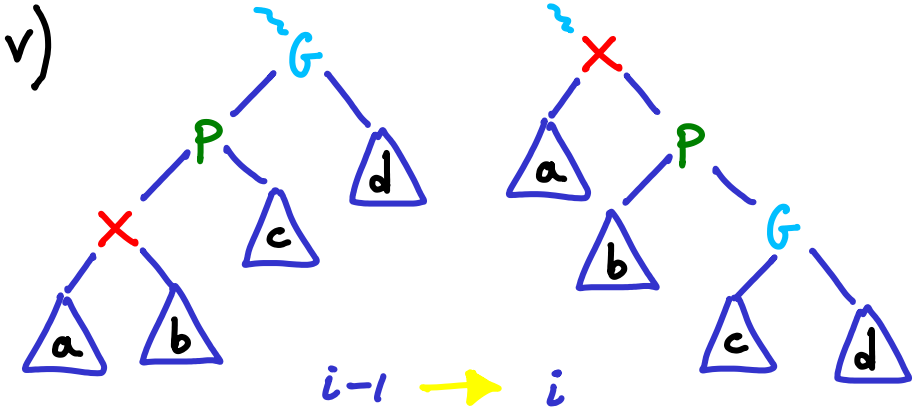
What happens when we zig-zig x?



$$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$$

$$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$



What happens when we zig-zig x?

$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$

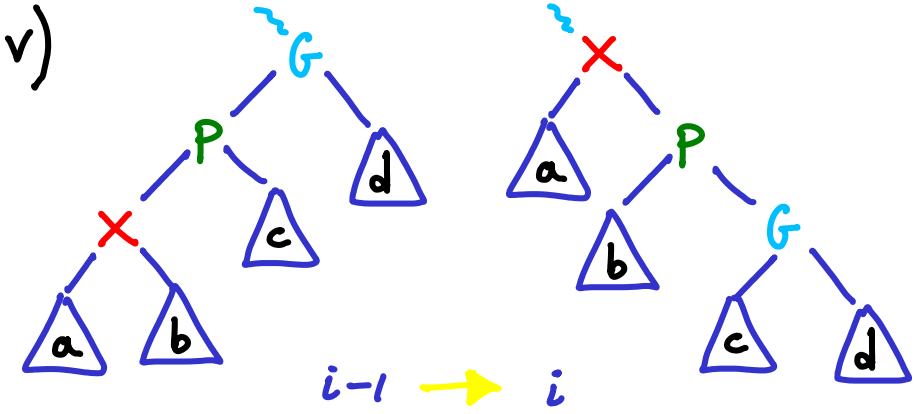
$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$

$s_i(G) + s_{i-1}(x) =$ ...

we will show: $s_i(G) + s_{i-1}(x) < 2s_i(x) - 2$

can then add to this

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$



What happens when we zig-zig x?

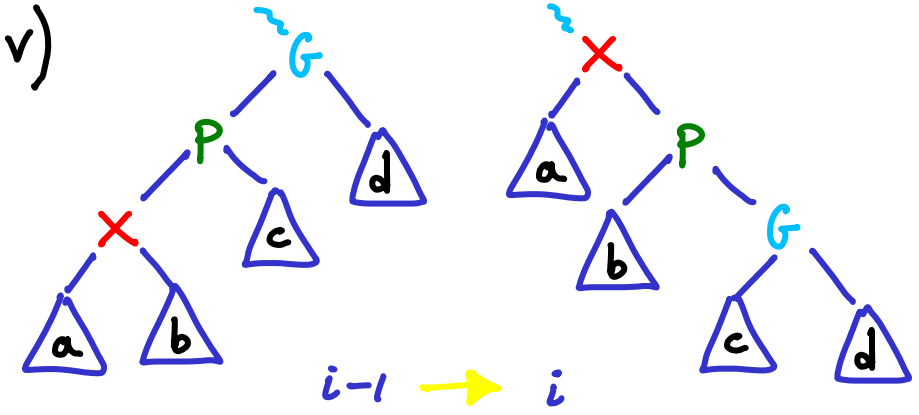
$$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$$

$$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$$

$$s_i(G) + s_{i-1}(x) = 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_{i-1}(x))}{2}$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

What happens when we zig-zig x?

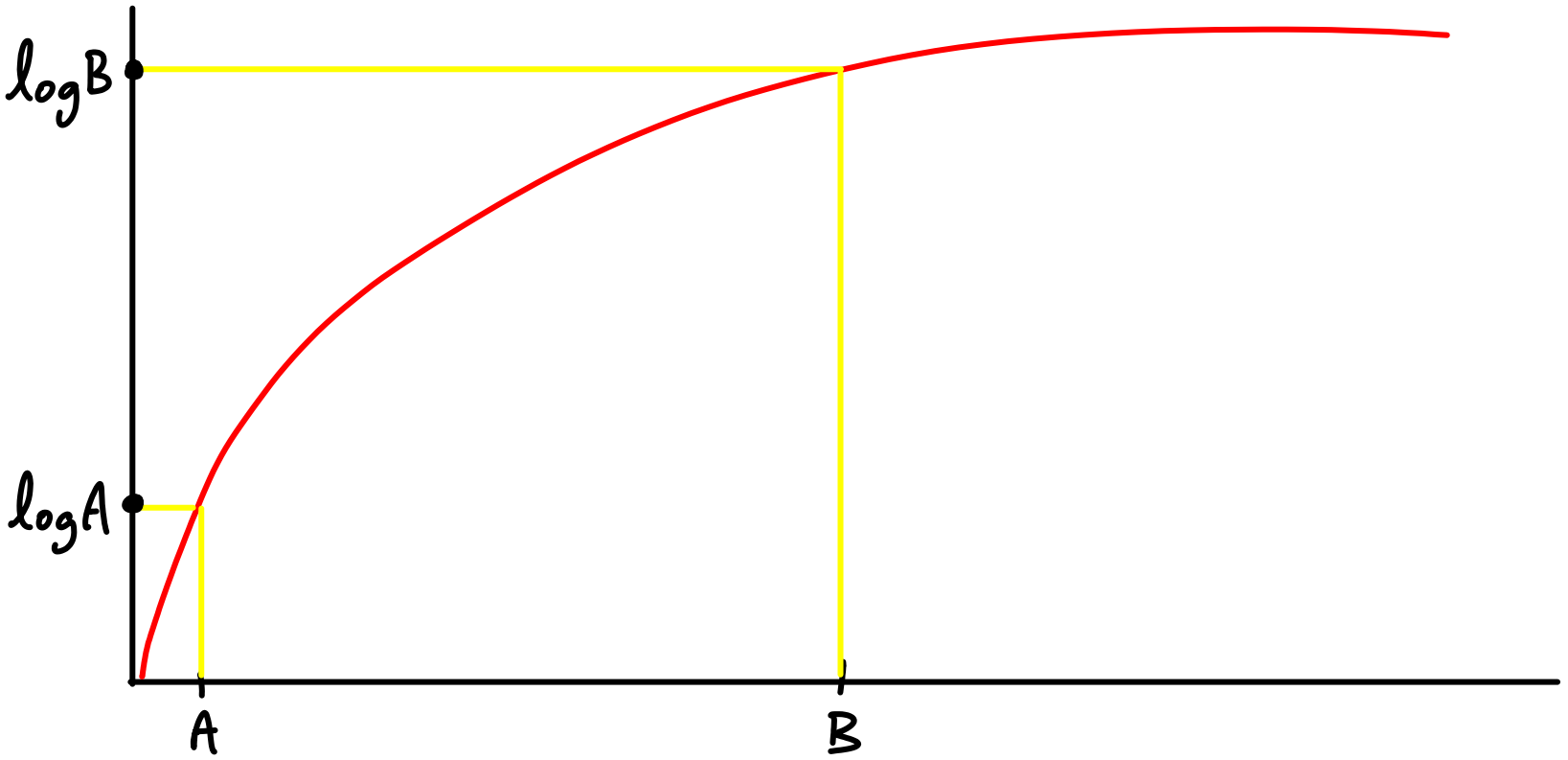


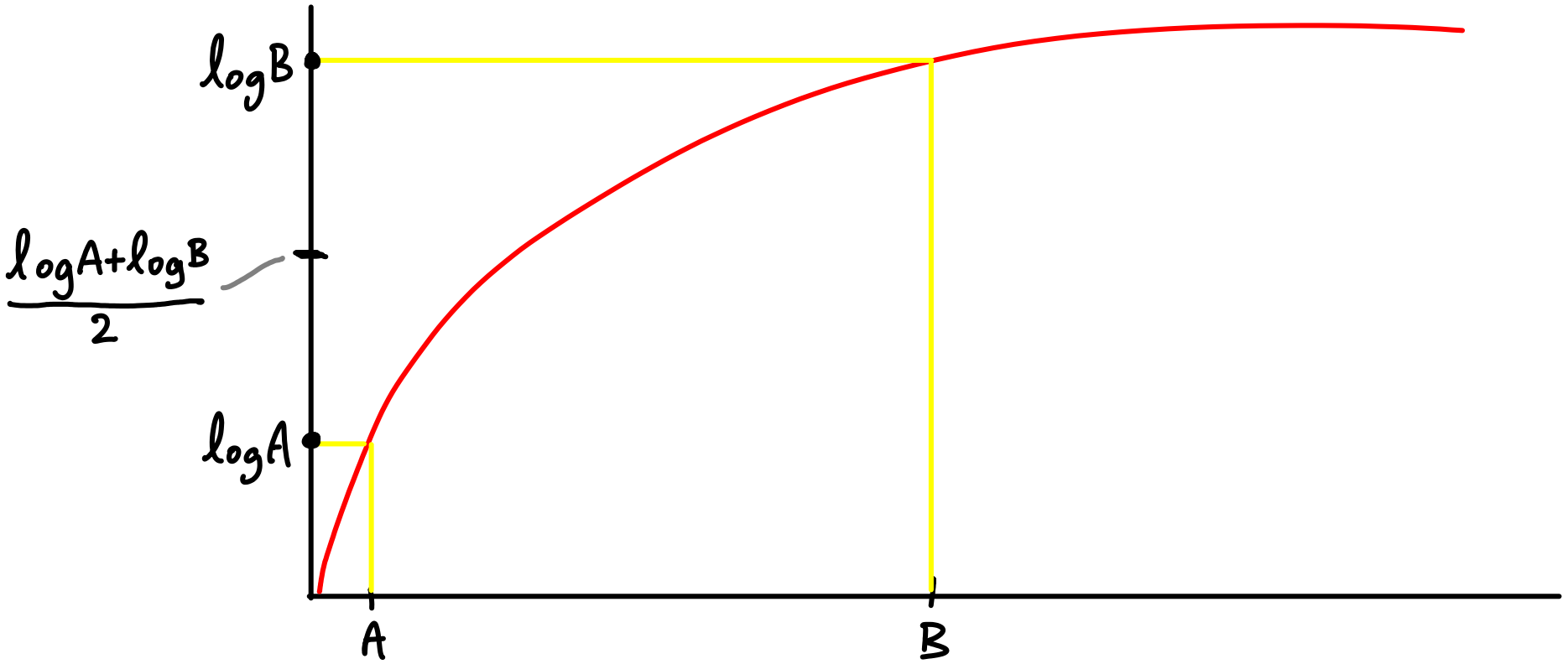
$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$

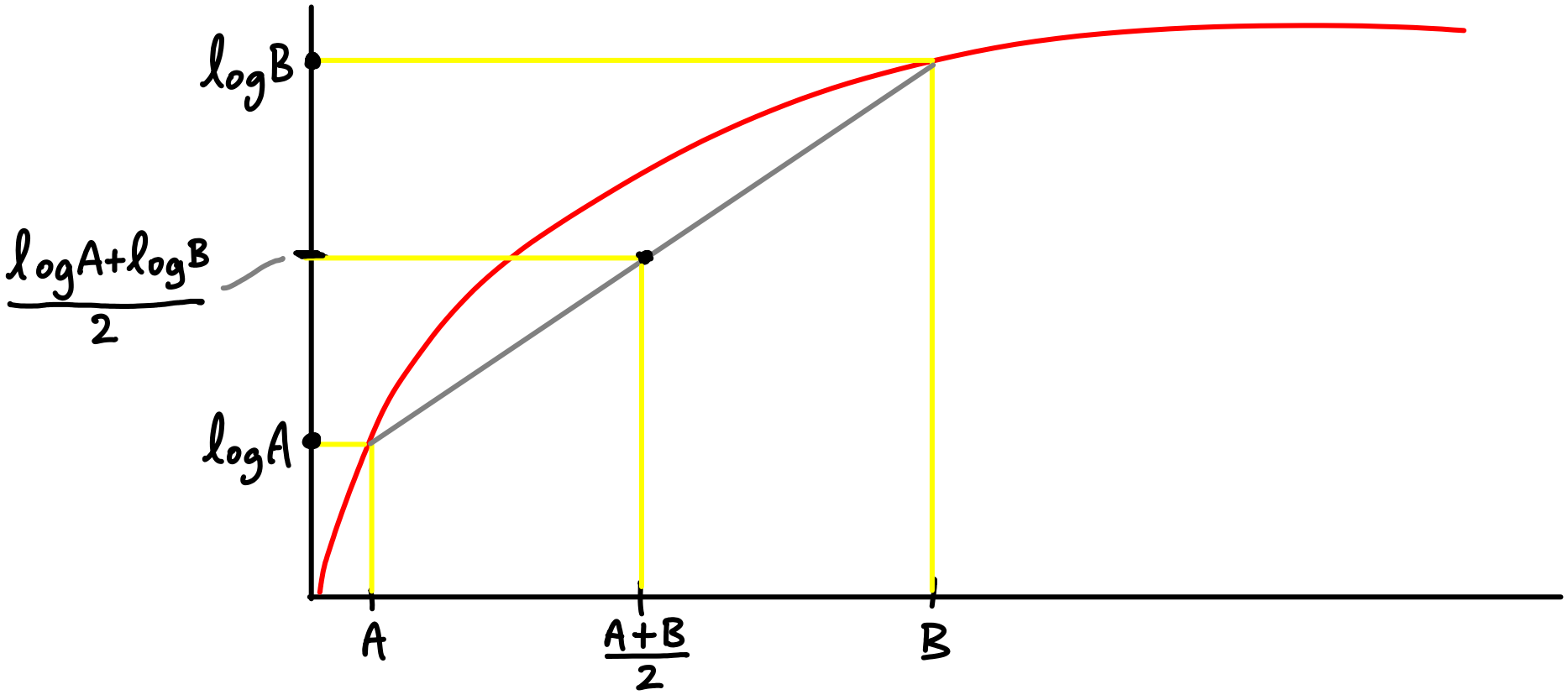
$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$

$\log A + \log B$ / 2

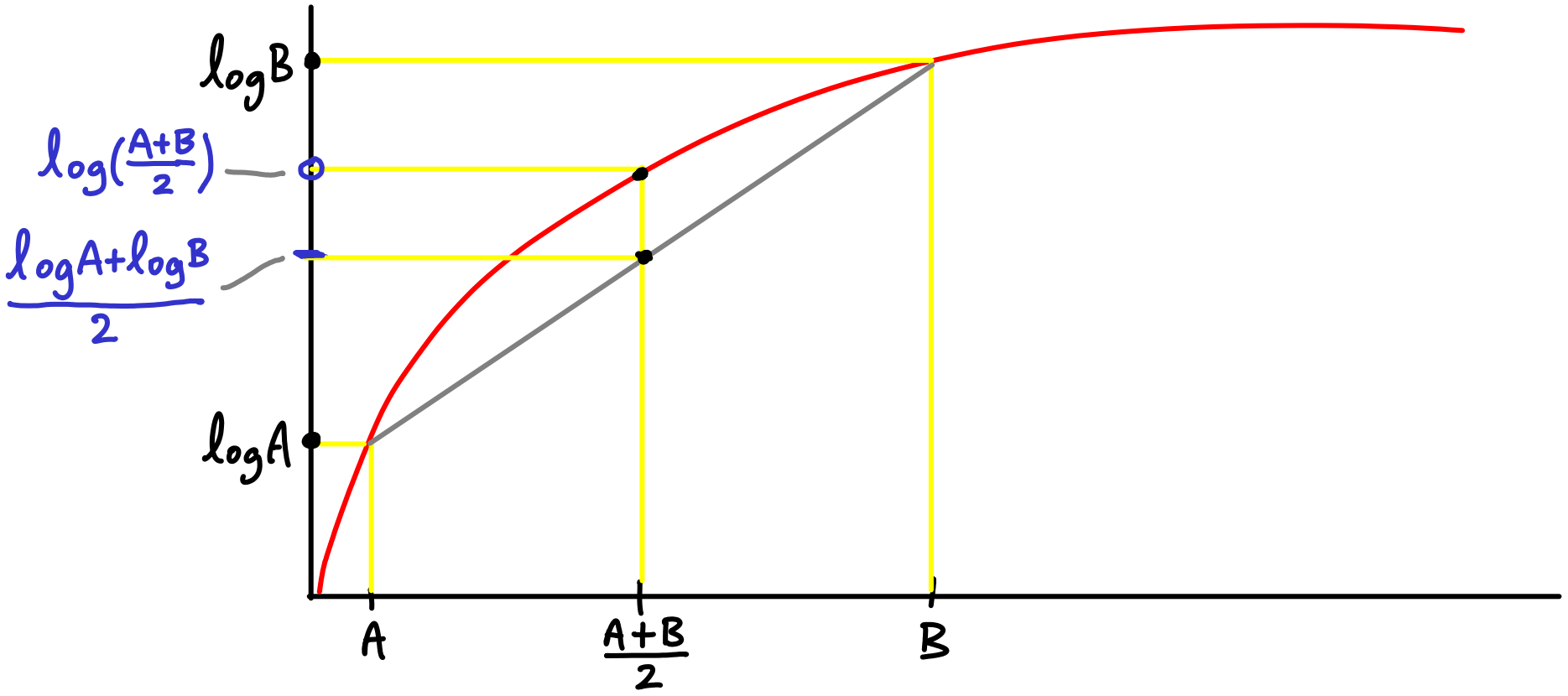
$$s_i(G) + s_{i-1}(x) = 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_{i-1}(x))}{2}$$


logB
logA
A
B


$\log B$
$\frac{\log A+\log B}{2}$
$\log A$
A
B

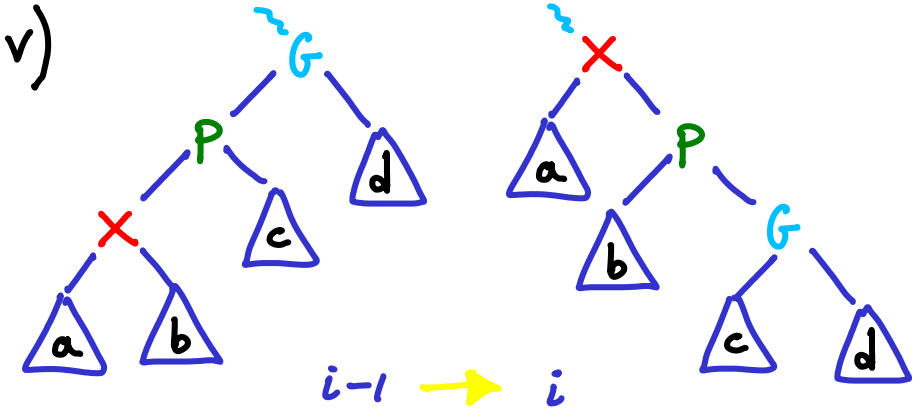

$\log B$
$\frac{\log A+\log B}{2}$
$\log A$
A
$\frac{A+B}{2}$
B

$$\frac{\log A + \log B}{2} \leq \log\left(\frac{A+B}{2}\right)$$



$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

What happens when we zig-zig x?



$$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$$

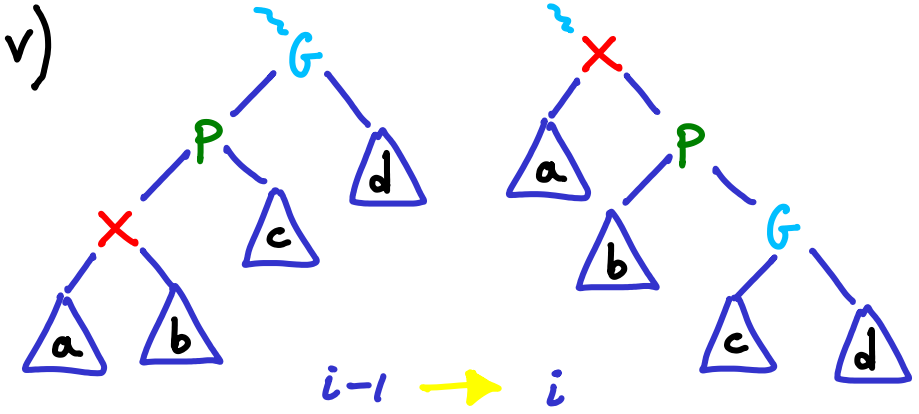
$$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$$

$$\frac{\log A + \log B}{2} \leq \log\left(\frac{A+B}{2}\right)$$

$$s_i(G) + s_{i-1}(x) = 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_{i-1}(x))}{2}$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

What happens when we zig-zig x?



$$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$$

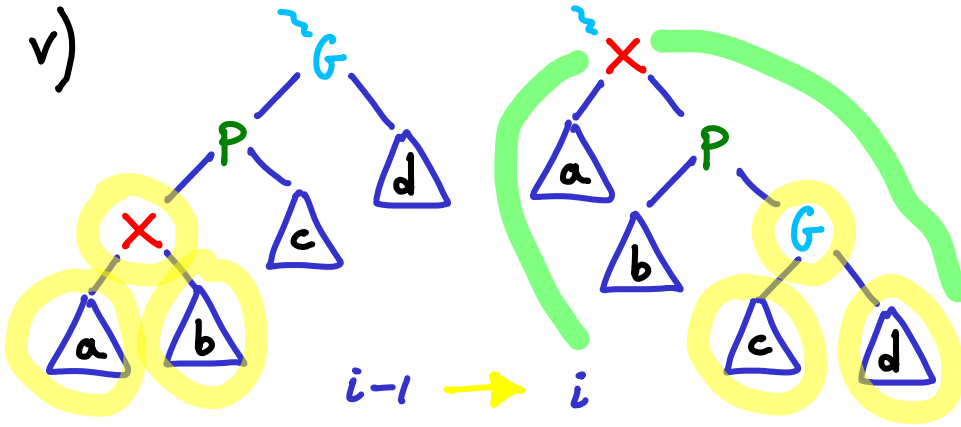
$$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$$

$$\frac{\log A + \log B}{2} \leq \log\left(\frac{A+B}{2}\right)$$

$$s_i(G) + s_{i-1}(x) = 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_{i-1}(x))}{2} \leq 2 \cdot \log\left(\frac{\text{size}_i(G) + \text{size}_{i-1}(x)}{2}\right)$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

What happens when we zig-zig x?



$$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$$

$$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$$

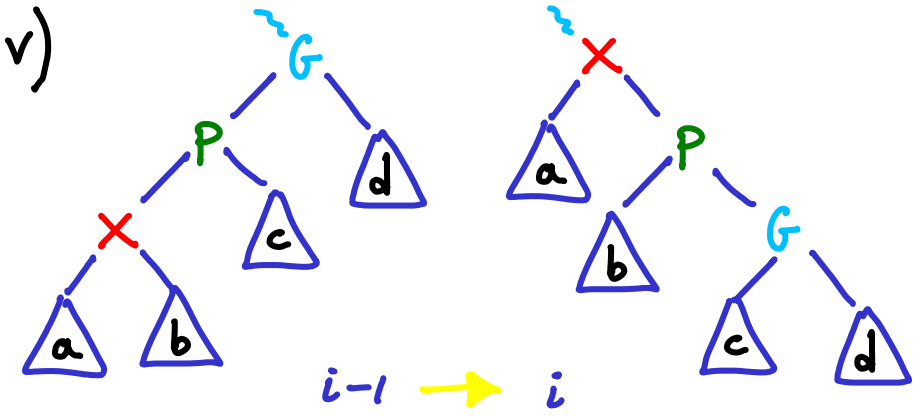
$$\frac{\log A + \log B}{2} \leq \log\left(\frac{A+B}{2}\right)$$

$$s_i(G) + s_{i-1}(x) = 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_{i-1}(x))}{2} \leq 2 \cdot \log\left(\frac{\text{size}_i(G) + \text{size}_{i-1}(x)}{2}\right)$$

$$< 2 \cdot \log\left(\frac{\text{size}_i(x)}{2}\right)$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

What happens when we zig-zig x?



$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$

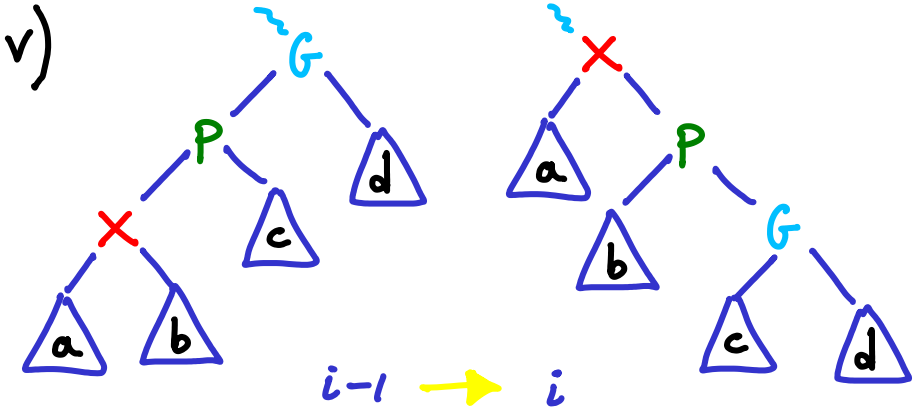
$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$

$\frac{\log A + \log B}{2} \leq \log\left(\frac{A+B}{2}\right)$

$s_i(G) + s_{i-1}(x) = 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_{i-1}(x))}{2} \leq 2 \cdot \log\left(\frac{\text{size}_i(G) + \text{size}_{i-1}(x)}{2}\right)$

$< 2 \cdot \log\left(\frac{\text{size}_i(x)}{2}\right) = 2 \cdot [s_i(x) - 1]$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$



What happens when we zig-zig x?

$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$

$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$

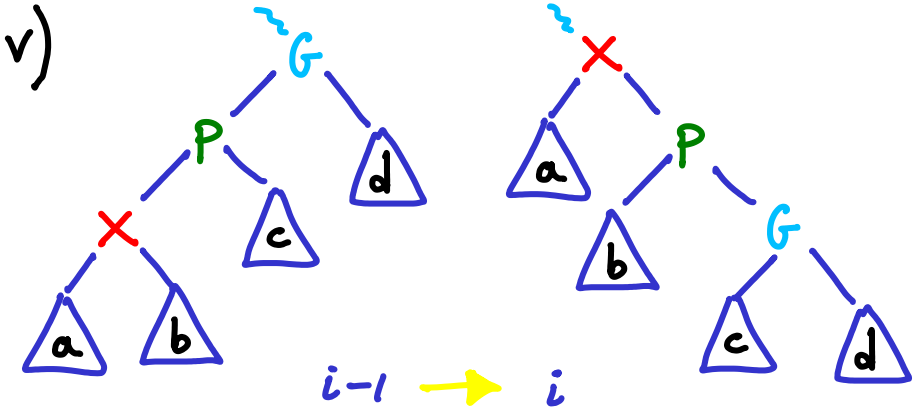
$\frac{\log A + \log B}{2} \leq \log\left(\frac{A+B}{2}\right)$

$s_i(G) + s_{i-1}(x) = 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_{i-1}(x))}{2} \leq 2 \cdot \log\left(\frac{\text{size}_i(G) + \text{size}_{i-1}(x)}{2}\right)$

$< 2 \cdot \log\left(\frac{\text{size}_i(x)}{2}\right) = 2 \cdot [s_i(x) - 1]$

$s_i(G) + s_{i-1}(x) < 2s_i(x) - 2$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$



What happens when we zig-zig x?

$$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$$

$$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$$

$$\frac{\log A + \log B}{2} \leq \log\left(\frac{A+B}{2}\right)$$

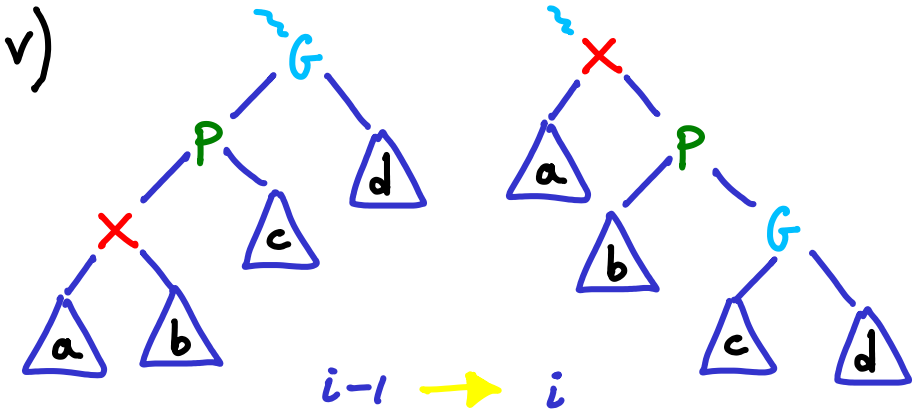
$$s_i(G) + s_{i-1}(x) = 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_{i-1}(x))}{2} \leq 2 \cdot \log\left(\frac{\text{size}_i(G) + \text{size}_{i-1}(x)}{2}\right)$$

$$< 2 \cdot \log\left(\frac{\text{size}_i(x)}{2}\right) = 2 \cdot [s_i(x) - 1]$$

$$s_i(G) + s_{i-1}(x) < 2s_i(x) - 2$$

$$\Delta\Phi < [2s_i(x) - 2] + s_i(x) - 3s_{i-1}(x)$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$



What happens when we zig-zig x?

$\Delta\Phi < s_i(G) + s_i(x) - 2s_{i-1}(x)$

$= s_i(G) + s_{i-1}(x) + s_i(x) - 3s_{i-1}(x)$

$\frac{\log A + \log B}{2} \leq \log\left(\frac{A+B}{2}\right)$

$s_i(G) + s_{i-1}(x) = 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_{i-1}(x))}{2} \leq 2 \cdot \log\left(\frac{\text{size}_i(G) + \text{size}_{i-1}(x)}{2}\right)$

$< 2 \cdot \log\left(\frac{\text{size}_i(x)}{2}\right) = 2 \cdot [s_i(x) - 1]$

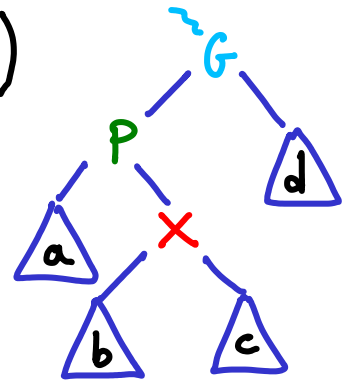
$s_i(G) + s_{i-1}(x) < 2s_i(x) - 2$

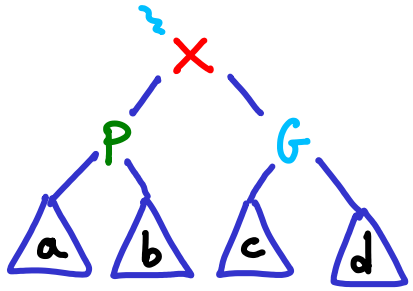
$\Delta\Phi < [2s_i(x) - 2] + s_i(x) - 3s_{i-1}(x) = 3[s_i(x) - s_{i-1}(x)] - 2$

no g: $\Delta\Phi < s_i(x) - s_{i-1}(x)$

zig-zig: $\Delta\Phi < 3[s_i(x) - s_{i-1}(x)] - 2$ per step

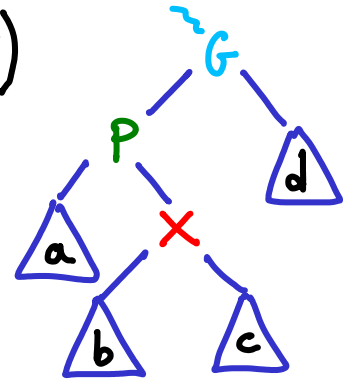
$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$

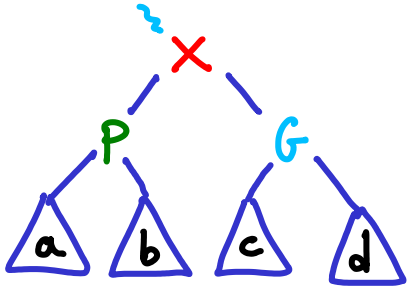




What happens when we zig-zag x ?

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





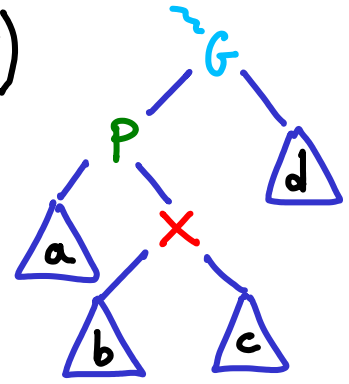
What happens when we zig-zag x ?

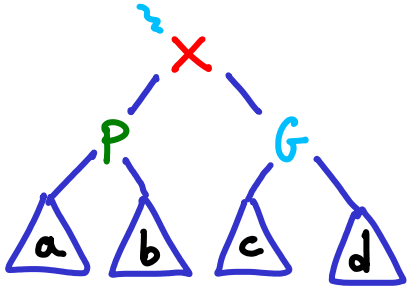
$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

$i-1 \rightarrow i$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x ?

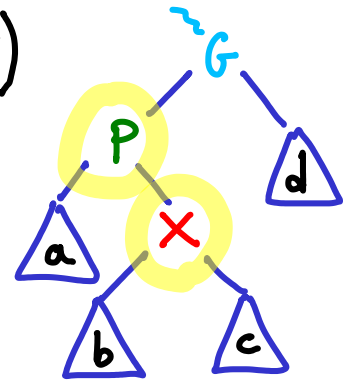
$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

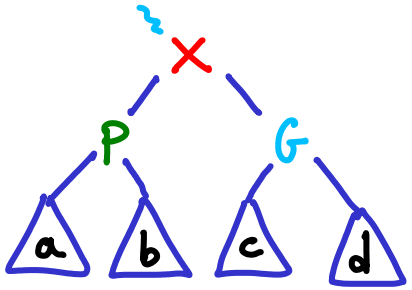
$i-1 \rightarrow i$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x ?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

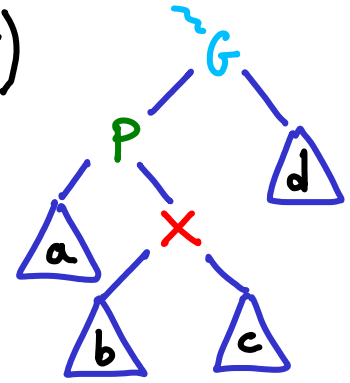
$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

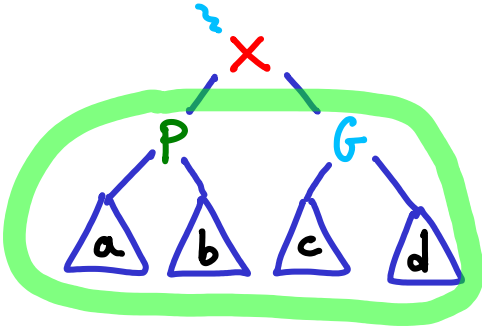
$< s_i(G) + s_i(P) - 2\,s_{i-1}(x)$

$s_{i-1}(x) < s_{i-1}(P)$

$i-1 \rightarrow i$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x ?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

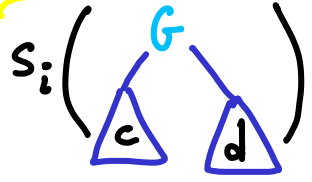
$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

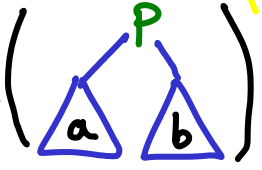
$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

$< s_i(G) + s_i(P) - 2\,s_{i-1}(x)$

$i-1 \rightarrow i$

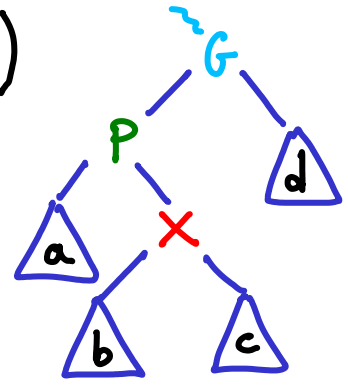
$s_{i-1}(x) < s_{i-1}(P)$

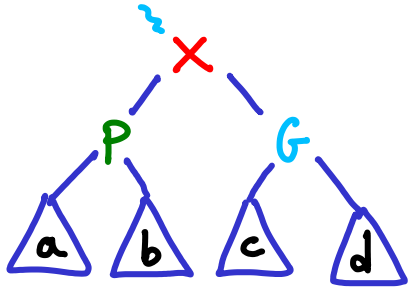
s_i(

) + s_i(

)

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

$< s_i(G) + s_i(P) - 2\,s_{i-1}(x)$

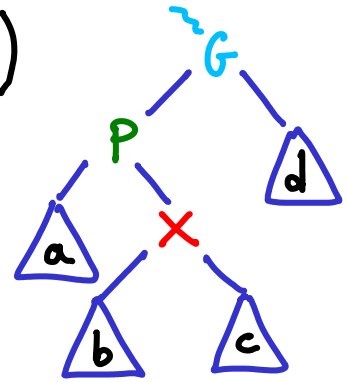
$s_{i-1}(x) < s_{i-1}(P)$

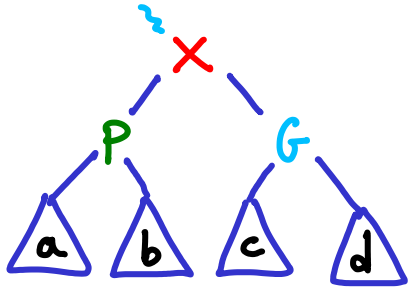
$i-1 \to i$

s_i(G, P, c, d) + s_i(P, a, b) $\sim$ s_i(G, c, d) + s_{i-1}(X, a, b)

from zig-zig case

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x ?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

$< s_i(G) + s_i(P) - 2\,s_{i-1}(x)$

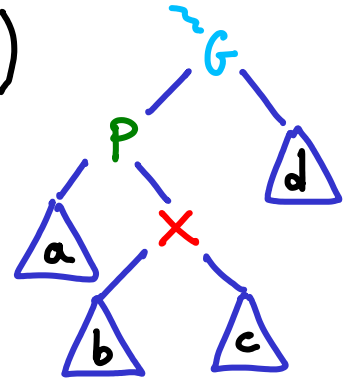
$s_{i-1}(x) < s_{i-1}(P)$

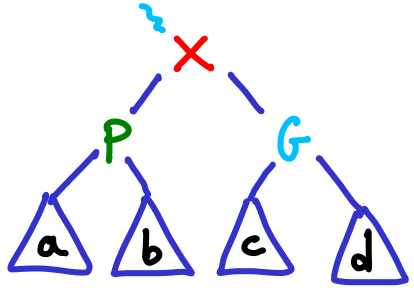
$i-1 \rightarrow i$

s_i(G: c, d) + s_i(P: a, b) $\sim s_i$(G: c, d) + s_{i-1}(X: a, b) from zig-zig case

$\Delta\Phi < 2s_i(x) - 2 - 2\,s_{i-1}(x) = 2[s_i(x)\ s_{i-1}(x)] - 2$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x ?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

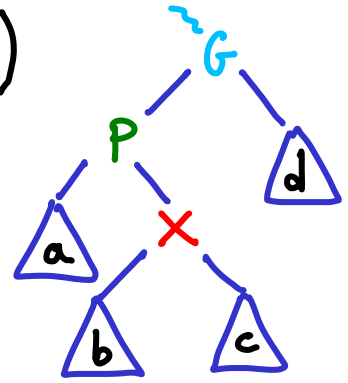
$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

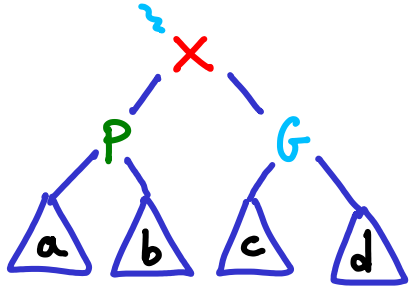
$< s_i(G) + s_i(P) - 2\,s_{i-1}(x)$

$i-1 \rightarrow i$

$s_{i-1}(x) < s_{i-1}(P)$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x ?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

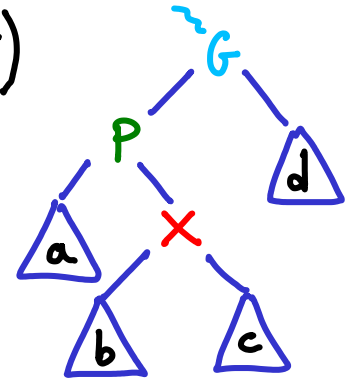
$< s_i(G) + s_i(P) - 2\,s_{i-1}(x)$

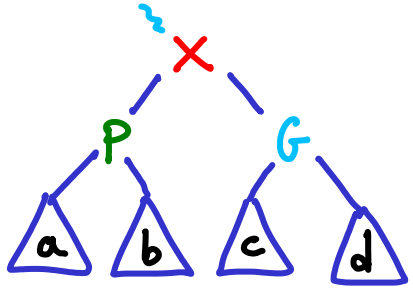
$= 2 \cdot \dfrac{\log(\text{size}_i(G)) + \log(\text{size}_i(P))}{2}$

$i-1 \rightarrow i$

$s_{i-1}(x) < s_{i-1}(P)$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x ?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$i-1 \rightarrow i$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

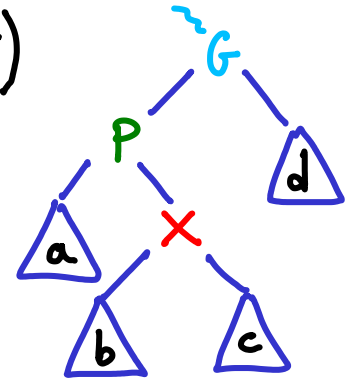
$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

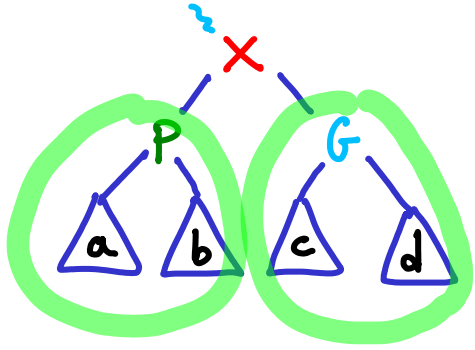
$s_{i-1}(x) < s_{i-1}(P)$

$< s_i(G) + s_i(P) - 2\, s_{i-1}(x)$

$$= 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_i(P))}{2} \leq 2\log\left(\frac{\text{size}_i(G) + \text{size}_i(P)}{2}\right)$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x ?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

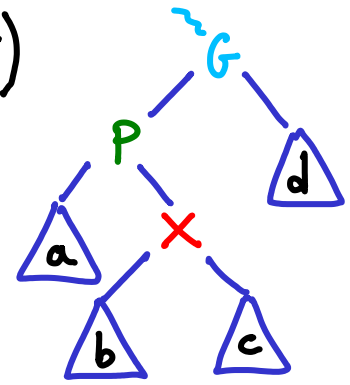
$< s_i(G) + s_i(P) - 2\,s_{i-1}(x)$

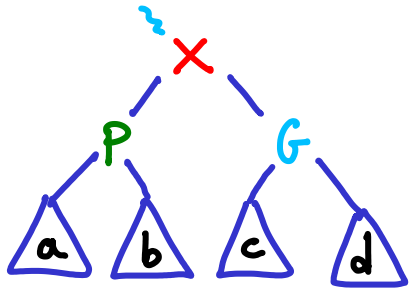
$i-1 \rightarrow i$

$s_{i-1}(x) < s_{i-1}(P)$

$$= 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_i(P))}{2} \leq 2\log\left(\frac{\text{size}_i(G) + \text{size}_i(P)}{2}\right) < 2\log\left(\frac{\text{size}_i(x)}{2}\right)$$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

$< s_i(G) + s_i(P) - 2\,s_{i-1}(x)$

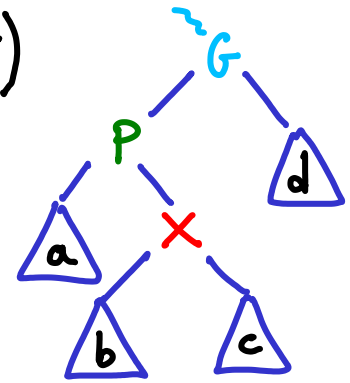
$s_{i-1}(x) < s_{i-1}(P)$

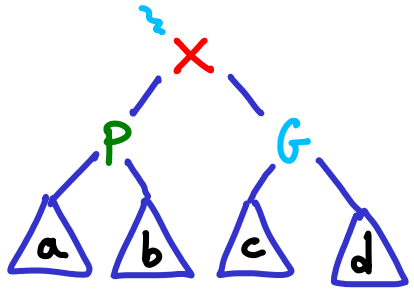
$i-1 \rightarrow i$

$$= 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_i(P))}{2} \leq 2\log\left(\frac{\text{size}_i(G) + \text{size}_i(P)}{2}\right) < 2\log\left(\frac{\text{size}_i(x)}{2}\right)$$

$\Delta\Phi < 2s_i(x) - 2 - 2\,s_{i-1}(x)$

$\Phi = \sum_{\text{all } v} s(v)$ // $s(v) = \log_2(\#\text{nodes in subtree of } v)$





What happens when we zig-zag x ?

$\Delta\Phi = \Delta s(G) + \Delta s(P) + \Delta s(x)$

$= s_i(G) - s_{i-1}(G) + s_i(P) - s_{i-1}(P) + s_i(x) - s_{i-1}(x)$

$= s_i(G) + s_i(P) - s_{i-1}(P) - s_{i-1}(x)$

$< s_i(G) + s_i(P) - 2\,s_{i-1}(x)$

$i-1 \rightarrow i$

$s_{i-1}(x) < s_{i-1}(P)$

$$= 2 \cdot \frac{\log(\text{size}_i(G)) + \log(\text{size}_i(P))}{2} \leq 2\log\left(\frac{\text{size}_i(G) + \text{size}_i(P)}{2}\right) < 2\log\left(\frac{\text{size}_i(x)}{2}\right)$$

$\Delta\Phi < 2s_i(x) - 2 - 2\,s_{i-1}(x) = 2[s_i(x) - s_{i-1}(x)] - 2$

no g: $\Delta\Phi < s_i(x) - s_{i-1}(x)$

zig-zig: $\Delta\Phi < 3[s_i(x) - s_{i-1}(x)] - 2$

zig-zag: $\Delta\Phi < 2[s_i(x) - s_{i-1}(x)] - 2$

} per step

no g: $\Delta\Phi < s_i(x) - s_{i-1}(x)$

zig-zig: $\Delta\Phi < 3[s_i(x) - s_{i-1}(x)] - 2$

zig-zag: $\Delta\Phi < 2[s_i(x) - s_{i-1}(x)] - 2$

← at most once per step

---

total $\Delta\Phi$ for one splay $< -2k + \sum 3[s_i(x) - s_{i-1}(x)]$ // k = # zig-zags & zig-zigs

no g: $\Delta\Phi < s_i(x) - s_{i-1}(x)$

zig-zig: $\Delta\Phi < 3[s_i(x) - s_{i-1}(x)] - 2$

zig-zag: $\Delta\Phi < 2[s_i(x) - s_{i-1}(x)] - 2$

} per step

---

total $\Delta\Phi$ for one splay $< -2k + \sum 3[s_i(x) - s_{i-1}(x)]$ // k = # zig-zags & zig-zigs

telescope

$= 3[s(\text{root}) - s_0(x)] - 2k$

no g: $\Delta\Phi < s_i(x) - s_{i-1}(x)$

zig-zig: $\Delta\Phi < 3[s_i(x) - s_{i-1}(x)] - 2$

zig-zag: $\Delta\Phi < 2[s_i(x) - s_{i-1}(x)] - 2$

} per step

---

total $\Delta\Phi$ for one splay $< -2k + \sum 3[s_i(x) - s_{i-1}(x)]$ // k = # zig-zags & zig-zigs

telescope

$= 3[s(\text{root}) - s_0(x)] - 2k$

$\leq 3 \cdot s(\text{root}) - 2k$

$\leq 3 \cdot \log n - 2k$

no g: $\Delta\Phi < s_i(x) - s_{i-1}(x)$

zig-zig: $\Delta\Phi < 3[s_i(x) - s_{i-1}(x)] - 2$

zig-zag: $\Delta\Phi < 2[s_i(x) - s_{i-1}(x)] - 2$

} per step

---

total $\Delta\Phi$ for one splay $< -2k + \sum 3[s_i(x) - s_{i-1}(x)]$ // k = # zig-zags & zig-zigs

telescope

$= 3[s(\text{root}) - s_0(x)] - 2k$

$\leq 3 \cdot s(\text{root}) - 2k$

$\leq 3 \cdot \log n - 2k$

Amortized cost $\hat{c} = c + \Delta\Phi$

no g: $\Delta\Phi < s_i(x) - s_{i-1}(x)$

zig-zig: $\Delta\Phi < 3[s_i(x) - s_{i-1}(x)] - 2$

zig-zag: $\Delta\Phi < 2[s_i(x) - s_{i-1}(x)] - 2$

} per step

---

total $\Delta\Phi$ for one splay $< -2k + \sum 3[s_i(x) - s_{i-1}(x)]$ // k = # zig-zags & zig-zigs

telescope

$= 3[s(\text{root}) - s_0(x)] - 2k$

$\leq 3 \cdot s(\text{root}) - 2k$

$\leq 3 \cdot \log n - 2k$

Amortized cost $\hat{c} = c + \Delta\Phi \leq 1 + 2k + 3 \cdot \log n - 2k$

$\leq$ 1 rotation at top
+2 rotations for all other steps

no g: $\Delta\Phi < s_i(x) - s_{i-1}(x)$

zig-zig: $\Delta\Phi < 3[s_i(x) - s_{i-1}(x)] - 2$

zig-zag: $\Delta\Phi < 2[s_i(x) - s_{i-1}(x)] - 2$

} per step

---

total $\Delta\Phi$ for one splay $< -2k + \sum 3[s_i(x) - s_{i-1}(x)]$ // k = # zig-zags & zig-zigs

telescope

$= 3[s(\text{root}) - s_0(x)] - 2k$

$\leq 3 \cdot s(\text{root}) - 2k$

$\leq 3 \cdot \log n - 2k$

≤ 1 rotation at top
+2 rotations for all other steps

Amortized cost $\hat{c} = c + \Delta\Phi \leq 1 + 2k + 3 \cdot \log n - 2k = O(\log n)$

Summary: one splay (to top) costs $O(\log n)$ amortized

↳ $\Delta\Phi$ of splay pays for rotations during splay

Summary: one splay (to top) costs $O(\log n)$ amortized

↳ $\Delta\Phi$ of splay pays for rotations during splay

insert causes $0 < \Delta\Phi \leq \log n$ but also costs up to n

Summary: one splay (to top) costs $O(\log n)$ amortized

↳ $\Delta\Phi$ of splay pays for rotations during splay

insert causes $0 < \Delta\Phi \leq \log n$ but also costs up to n

↳ can adjust (multiply) Φ so that splay pays for this cost too

splay path = search path

Summary: one splay (to top) costs $O(\log n)$ amortized

↳ $\Delta\Phi$ of splay pays for rotations during splay

insert causes $0 < \Delta\Phi \leq \log n$ but also costs up to n

↳ can adjust (multiply) Φ so that splay pays for this cost too

splay path = search path

Amortized cost $\sum_{j=1}^{m} \hat{c}_j$ of m operations on n elements $= O(m \log n)$

Summary: one splay (to top) costs $O(\log n)$ amortized

↳ $\Delta\Phi$ of splay pays for rotations during splay

insert causes $0 < \Delta\Phi \leq \log n$ but also costs up to n

↳ can adjust (multiply) Φ so that splay pays for this cost too

splay path = search path

Amortized cost $\sum_{j=1}^{m} \hat{c}_j$ of m operations on n elements $= O(m \log n)$

If tree doesn't start empty,

each node could contribute $\leq \log n$ to Φ_0

could be $\lceil n \log n$

Summary: one splay (to top) costs $O(\log n)$ amortized

↳ $\Delta\Phi$ of splay pays for rotations during splay

insert causes $0 < \Delta\Phi \leq \log n$ but also costs up to n

↳ can adjust (multiply) Φ so that splay pays for this cost too

splay path = search path

Amortized cost $\sum_{j=1}^{m} \hat{c}_j$ of m operations on n elements $= O(m \log n)$

If tree doesn't start empty,

each node could contribute $\leq \log n$ to $\Phi_n - \Phi_0$

(could be zero) (could be $n \log n$)

Summary: one splay (to top) costs $O(\log n)$ amortized

↳ $\Delta\Phi$ of splay pays for rotations during splay

insert causes $0 < \Delta\Phi \leq \log n$ but also costs up to n

↳ can adjust (multiply) Φ so that splay pays for this cost too

splay path = search path

Amortized cost $\sum_{j=1}^{m} \hat{c}_j$ of m operations on n elements $= O(m \log n)$

If tree doesn't start empty,

each node could contribute $\leq \log n$ to $\Phi_n - \Phi_0$

(could be zero; could be $n \log n$)

$$\sum_{j=1}^{m} \hat{c}_j = \sum_{j=1}^{m} c + \Phi_n - \Phi_0$$

Summary: one splay (to top) costs $O(\log n)$ amortized

↳ $\Delta\Phi$ of splay pays for rotations during splay

insert causes $0 < \Delta\Phi \leq \log n$ but also costs up to n

↳ can adjust (multiply) Φ so that splay pays for this cost too

splay path = search path

Amortized cost $\sum_{j=1}^{m} \hat{c}_j$ of m operations on n elements $= O(m \log n)$

If tree doesn't start empty,

each node could contribute $\leq \log n$ to $\Phi_n - \Phi_0$ so

(could be zero; could be $n \log n$)

$$\sum_{j=1}^{m} \hat{c}_j = \sum_{j=1}^{m} c + \Phi_n - \Phi_0 \Rightarrow \sum_{j=1}^{m} c = \sum_{j=1}^{m} \hat{c}_j + \Phi_0 - \Phi_n = O(m \log n) + O(n \log n)$$