Amortized Revisited

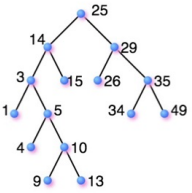
What if $\Phi_0 \neq 0$ & $\Phi_n \neq 0$. Then

$$\Phi_0 - \Phi_n + \sum a_i = \sum t_i$$

BST

Binary Search Trees

- Binary search tree Review:
  - Stores ordered set S of *keys*: one node per key.
  - An in-order traversal of the tree lists the keys in sorted order
  - Supports many operations:
    - insert(x) : S := S ∪ {x}
    - delete(x) : S := S \ {x}
    - lookup(x) : x ∈ S?
    - pred(x) : max{y ∈ S | y ≤ x}
    - succ(x) : min{y ∈ S | y > x}
    - ...and others.

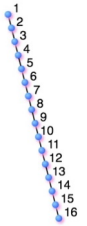


Binary Search Trees

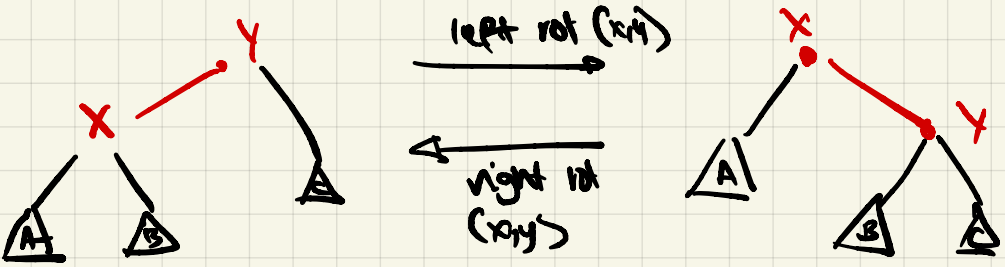
- Straightforward lower bound for searching when $|S| = n$.
  - Number of nodes at depth i is 2^i.
  - So worst-case access time (depth of a node) is $\geq \log n$
- Matching upper bound:
  - Arrange keys in a perfectly balanced tree (height $\log n$)
- One wrinkle...
  - If you're inserting & deleting keys the tree could become unbalanced
  - Red-black trees, AVL trees,... keep height $O(\log n)$
- End of story?

Binary Search Trees

- A balanced search tree may not be optimal!
  - An extreme example: keys 1,2,3,4,5,...,16
  - Probability of accessing 1 = 1/2
  - Prob. for 2 = 1/4
  - Prob. for 3 = 1/8
  - ...
  - Prob for 15,16 = $(1/2)^{15}$
  - The optimal search tree is very unbalanced



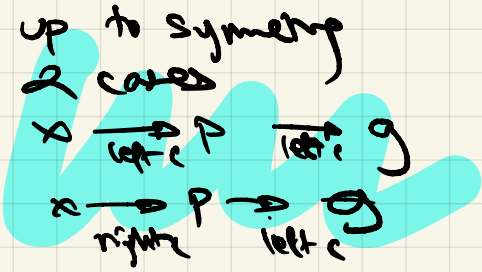
Adjusting tree with Rotation



Self-Adjusting Binary Search Trees

- The ***splay*** heuristic:
  - After searching for x, rotate x to the root position 2 edges at a time
  - A few cases to consider...

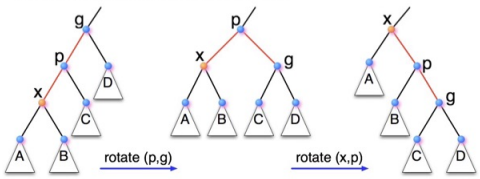
to keep going until root



(1)

Self-Adjusting Binary Search Trees

- After searching for x, rotate x to the root *2 edges at a time*
- Case 1 **"zig-zig"**
  - x is the left child of its parent p, which is the left child of g
  - Rotate the edge (p, g) then the edge (x, p)
    (now x has a new parent & grandparent)

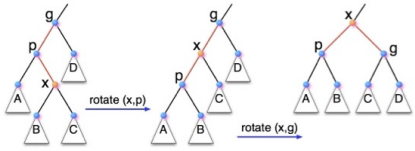


- The reverse situation (x is the right child of p, which is the right child of g) is symmetric

(2)

Self-Adjusting Binary Search Trees

- After searching for x, rotate x to the root *2 edges at a time*
- Case 2 **"zig-zag"**
  - x is the right child of its parent p, which is the left child of g
  - Rotate the edge (x, p) then the edge (x, g)
    (now x has a new parent & grandparent)



- The reverse situation (x is the left child of p, which is the right child of g) is symmetric

(3) only 1 ancestor → zig → one rotation

Amortized Analysis

Amortized Analysis

- Need a good potential function Φ...
- If the tree is very unbalanced, splaying seems to rebalance (part of) the tree.

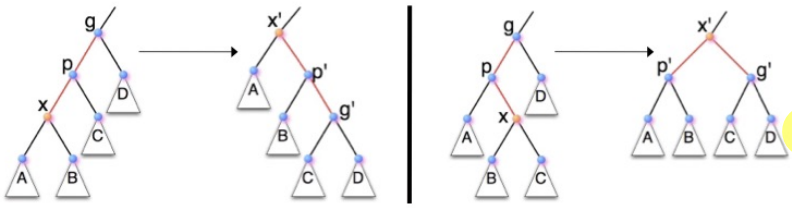
- Φ_i = potential after i searches
- t_i = actual time of i^{th} search = depth of the key we're searching for
- $a_i = t_i + \Phi_i - \Phi_{i-1}$
- If splaying long paths creates more balance in the tree
  - Φ should be *large* when the tree is unbalanced and *small* when balanced

Amortized Analysis

- $w(x)$ = the weight of x. *(We get to choose the weight function.)*
- $|x|$ = the ***total*** weight of all nodes in x's subtree.
- $\Phi = \Sigma_x \log_2|x|$
- $a_i = t_i + \Phi_i - \Phi_{i-1}$

- **Theorem:** (Sleator and Tarjan) Fix any weight function w and let $W = \sum_x w(x)$ be the sum of all weights. The amortized time to access x is $O\left(\log\left(\frac{W}{w(x)}\right)\right)$.

- $x', p', g' = x, p, g$ after one splay-step (zig-zig, zig-zag, zig)
- **Claim:** If $\Delta\Phi \le -2 + c \cdot (\log|x'| - \log|x|)$ then the amortized access time is $\le c \cdot \log\left(\frac{W}{w(x)}\right)$.



- $x', p', g' = x, p, g$ after one splay-step (zig-zig, zig-zag, zig)
- **Claim:** If $\Delta\Phi \le -2 + c \cdot (\log|x'| - \log|x|)$ then the amortized access time is $\le c \cdot \log\left(\frac{W}{w(x)}\right)$.
- **Pf.** Suppose there are k zigzig/zigzag/zig steps
  - $x_i = x$ after the i^{th} step.
  - $[\text{amort. time}] \le [\text{time}] + \sum_{1 \le i \le k}[-2 + c(\log|x_i| - \log|x_{i-1}|)]$

$$\le 2k + (-2k) + c \cdot \log(|\text{root}|) - c \cdot \log|x_0|)$$

$$\le c \log(|\text{root}|/w(x)) = c \log(W/w(x)).$$

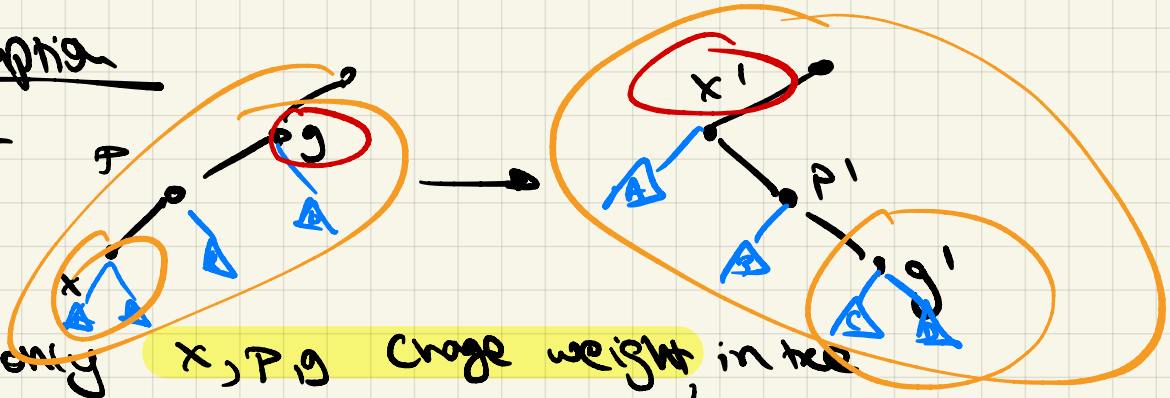
- **Theorem:** (Sleator and Tarjan) Fix any weight function w and let $W = \sum_x w(x)$ be the sum of all weights. The amortized time to access x is $O\left(\log\left(\frac{W}{w(x)}\right)\right)$.

- **Working Set Theorem.** The key-set is $\{1,2,\dots,n\}$. The access sequence is $\sigma = 123\cdots n\sigma_1\sigma_2\sigma_3\dots\sigma_m$.
  - Let $\text{last}(i) = \max\{j < i : \sigma_j = \sigma_i\}$.
  - $\text{WS}(i) = \{\sigma_{\text{last}(i)}, \sigma_{\text{last}(i)+1}, \dots, \sigma_i\}$.
  - The total cost for accessing σ in a splay tree is

$$O\left(n \log n + m + \sum_{i=1}^{m} \log(|\text{WS}(i)|)\right)$$

Show assumption

① zig zig –



note: only x, p, g change weight in tree

$$\Delta\Phi = \log|x'| + \log|g'| + \log|p'| - (\log|x| + \log|g| + \log|p|)$$

$$= \log_2\left(\frac{|x'| \cdot |g'| \cdot |p'|}{|x| \cdot |g| \cdot |p|}\right) \rightarrow \text{worst} \leq -2 + c\log\left(\frac{|x'|}{|x|}\right)$$

$c = 3$

equiv

$$\log_2\left(\frac{|x'| \cdot |g'| \cdot |p'|}{|x| \cdot |p| \cdot |x'|}\left(\frac{|x|}{|x'|}\right)^3\right) \leq -2$$

note $|g| = |x'|$

$$= \log_2\left(\frac{|g'| \cdot |p'|}{|x|\ |p|}\frac{|x|^3}{|x'|^3}\right)$$

$|p'| \leq |x'|$

$|p| \geq |x|$

bound

$$\leq \log_2\left(\frac{|g'| \cdot |x|^3}{|x|\ |x|\ |x'|^3}\right)$$

$$= \log_2\left(\frac{|g'|}{|x'|} \cdot \frac{|x|}{|x'|}\right)$$

triangle

ratio of non-overlapping trees

$\frac{|g|}{|x|} = \varepsilon \in [0,1)$

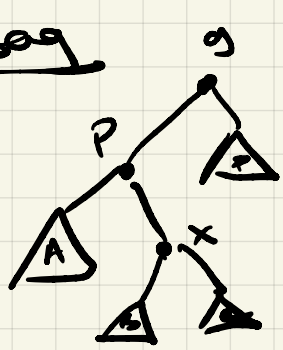
$\frac{|x|}{|x'|} < 1 - \varepsilon$

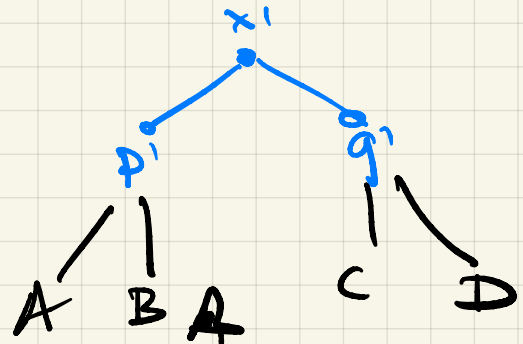
$$= \log_2(\varepsilon(1-\varepsilon))$$

worst max $\varepsilon = \frac{1}{2}$

$$\leq \log_2(0.25) = -2$$

Zig Zag





$$\Delta\Phi = \log\left(\frac{x' \cdot p' \cdot g'}{x \cdot p \cdot g}\right) \leq -2 + 3\log\left(\frac{w}{x}\right)$$

no bang

$$\Rightarrow \log\left(\frac{x' \cdot p' \cdot g'}{x \cdot p \cdot g} \cdot \frac{x^3}{x'^3}\right) \leq -2$$

$$\leq \log\left(\frac{x'}{x} \cdot \frac{p'}{p} \cdot \frac{g'}{g} \cdot \frac{x^2}{x'^3}\right) \qquad g = x' \qquad x < p$$

$$\leq \log\left(\frac{p'}{x'} \cdot \frac{g'}{x'}\right) \rightarrow \text{disjoint ranges} \qquad \frac{p'}{x'} = \varepsilon \leq 1, \quad \frac{g'}{x'} = 1-\varepsilon$$

$$\leq \log(\varepsilon(1-\varepsilon)) \rightarrow \text{maximal val}$$

$$\leq \log\left(\frac{1}{4}\right) = -2$$

Implication

Result

t_i = depth of x

$$a_i = t_i + \Delta\Phi \leq 3\log\left(\frac{w}{w(x)}\right) + 1$$

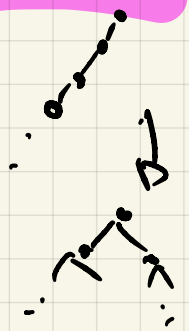
→ Zig case

Access thm $w(x) = 1 \ \forall x \Rightarrow W = n$, n accesses

$$\sum t_i = (\Phi_0 - \Phi_n) + \sum a_i$$

$$\leq n\log n + n\,3\log n + 1$$

worst case



Cor p_i is prb of accessing node i $\quad \sum p(i) = 1$

let $w(i) = p(i) \Rightarrow W = 1$

So $a_i \leq 3\log\left(\frac{1}{p(x)}\right)$

$\mathbb{E}(\sum t_i) = \phi_0 - \phi_n + \mathbb{E}(\sum a_i)$

$= \phi_0 - \phi_n + 3 \sum_x p(x) \log\left(\frac{1}{p(x)}\right)$

$\rightarrow H(p) \rightarrow$ entropy

Working Set Theorem

- **Theorem:** (Sleator and Tarjan) Fix any weight function w and let $W = \sum_x w(x)$ be the sum of all weights. The amortized time to access x is $O\left(\log\left(\frac{W}{w(x)}\right)\right)$.

- **Working Set Theorem.** The key-set is $\{1,2,\dots,n\}$. The access sequence is $\sigma = 123\cdots n\sigma_1\sigma_2\sigma_3\dots\sigma_m$.
  - Let $\text{last}(i) = \max\{j < i : \sigma_j = \sigma_i\}$.
  - $\text{WS}(i) = \{\sigma_{\text{last}(i)}, \sigma_{\text{last}(i)+1}, \dots, \sigma_i\}$.
  - The total cost for accessing σ in a splay tree is

$$O\left(n\log n + m + \sum_{i=1}^{m} \log(|\text{WS}(i)|)\right)$$

prepend makes "last access time" well def

Say

$w(x) = \frac{1}{k^2}$ if x is at position k in Move to front list → this isn't constant

Access 1) splay x to the root → amor time → $\log(w(x))$

2) Adjust weight function & account for changing potential

1) $a_i \le 3\log\left(\frac{W}{w(x)}\right) = 3\log\left(\frac{W}{1/k^2}\right) = 3\log\left(\frac{n^2k^2}{6}\right)$ ← W is const

$\in O(\log(k))$

Claim (2) doesn't increase potential ∎

(3) In particular total weight of any subtree don't chg.

i.e. $|y|$ doesn't increase

note that for non x (splayed at), the weight decreases.

If $y \neq x$ then $|y|$ doesn't increase as x within y's subtree

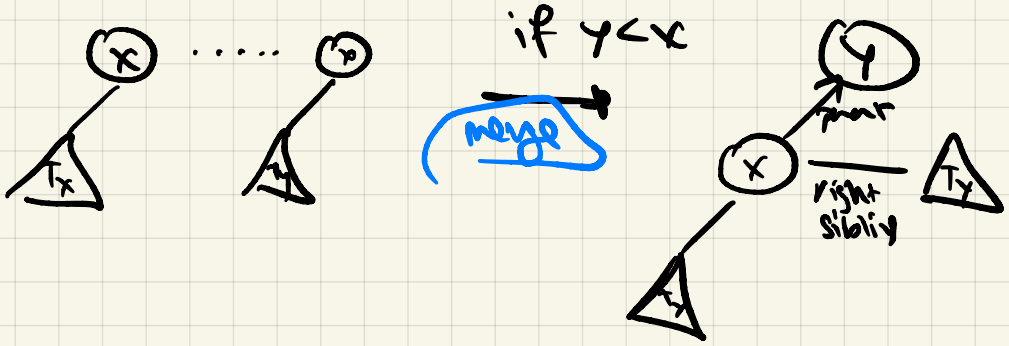
If $y = x$ → $|y| = |W|$ is total weight of tree (same)

$|y|$ is stable

Pairing Heap

Supports → insert, delete-min, decrease key in $O(\log n)$ amor

big idea → merging trees on same level



each node has a parent ptr (only actual parent if leftmost child else left sibling)

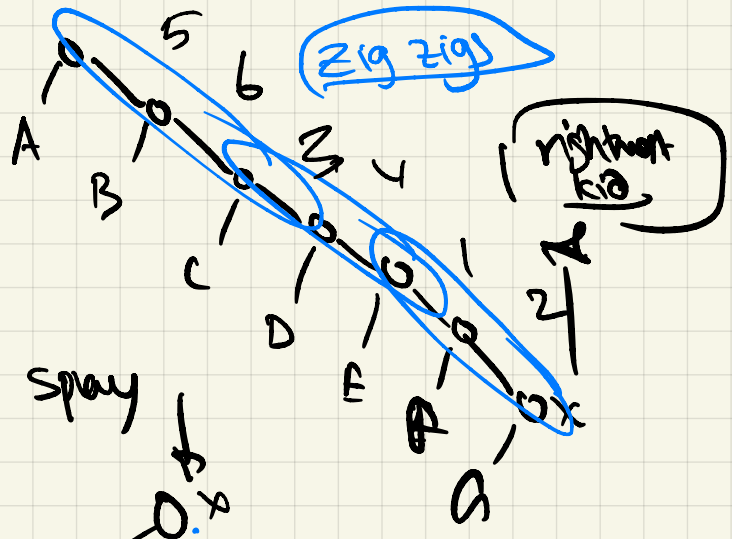
and right sibling ptr

Analyzing Paring Heap

$|x|$ = # nodes in x's subtree in the binary representation

$\Phi = \sum_x \log(x)$

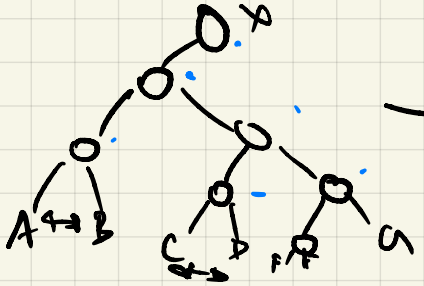
Claim deletemin $\leq$ access rightmost child in splay



rotating x to top is like rotating (5,3,1) → 2,4 6

Simultaneous paring pass

Accum pass



→ Can rotate labels of blue without consequence

→ flip letters maybe too

↳ doesn't change potential

} i.e pairing up to swap is identical to splay

Pairing heap — 1986

delete-min	$O(\log n)$
decrease-key	$O(\log n)$
insert	$O(\log n)$

$\rightarrow 2^{O(\sqrt{\log n})}$ $\quad \Omega(\log n)$