Advanced Data Structures

Splay Trees

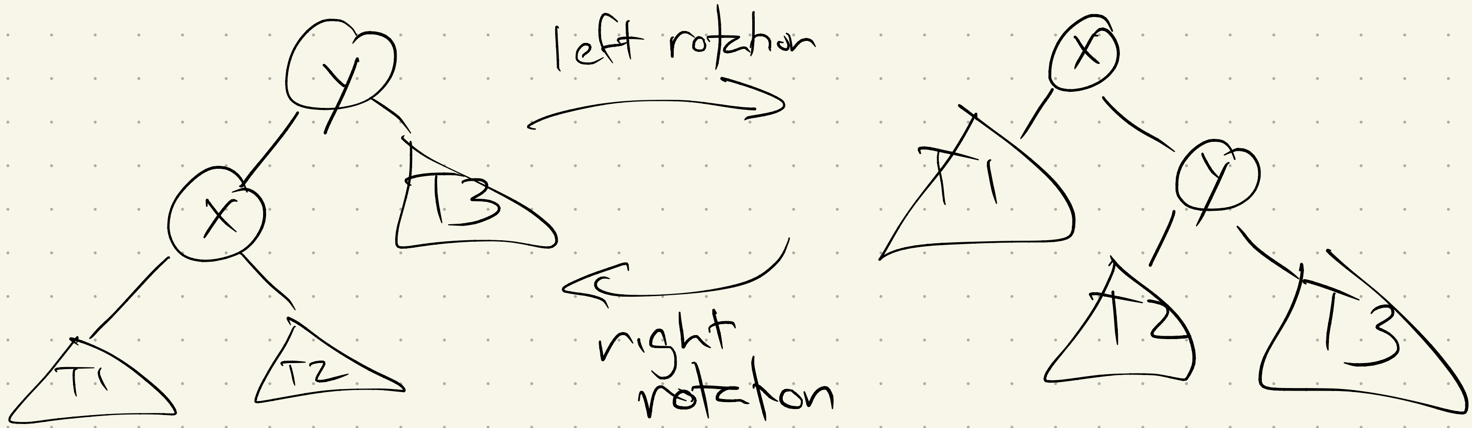
Recap

- Sub to finish next Friday
- HW due Friday

Questions?

Today: Back to binary trees

Rotations: (single rotation)

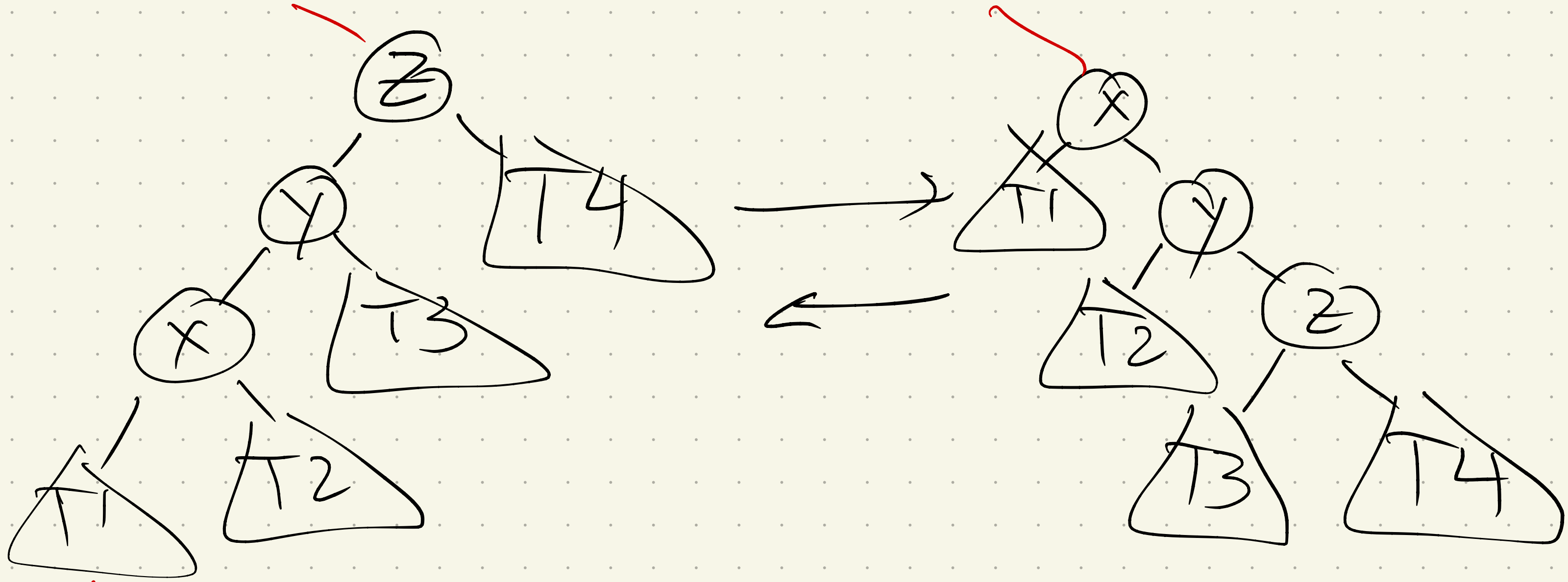


Note: If in an unbalanced tree, "rotate up to root" can take $O(n)$ time

Next: 2 kinds of double rotations

① Roller coaster "zig zig"

(Just what it sounds like)



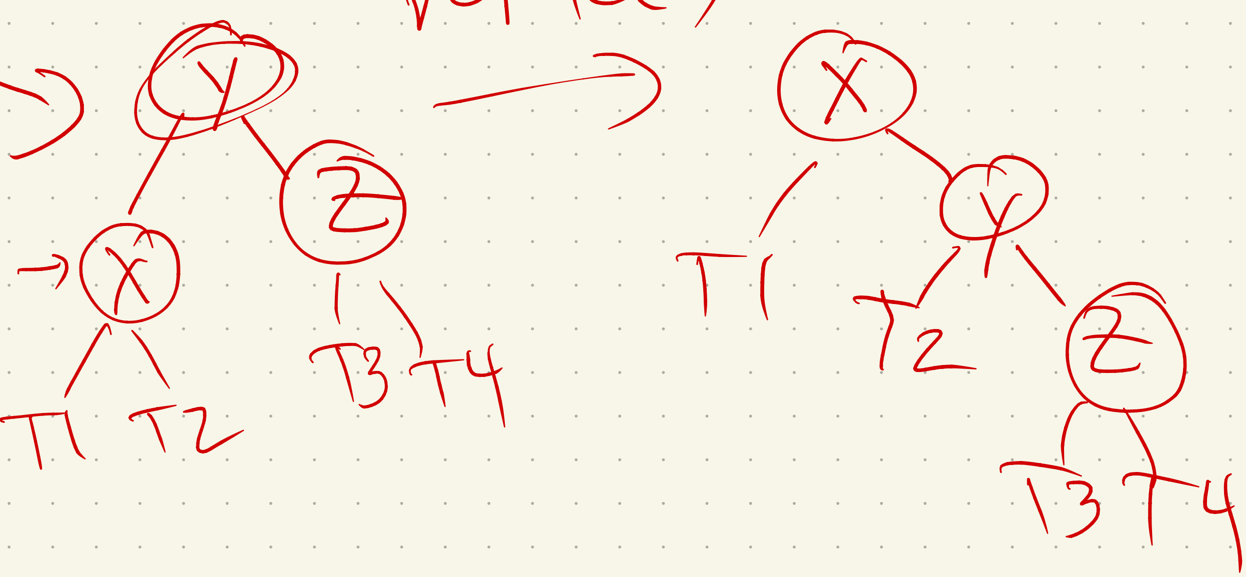
How: rotate(y)
rotate(x)

both same direction

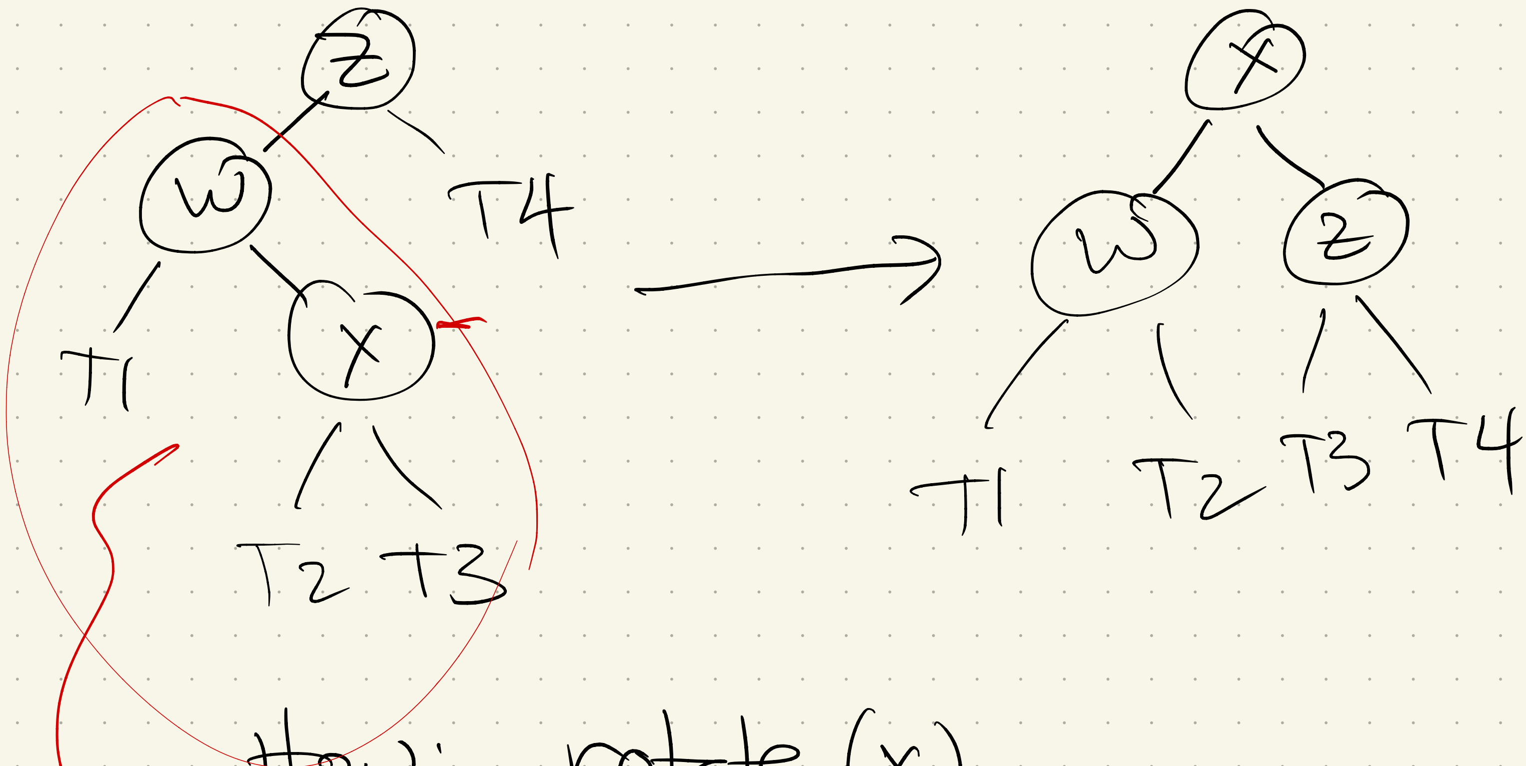
rotate(y)

Why?

rotate(x)



② Zig-Zag:

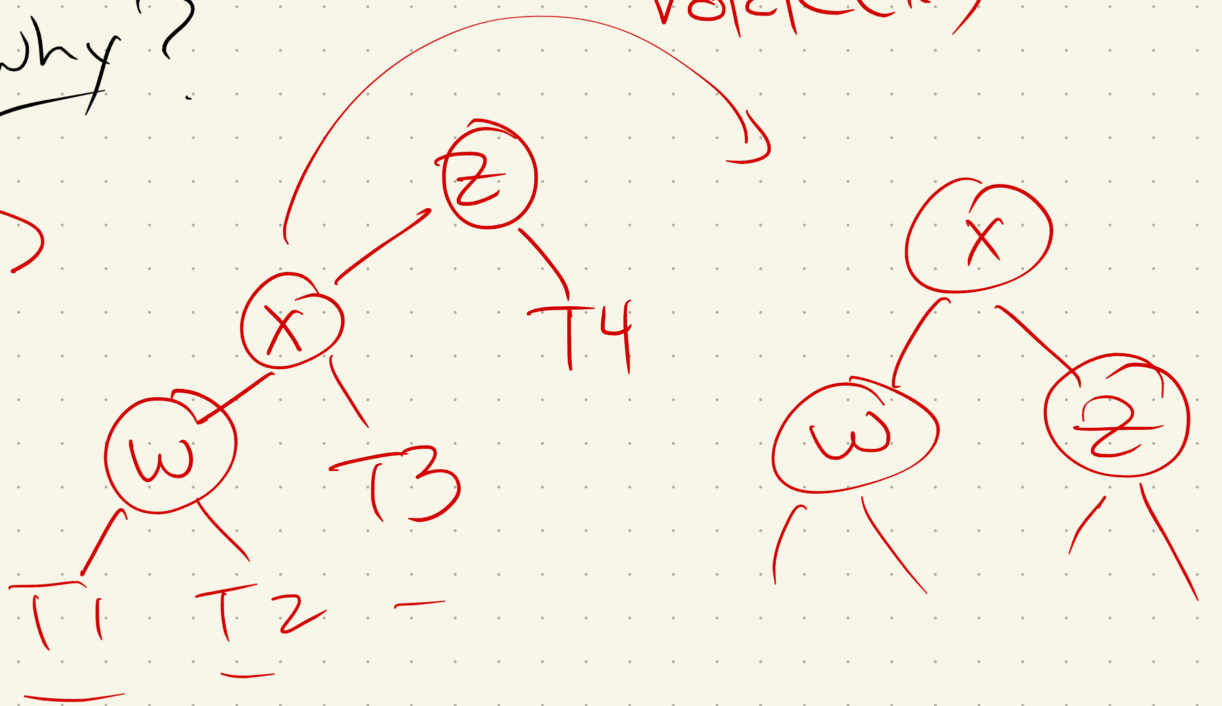


How: rotate(x)
rotate(x)

rotate(x)

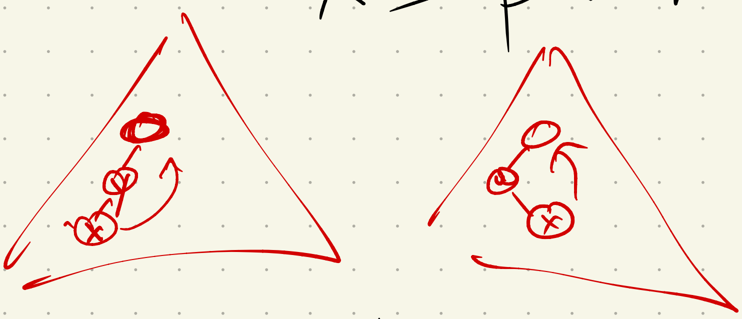
why?

rotate(x)



Note: Each double rotation

- affects x's depth: -2
- x's parent's depth (y or w): unchanged



- x's grandparent: z

$+1$ or $+2$

Runtime: $O(1)$

≤ 14 pointer updates (plus if's to check cases)

Splay(x):

While $(x \neq root)$ or $(parent(x) \neq root)$

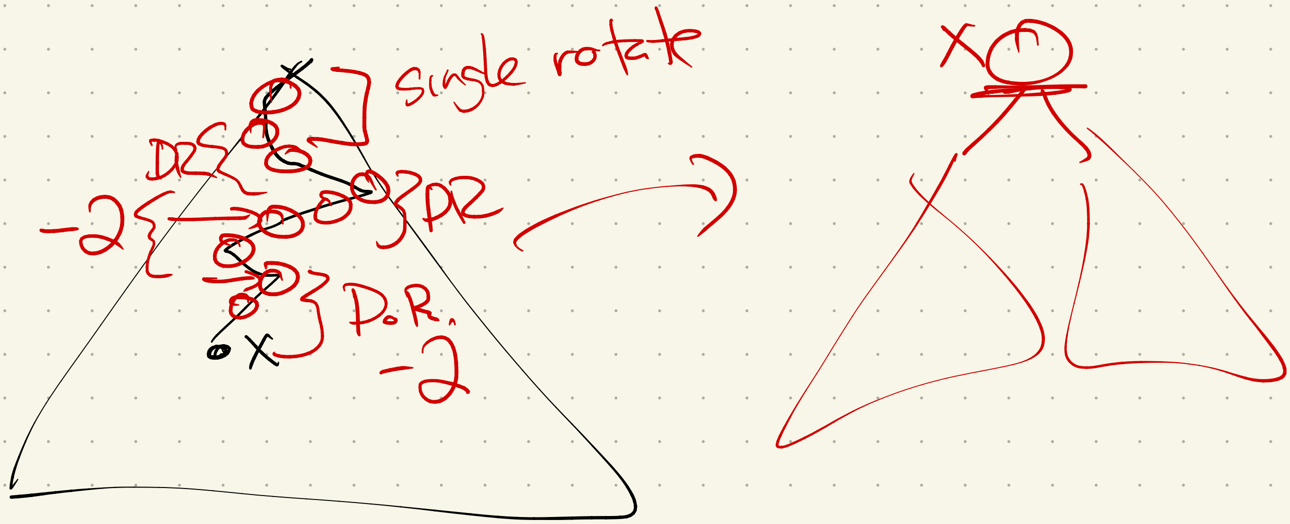
  double rotation(x)

If $x \neq root$

  rotate(x)

14 ptrs
20 if

Runtime: $\approx \frac{depth(x)}{2} \cdot O(1)$



$= O(depth(x))$

(Data structure doesn't track height/depth)

Splay Tree:

A (more or less) balanced binary tree where we splay to balance*
(* mostly!)

High level idea:
Any time a node is accessed (search/insert/delete), splay it to the root.

Why??

Amortization!

If you splay, other things balance - works out to $\Theta(\log n)$ <u>amortized</u> time per operation.

More concretely:

<u>Search(x)</u>: value

node ← BSTFind(x)
(assume this returns x, or pred/succ if x is not in tree)
splay(node)

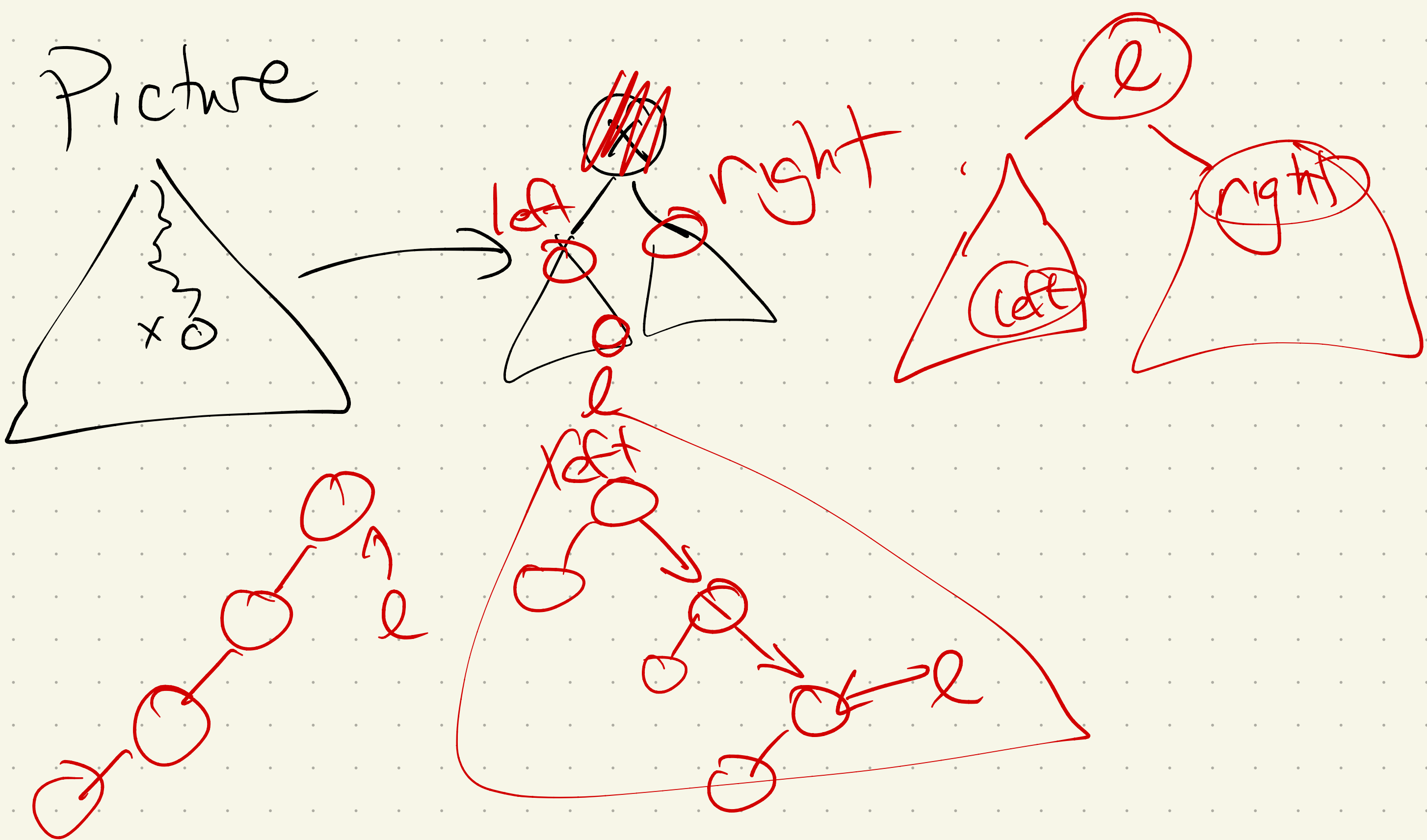
<u>Insert(x)</u>

node ← BSTInsert(x)
(assume this returns x's node in tree)
Splay(node)

Delete (x):

```
xnode ← BSTFind (x)
if xnode.value = x :
    splay (xnode)
    left ← (xnode.left)
    right ← (xnode.right)
    delete (xnode)
    l ← FindLargest (left)
    splay (l)
    l.right ← right
```

Picture

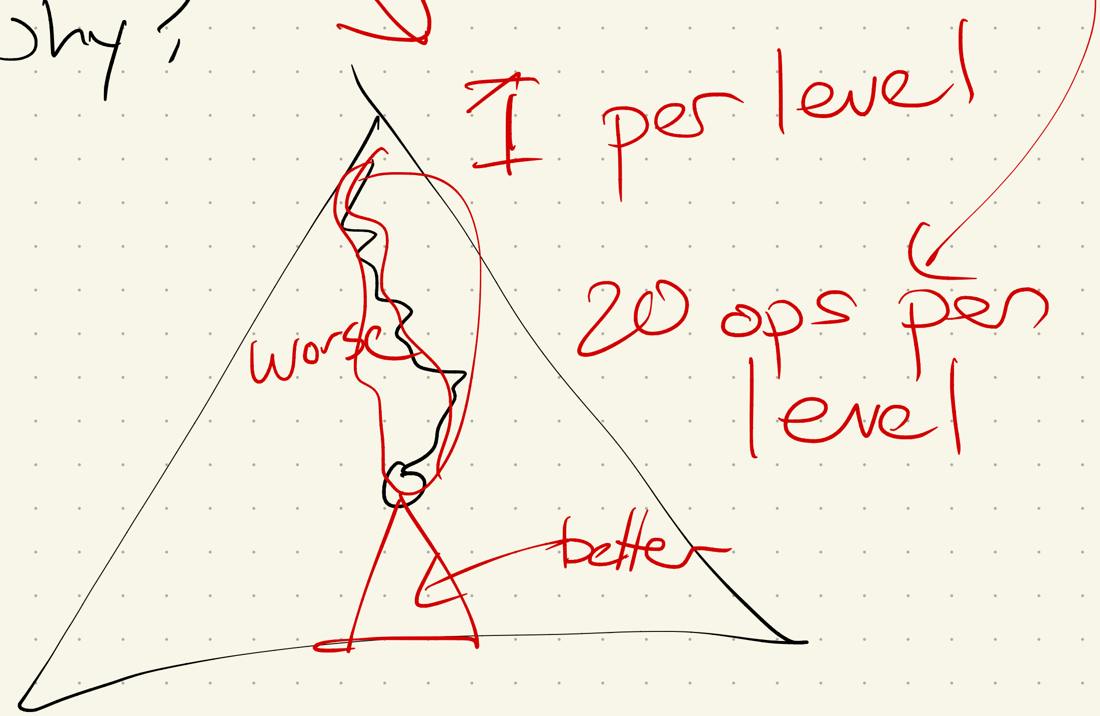


Note: Each of these has a constant # of the following:

- walk down to some node
- splay that node to root

Cost (walk) $\leq$ cost(splay)

why?



$= O(\text{depth}_T(x))$

What does it cost to splay??

Worst case: $O(n)$

To get amortized, need a potential function:

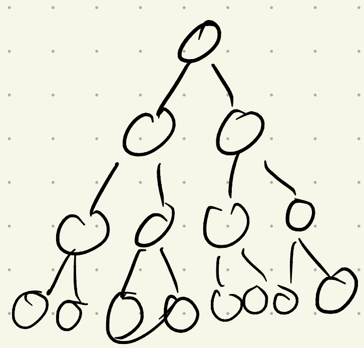
Let $w(v)$ = weight of v's subtree

$$s(v) = w(v) + s(v.left) + s(v.right)$$

Set $s(null) = 0$

Let $rank(v) = \lg(s(v))$

Note: if $s(v)=1$ for all v:



Potential function:

$$\Phi(T) = \sum_{v \in T} r(v)$$

$$= \sum_{v \in T} \lfloor \lg(s(v)) \rfloor$$

(useful later...)

Amortized cost:

$$\text{time} + \Phi' - \Phi$$

Access Lemma:

Amortized time to splay a binary tree T with root t at x is $\leq 3(r(t) - r(x)) + 1$

$$= O\left(\frac{\log s(t)}{\log s(x)}\right)$$

Restate:

Let $\left.\begin{aligned} r(x) &= \text{rank before} \\ r'(x) &= \text{rank after} \end{aligned}\right\}$ splay

rotation:

single

double