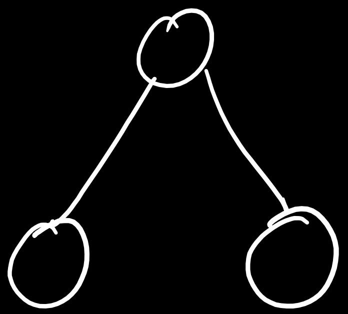
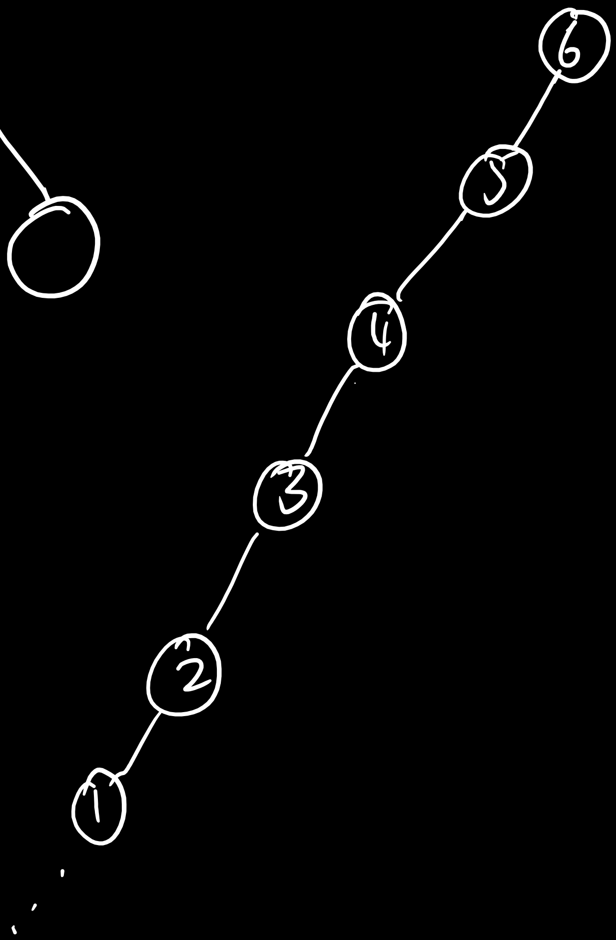
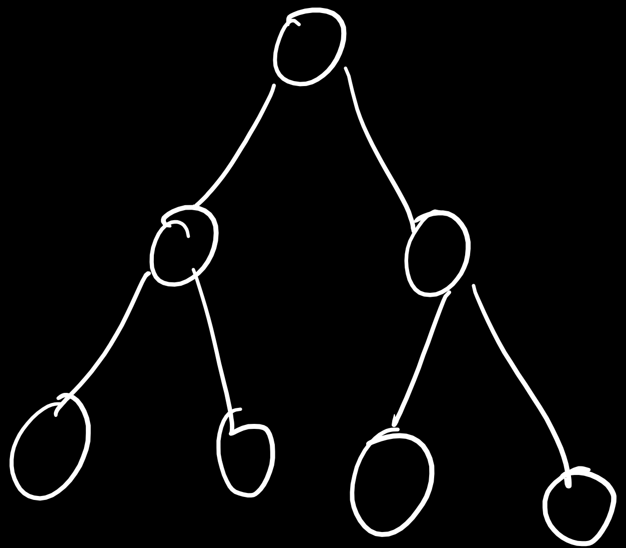
Lecture 4

Splay Trees!!

Binary Search Trees



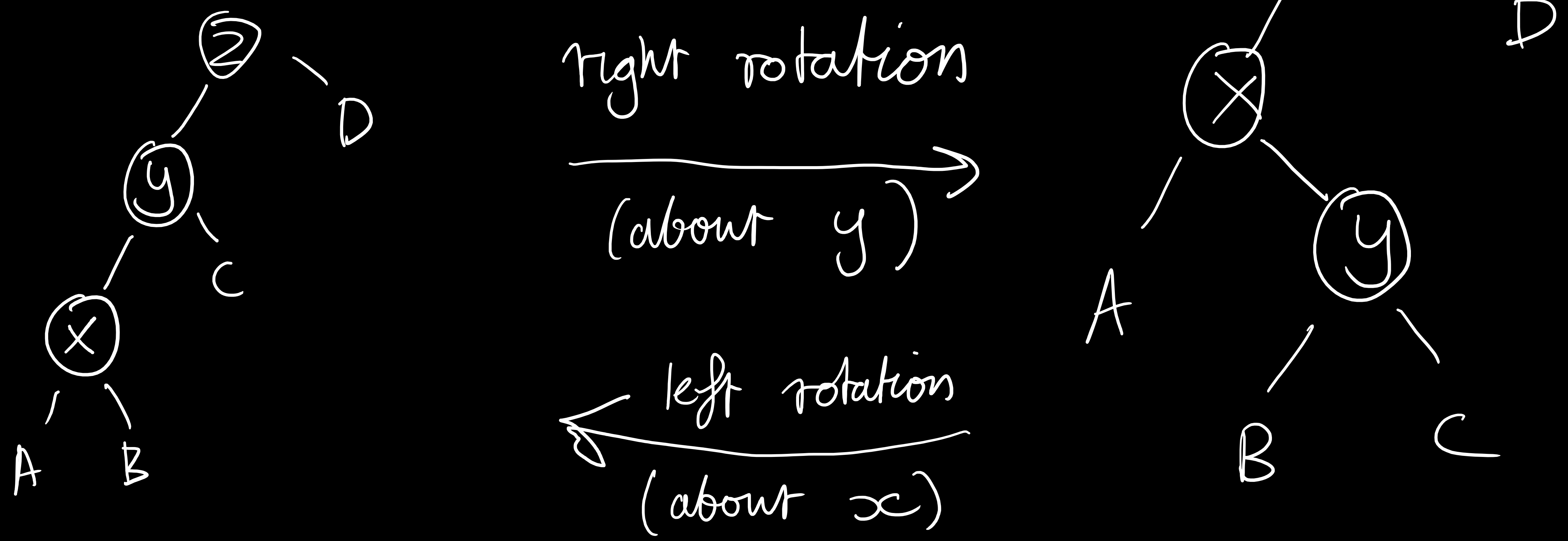




Several algorithms

AVL, Red-Black

Rotations


z
D
y
C
x
A
B
right rotation
(about y)
left rotation
(about x)
z
x
D
A
y
B
C

Locality

When we access a node, move it to the root

1

2

3

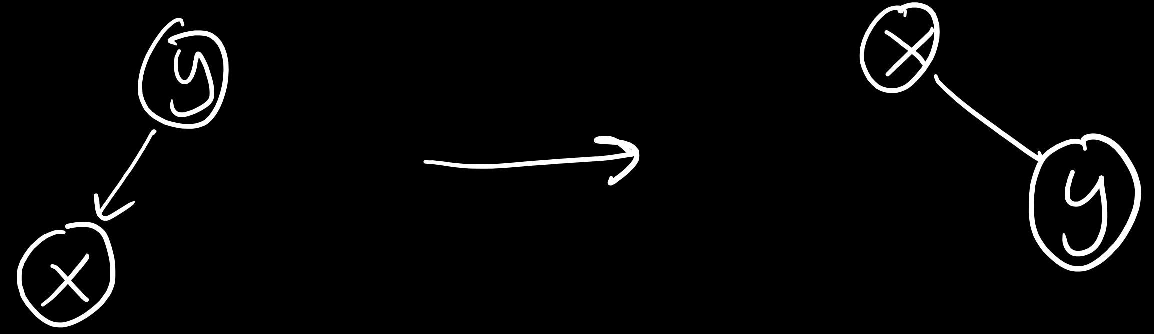
$\ddots$

n

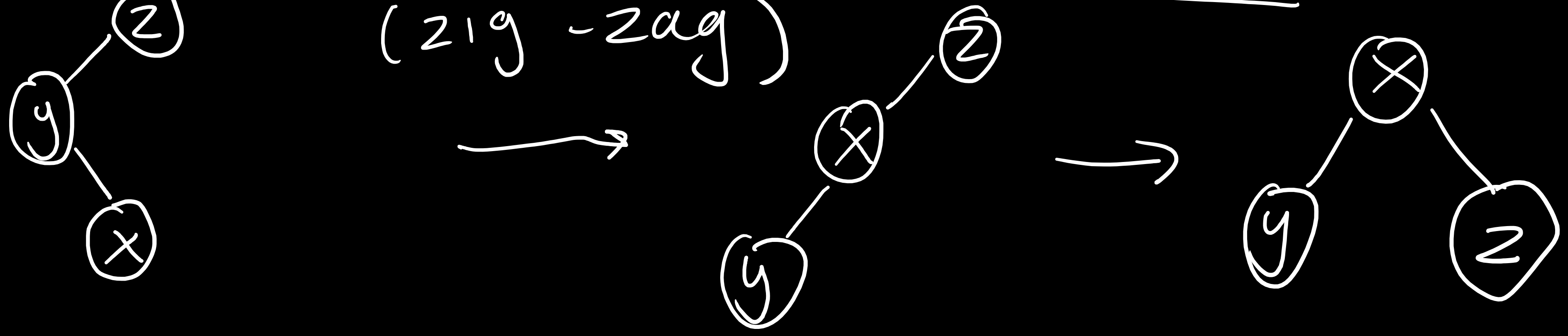
Naively rotating to the root doesn't work ($\Omega(n)$ time per operation)

Key Idea: Do rotations in pairs

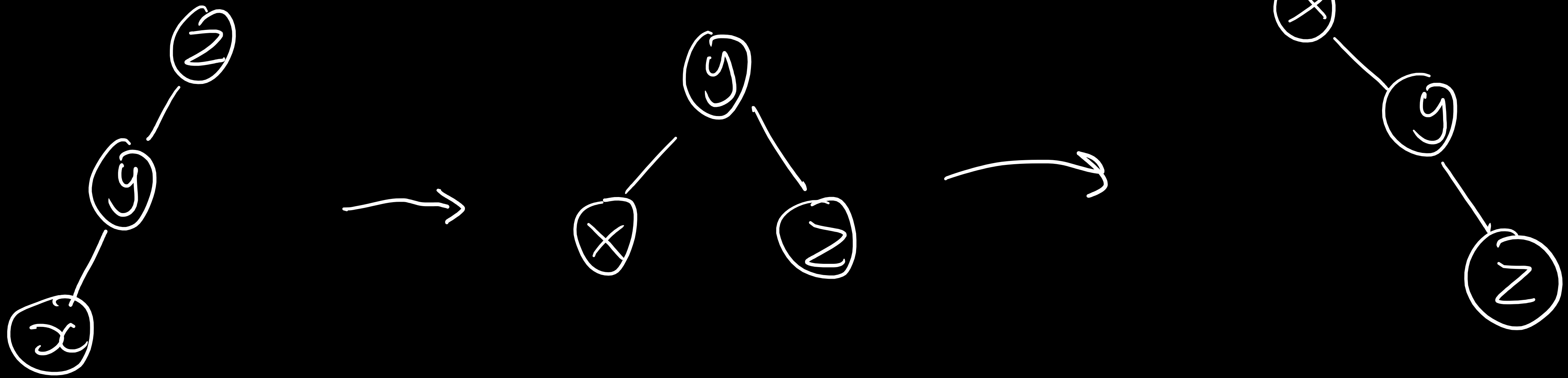
Three cases (zig rule)



(zig-zag)



(zig-zig)



splay(x)

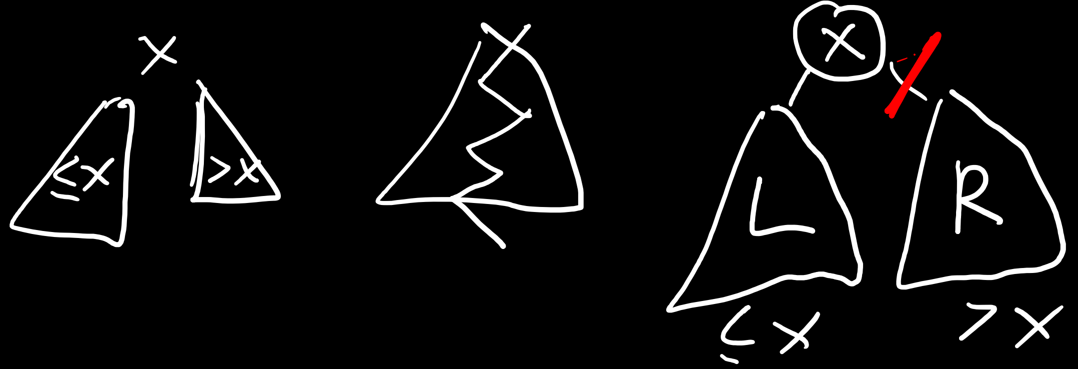
while x is not root

do one of those splay steps

Implement BST operations

Search Regular BST search alg, then splay the last element touched

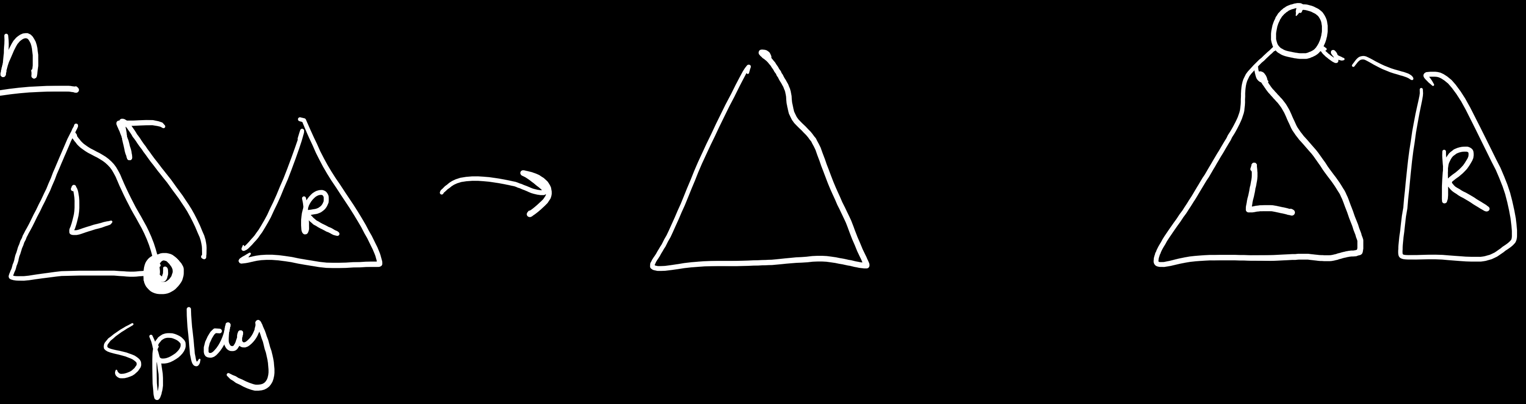
Split



splay(x)

chop right child

Join



splay max element in L
join R as the right subtree

Definitions

Nodes have weights $w(x)$

"size" of a node $\quad s(x) = \sum_{y \in T(x)} w(y)$

"rank" of a node $\quad r(x) = \lfloor \log_2 (s(x)) \rfloor$

Potential

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

Access Lemma

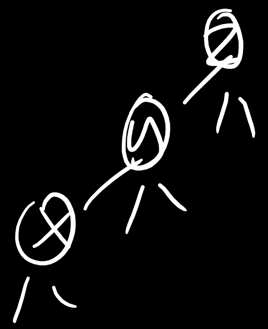
amortized # splay steps $\leq 3(r(t) - r(x)) + 1$

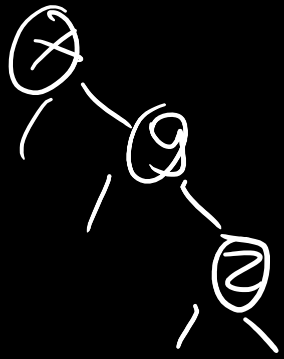
$(t = \text{root})$

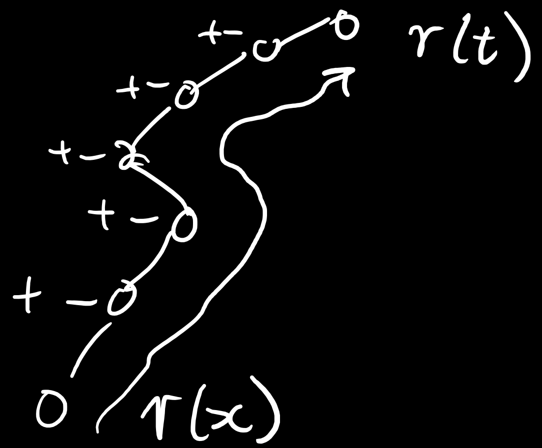
Cost of a step

cost of zig-zag or zig zig is $3(r(z) - r(x))$,

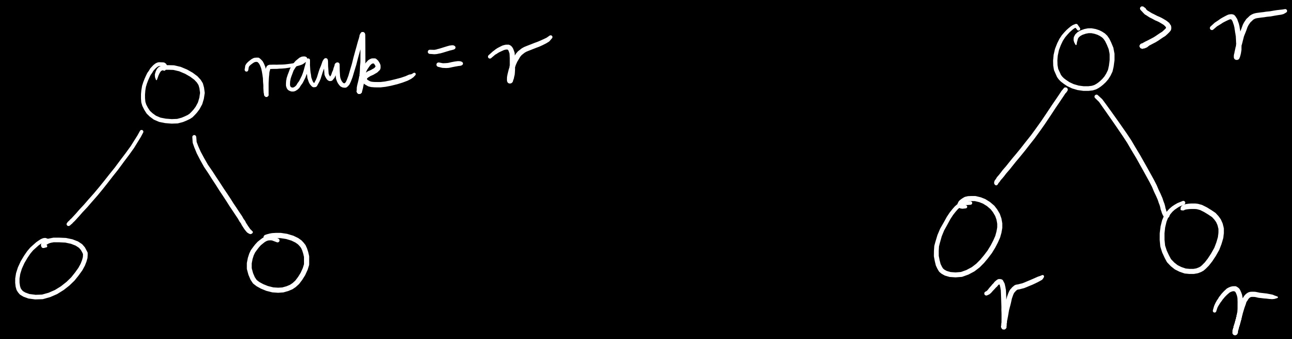
cost of zig is $3(r(y) - r(x)) + 1$



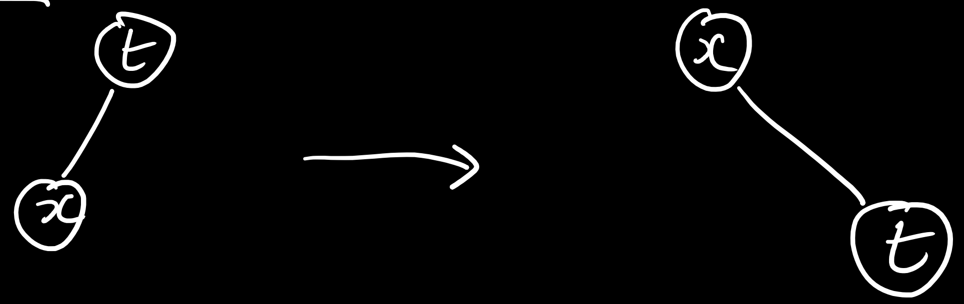




Rank Rule



Proof Zig:



Actual cost = 1

$r'(x) = r(t)$

$r'(t) \le r(t)$

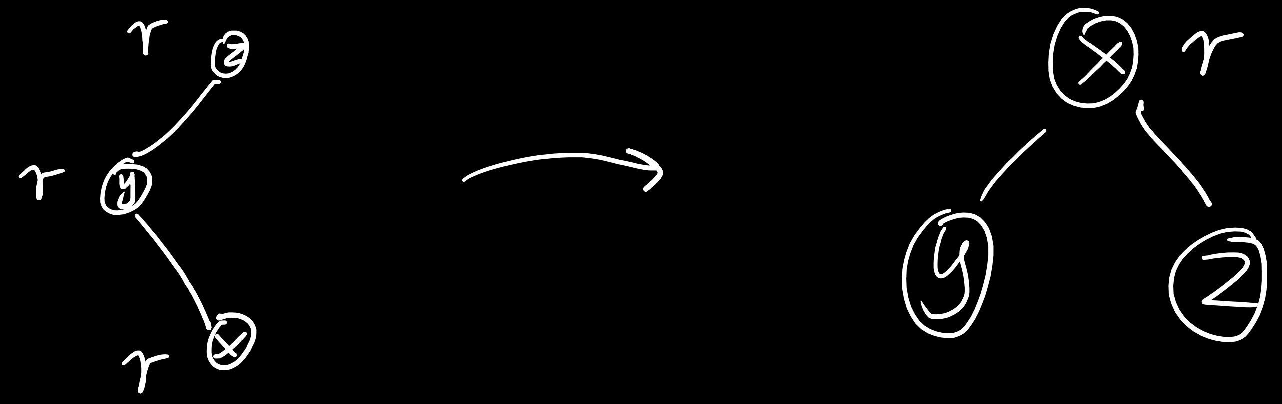
New $\Phi \le 2r(t)$ Old $\Phi = r(t) + r(x)$

$\Delta \Phi \le r(t) - r(x)$ $\quad ac = 1 + (r(t) - r(x))$

$\le 1 + 3(r(t) - r(x))$

Zig Zag

case 1



$r'(x) = r(z)$

$r'(x) + r'(y) + r'(z) < r(x) + r(y) + r(z) \Rightarrow \Delta \bar{\Phi}$ went down by ≥ 1

$ac \leq 1 - 1 \leq 0$

Balance Theorem

$ac \le 3(r(t) - r(x)) + 1$

ac splay $= O(\log n)$

(by settings to 1)

m splays in an n node tree, cost is

$$O(m \log n + n \log n)$$

proof: $m \log n + \underbrace{\Phi(T') - \Phi(T)}$

$= m \log n + n \log n$ $\square$