lecture 7 - Sep 18 [NOTE: shorter office hour on Sep 25] 12:20 - 1:20

1.4 AMORTIZED COST — SPLAY TREES

idea: simple binary search tree structure in which

- a single operation may take $\Theta(N)$ time in the worst case,
- but M operations in sequence take $O(M \log N)$ time

Definition 3. Let A be an algorithm and $f: \mathbb{N} \to \mathbb{R}^{\geq 0}$.
If a sequence of M applications of A has a running time of $O(M \cdot f(N))$ then we say the <u>amortized running time</u> of A is $O(f(N))$.

Example 13.

(1) Cost of the i^{th} op. in a sequence is i.

$\Rightarrow$ M operations cost $1 + 2 + 3 + \ldots + M = \sum_{i=1}^{M} i = \Theta(M^2)$

$\rightsquigarrow$ no proper amortization happening (still have linear time per op.)

(2) Cost of the 1st op. is N, 2nd op. $\frac{N}{2}$, 3rd op $\frac{N}{4}$, ...

$\Rightarrow$ M operations cost $N + \frac{N}{2} + \frac{N}{4} + \ldots + \frac{N}{2^{M-1}}$

$$= N \cdot \underbrace{\sum_{i=0}^{M-1} \frac{1}{2^i}}$$

$$= N \left(\underset{\substack{\uparrow \\ 1 \text{ pizza}}}{1} + \underbrace{\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots + \frac{1}{2^{M-1}}}_{< 1 \text{ pizza}} \right)$$

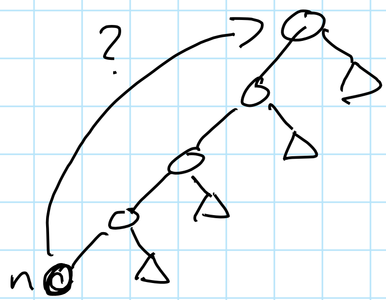
$$\leq 2 \cdot N$$

$\rightsquigarrow$ per op. cost $\leq \frac{2N}{M}$

for $M = N$: amortized cost constant!

<u>splay tree:</u> a BST data structure whose op.s have amortized running time $O(\log(N))$.

after a node n is accessed, it is pushed to the root



$\rightsquigarrow$ if n deep, accessing n is costly, but pushing n to the root will help balance the tree

$\rightsquigarrow$ If n is re-accessed, it is higher up in the tree and cheaper to access.

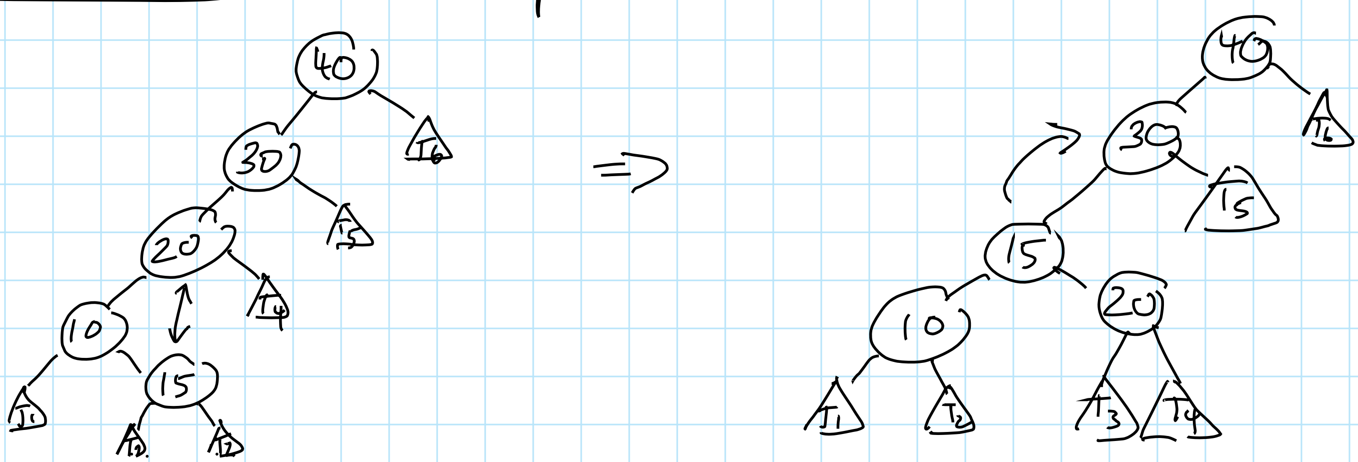
$\rightsquigarrow$ very useful in practice!

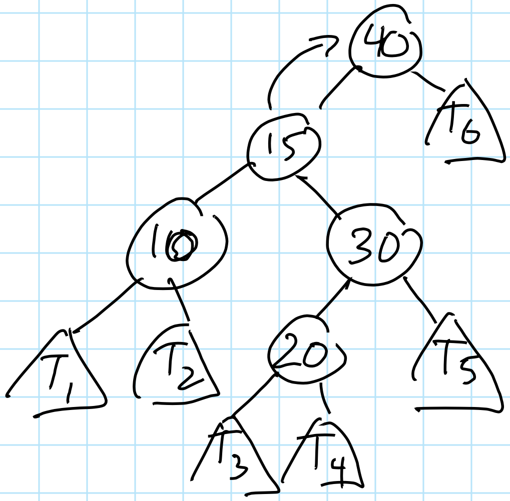
often if a data item has been accessed, it is very likely to soon be accessed again.

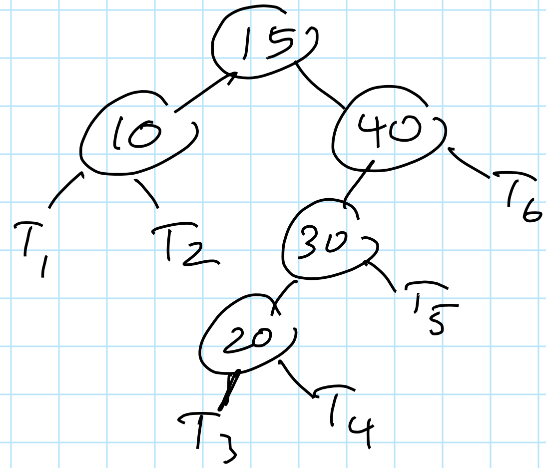
$\rightsquigarrow$ No balance factors need to be stored.

how to push last accessed node to the root?

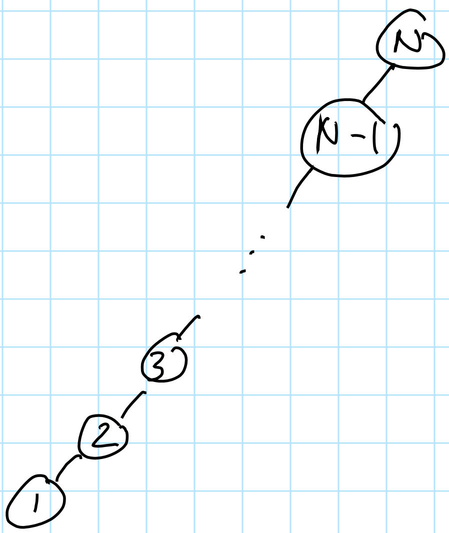
<u>NON-IDEA</u> Example 14. access 15







for this strategy, there is a sequence of M ops that needs $\Omega(M \cdot N)$ time:



access sequence:
1, 2, 3, 4, ..., N−1, N

length $M = N$

claim: this needs

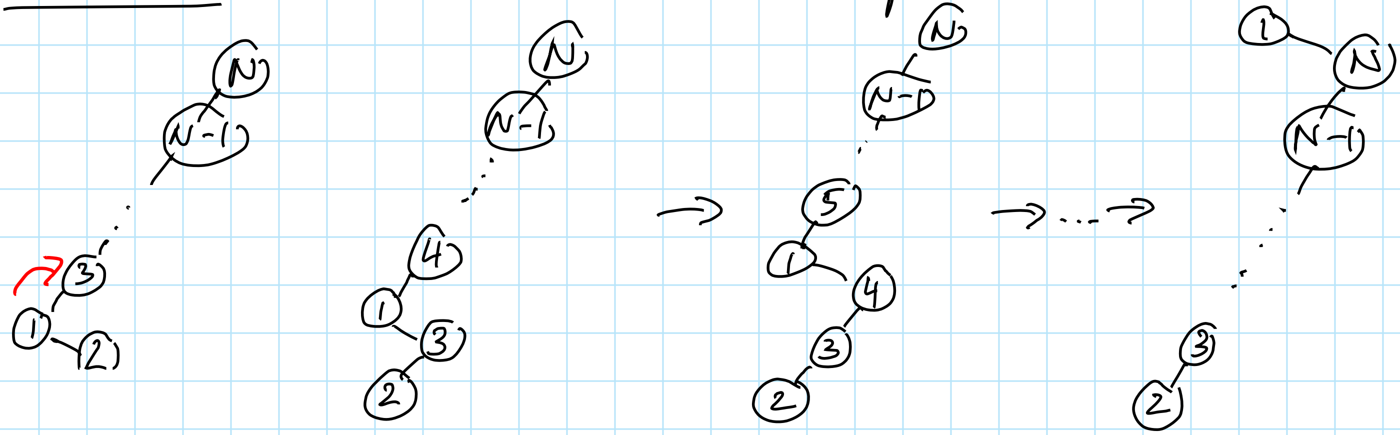
$\Omega(M \cdot N) = \Omega(N^2)$

time!

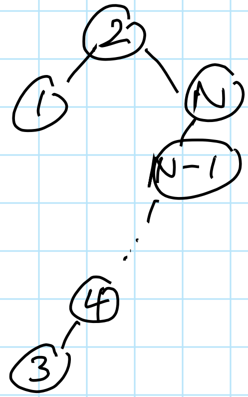
⇝ amortized cost is <u>not</u> logarithmic, but linear!

<u>access 1</u>: N units

then push 1 to root:

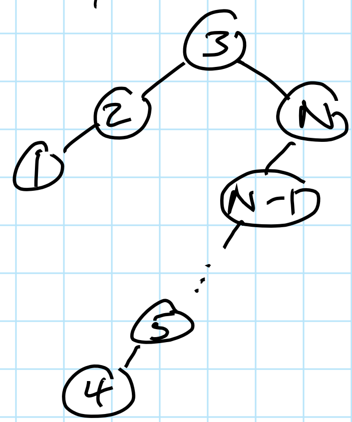


access 2: N units then push 2 to root



access 3 : N-1 units

push 3 to root



access 4 : N-2 units etc.

$\Rightarrow$ a total of $N + N + (N-1) + (N-2) + \ldots + 2$ units

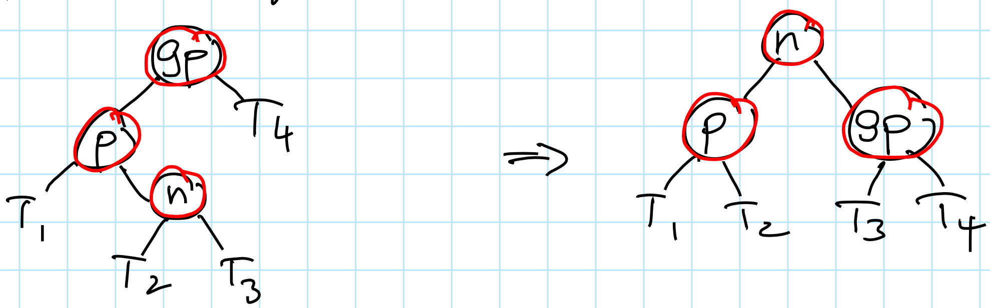
$$N + \sum_{i=2}^{N} i = \Theta(N^2)$$

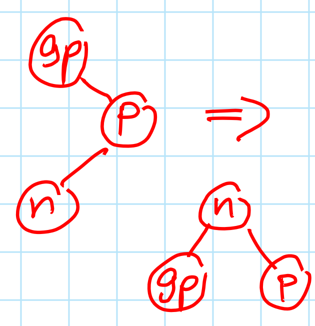
$\Rightarrow$ amortized cost for N ops. is $\Theta(N)$.

SPLAYING 3 cases (+3 symmetric version)

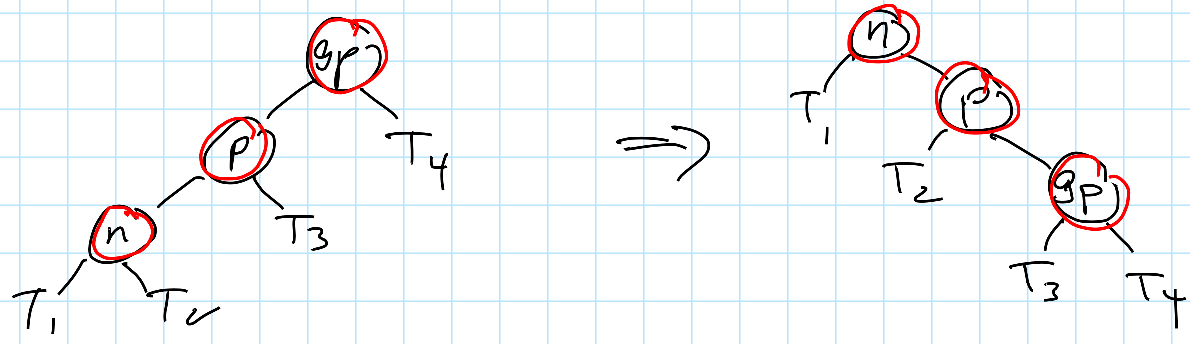
depending on the position of • the accessed node n
• its parent
• its grandparent

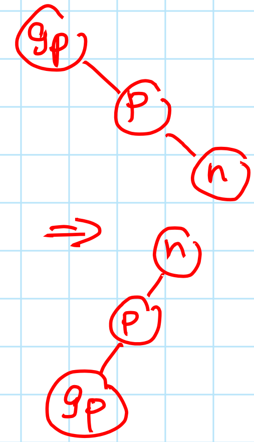
Case 1. zig-zag



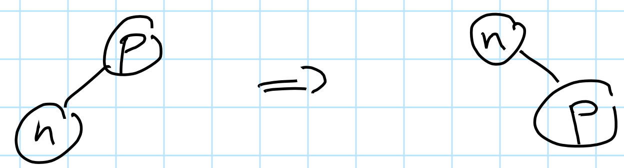


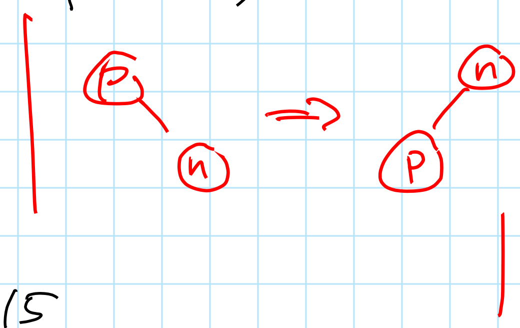
Case 2. zig-zig





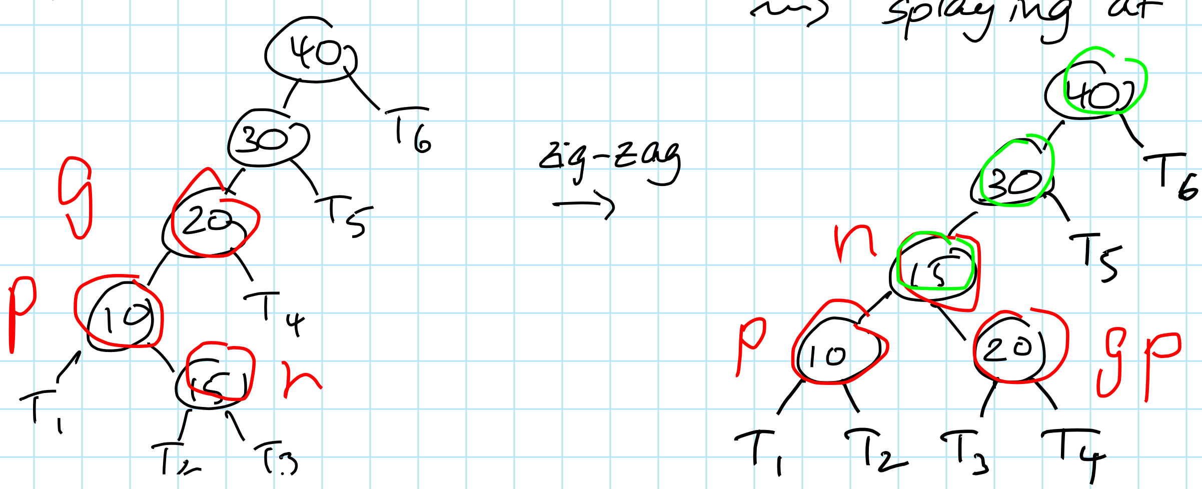
Case 3. zig no grandparent (p = root)





Example 15

access 15 ⇝ splaying at 15



zig-zig →