Lec 13 Balanced Binary Tree III Treaps

Treaps : - in treap ordering
- optionally must be a BST] Given both, treap unique

Distribution of tree shape

think picking pivot by assigning highest priority so far.

It's same as distribution of quicksort recursion tree

⇓

height of treap $\in O(\lg n)$ w.h.p.

Create RIV $A_j^i = \begin{cases} 1 & \text{if } S[i] \text{ is ancestor of } S[j] \text{ (inclusive)} \\ 0 & \text{else} \end{cases}$

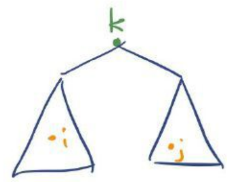
$$\text{depth}(j) = \sum_{i=0}^{n-1} A_j^i \qquad \text{size}(i) = \sum_{j=0}^{n-1} A_j^i$$

$$\mathbb{E}[A_j^i] = \mathbb{P}[\, S[i] \text{ is ancestor of } S[j] \,] = \frac{1}{|j-i|+1}$$

Intuition :

i and j are not ancestor of each other $\Leftrightarrow$ $\exists\, k$ s.t. $p_k > \begin{cases} p_i \\ p_j \end{cases}$

$\Rightarrow$

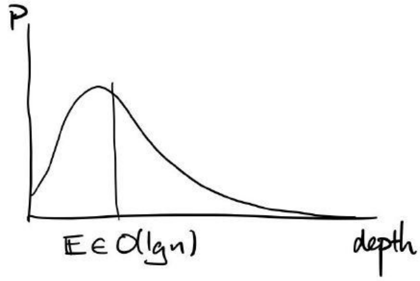


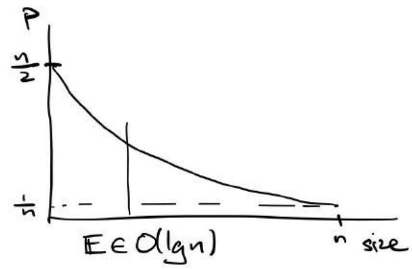
$$\mathbb{E}[\text{depth}(j)] = \sum_{i=0}^{n-1} \mathbb{E}[A_j^i] = \sum_{i=0}^{n-1} \frac{1}{|i-j|+1} = H_{j+1} + H_{n-j} - 1 \le 2H_n \le 2\ln n + O(1)$$

$$\mathbb{E}[\text{size}(i)] = \text{--- sth similar ---} \le 2\ln n + O(1)$$

Got to have depth$(j) \in O(\lg n)$ whp

BUT $\mathbb{E}[\text{size}(i)] \in O(\lg n) \not\Rightarrow \text{size}(i) \in O(\lg n)$ whp



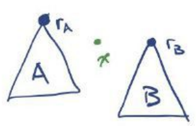


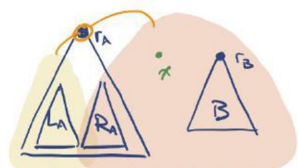
Treap Impl

track size

T = leaf | node of $T \times \mathbb{E} \times \mathbb{Z} \times T$

mkNode (A, x, B) ⇒ node (A, x, size A + size B + 1, B)
joinM (A, x, B) ⇒ case
  p x > pr A and p x > pr B ⇒ mkNode (A, x, B)
  pr A > pr B ⇒ case expose A of
    leaf ⇒ raise Absurd // we defined pr leaf = $-\infty$
    | Node (L_A, r_A, R_A) ⇒ mkNode (L_A, r_A, joinM(R_L, x, B)





$W \in O($ depth A + depth B $)$ ← each time we go down a level
$\subseteq O($ ln n   ln n $)$ n = depth A + depth B
$\subseteq O($ lg n $)$ ← also $O(\lg n)$ whp.

joinM preserves BST property if $A < x < B$
  preserves treap invar always

Table interface

Store key vals in tree, keep invariants by keys.
→ See documentation

Augmentation

Adding extra information in nodes (other than just balancing info)

Ex. dynamic paren matching

support: type paren = (|)
type dpm
insertAt dpm $\times$ paren $\times$ $\mathbb{Z}$ $\rightarrow$ dpm $O(\lg n)$
isMatched dpm $\rightarrow$ $\mathbb{B}$ $O(1)$?

$\rightarrow$ Keep track of unmatched left & unmatched right at every node

Reduced value augmentation

1. Associate tree T with associative func $f: \mathbb{E} \times \mathbb{E} \rightarrow \mathbb{E}$ and its identity I.

2. Modify T to keep the "sum" of f at each node

3. Modify joinM to maintain the "sum"

4. Add func reduceVal : $T \rightarrow \mathbb{E}$ that returns the sum at root

<u>Impl</u>

functor : $\mathbb{E} \times f \times I \rightarrow$ augT
  T = leaf | node of $T \times \mathbb{E} \times \mathbb{E} \times \mathbb{Z} \times T$
  reduceVal A = case expose A of leaf $\Rightarrow$ I
                  | node(_, _, s, _, _) $\Rightarrow$ s
  joinM (A, x, B) =
    node (L, x, f (x, f(reduceVal A, reduceVal B), size A + size B + 1, R)