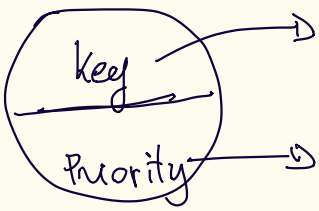
RANDOMIZED DS: DS that use random bits to improve their speed of operation

TREAP: tree + heap
↓
BST

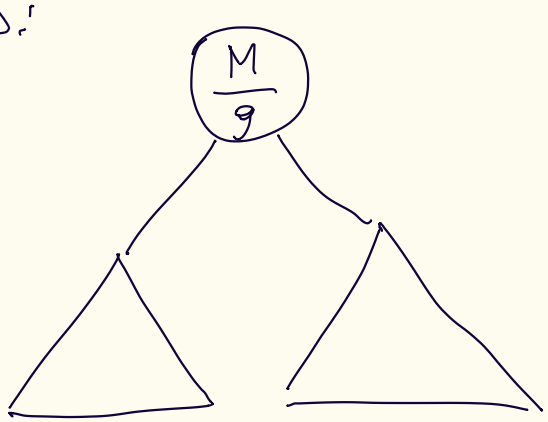


→ Binary search tree : letters

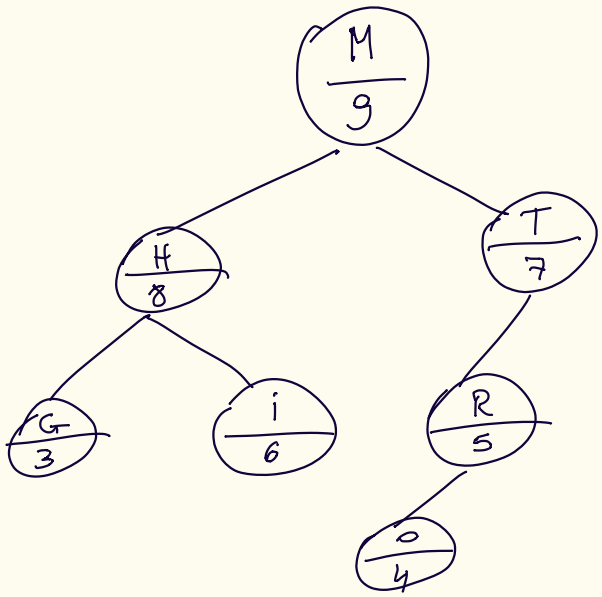
→ heap : numbers (Max / or min)

BST + MAX heap

es.:

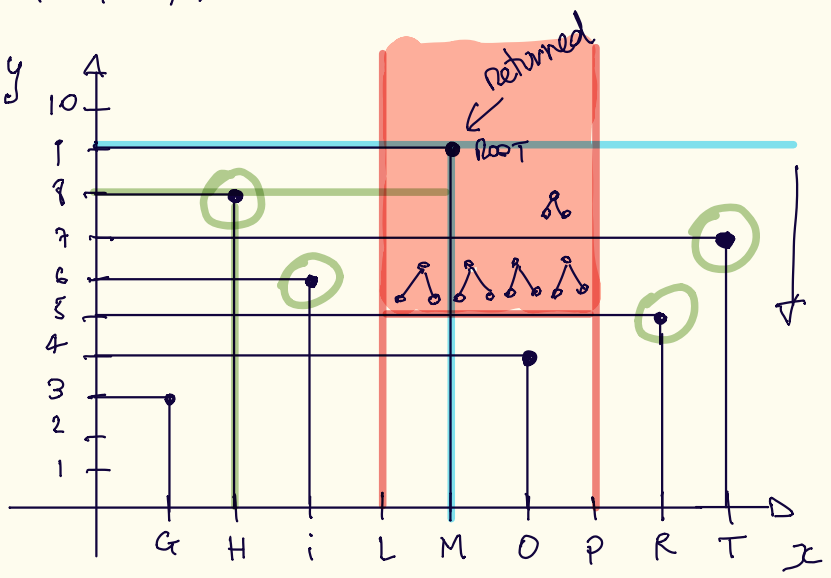


- priority tells to the key what is the level in the tree where it occurs



. recap geometric POV:

- Draw points on the cartesian plane: (H,9); (H,8); (T,7); (G,3); (i,6) (R,5); (O,4)



. SOLVES:

⇒ 3-SIDE RANGE QUERY

⇒ RECTANGLE formed by 3 sides (infinite in one direction, bounded in the other)

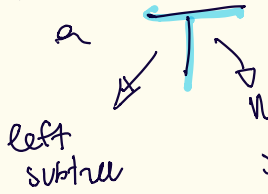
⇓

$[a,b] \times [c,\infty]$

$[a,b]$: x (bounded) $[c,\infty]$: y (not bounded)

ex: $[L,P] \times [5,\infty]$ RETURNS (M,9)

- Representation of the heap in the plane: ROOT (higher point) and draw a



- to the right & to the left

- work I waste: $h_l + h_r + 2 \cdot occ$

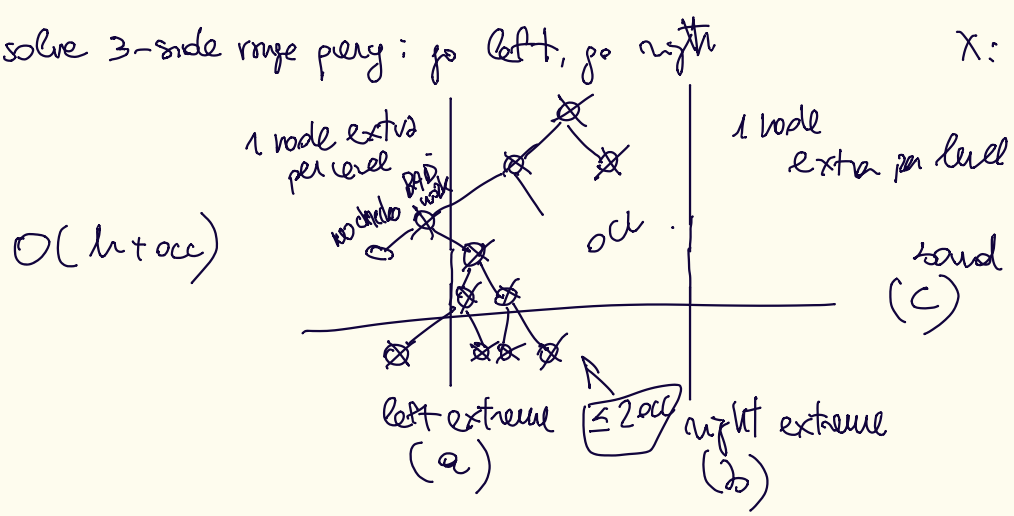
⇒ tbl cost $O(h + occ)$

↑ if the tree is balanced, is logarithmic in the # of points

Same explanation in different form

- solve 3-side range query: go left, go right

X: checked



- Total cost for visiting the tree: cost inside + cost spent to left + cost to right
(occ) (h) (h)

+ cost below
(2·occ)

$\Rightarrow \quad \leq occ + h + h + 2 \cdot occ = O(h + occ)$

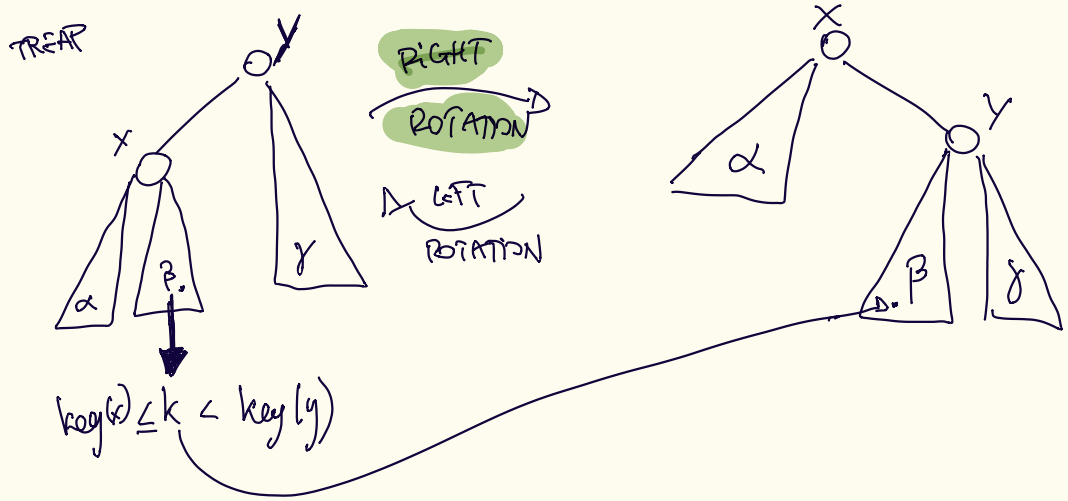
extra — optimal

but
if the heap is balanced becomes:

$O(\log_2 n + occ)$

$\log_2$ ↓ # of points

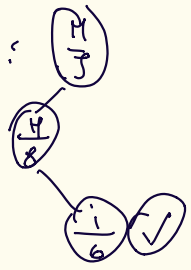
- TO BUILD IT I need: ROTATIONS

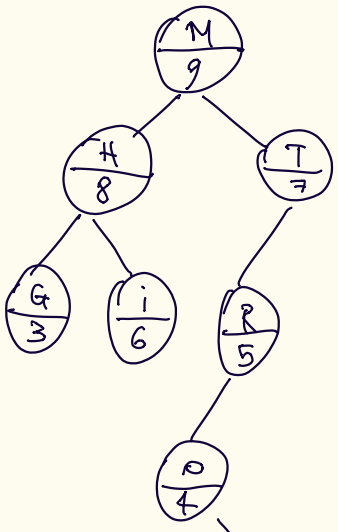


$O(h)$

- SEARCH (k): go left and right according to the key
(BINARY SEARCH TREE)

es. search (i):



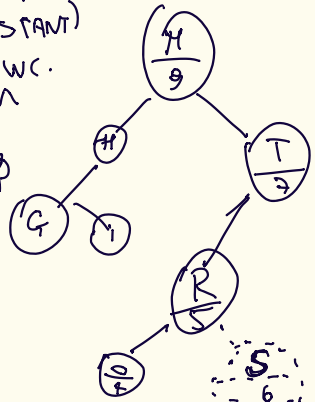


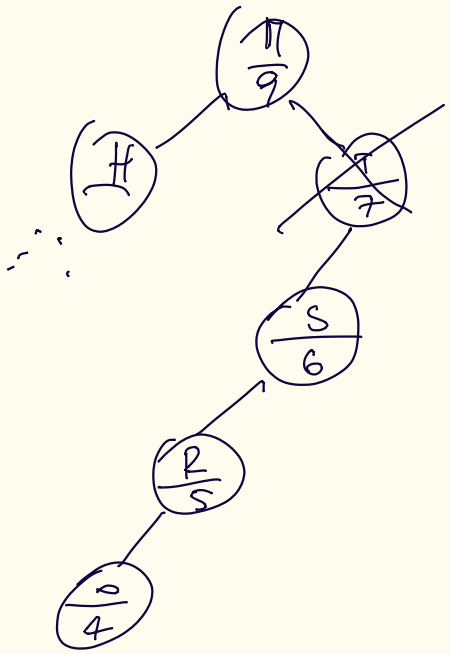
- INSERTION (key, priority) = search + ROTATION (COSTANT) in WC. $\Rightarrow O(h)$
$O(h)$ h

INSERT (S, 6):

=> search for the key s in the heap

- but is not an heap $5 < 6$
=> left rotation





- DELETE(k): force rotation putting priority like $+\infty$ $O(h)$
  - search(k)
  - set priority $= -\infty$
  - apply rotation to restablish the heap property
  - delete

- MERGE(T_1, T_2) ① $\forall x \in T_1$ & $\forall y \in T_2 \Rightarrow key(x) < key(y)$ ② $max(T_1) < \alpha < min(T_2)$

$O(h)$

CREATE a DUMMY root, logical key between $max(T_1)$ & $min(T_2)$ with priority $-\infty$

③ ROTATE to restablish condition

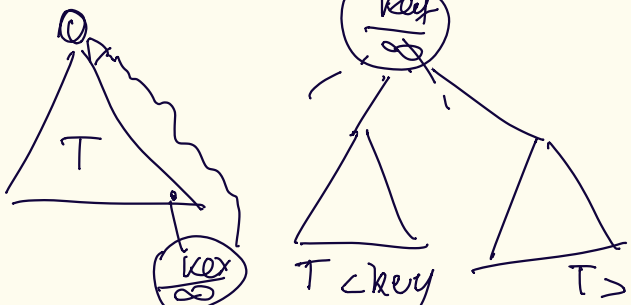
⇒ goes down to the leaf and remove it ④

dummy, doesn't exist

- SPLIT(k, T) → $T_1 < k$ → $T_2 > k$

$O(h)$

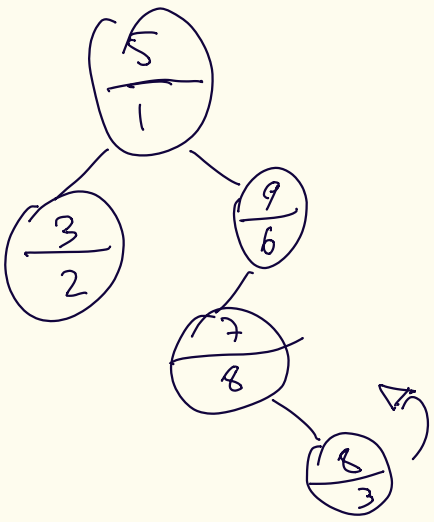
1) - insert(k, ∞)
2) - rotate
3) - delete root

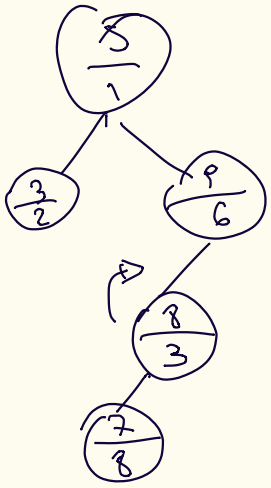


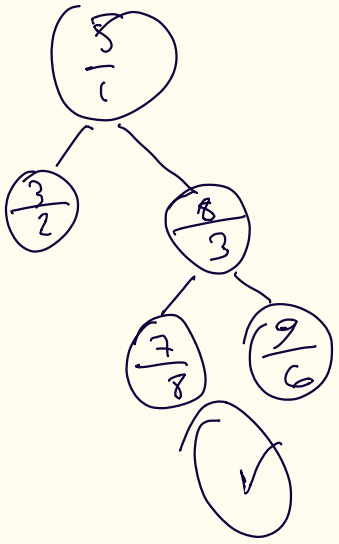
- min-treap <key, priority>

order: <5,1> <9,6> <3,2> <7,8>

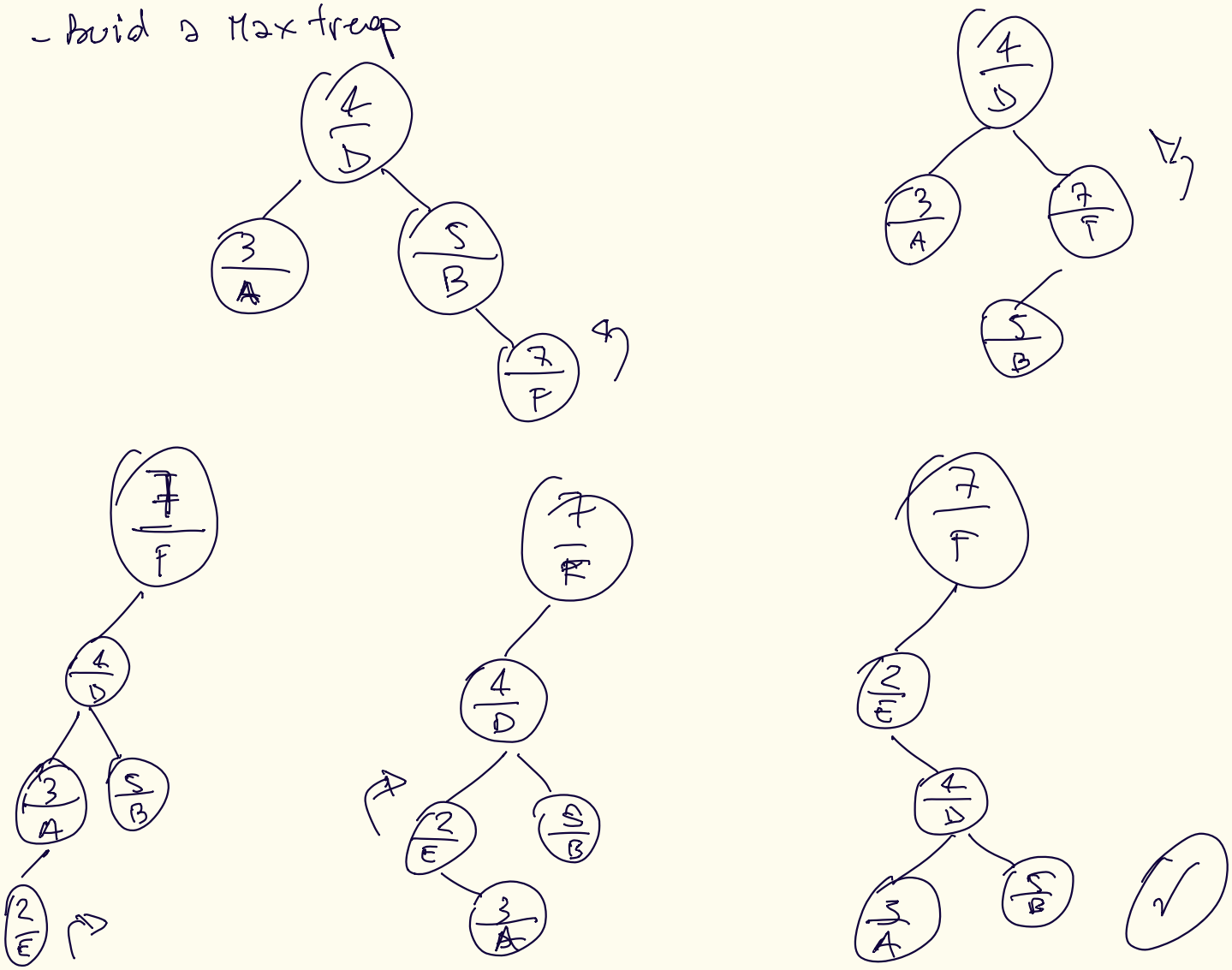
insert <8,3>



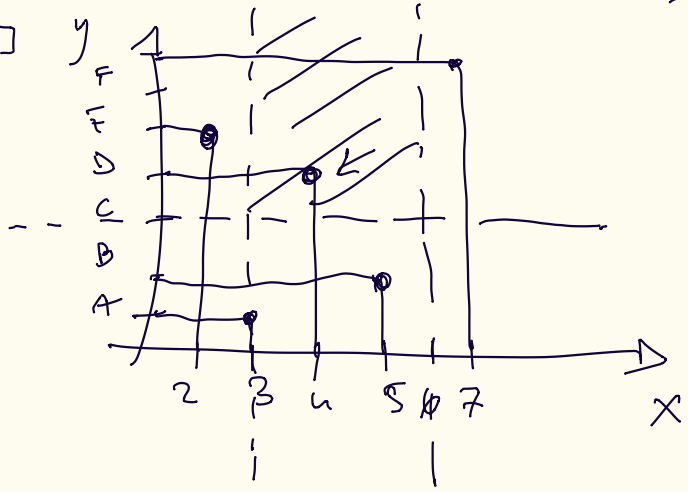




- Given the set S of pairs: $\{\langle 4,D\rangle \langle 5,B\rangle \langle 3,A\rangle \langle 7,F\rangle \langle 2,E\rangle\}$
  $\langle key, priority\rangle$
- Build a Max treap



- Solve the 3-Side range query $[3,6] \times [C, +\infty]$
  - to solve we visit the treap and stop when a node has priority $< C$, recurse left & right until a subtree includes keys in the range $[3,6]$



$\Rightarrow$ returns $\langle 4,D\rangle$

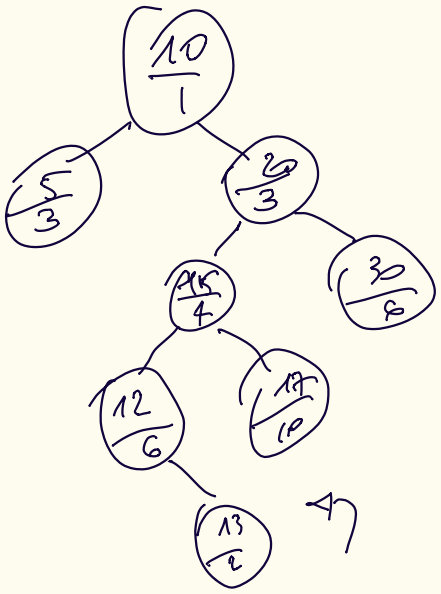
6. Construct a treap by inserting <10,1> <5,3> <20,3> <15,4>
<30,6> <12,6> <17,10>

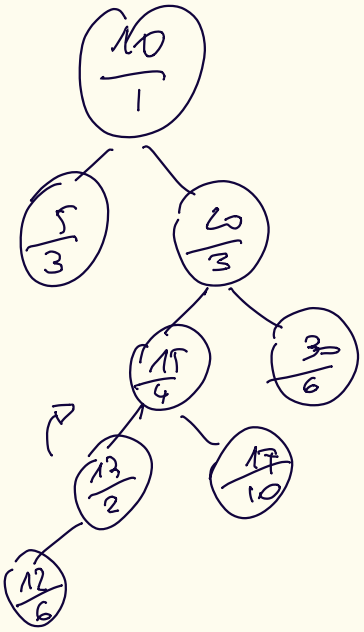
using a MIN-heap over y-coordinate (priority); x is the key

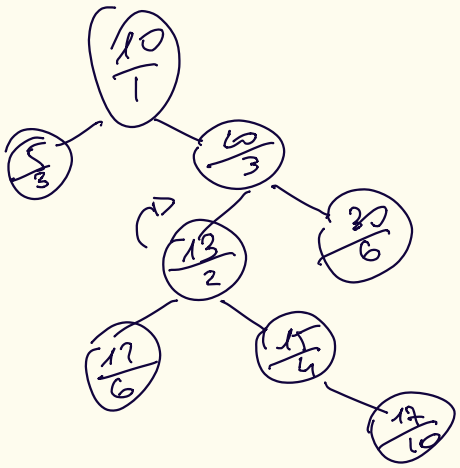
<key, priority>

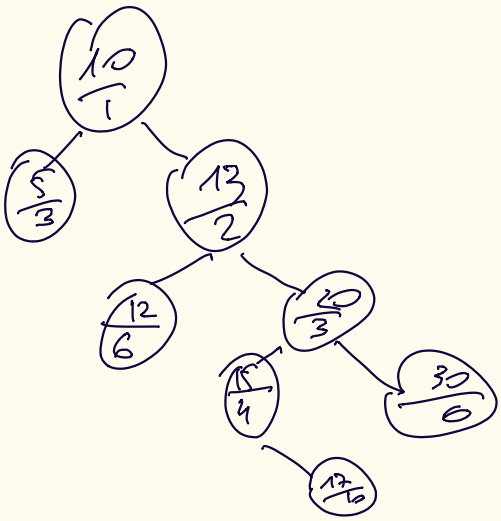
a) Show the final treap

b) Show the rotations by inserting <13,2>

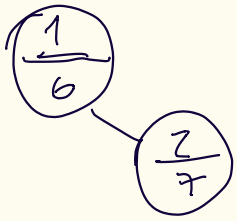


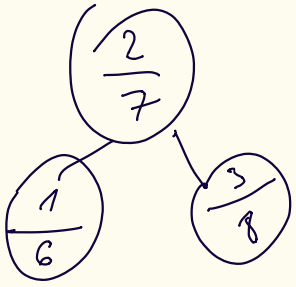


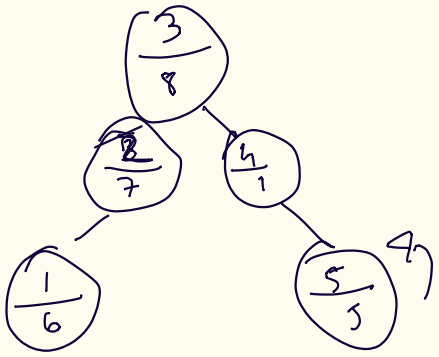


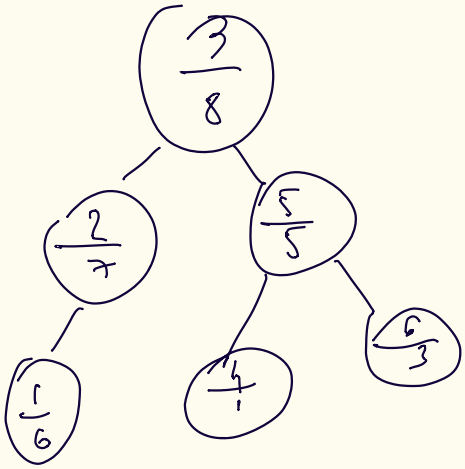


Build a MAX TREAP : (1,6) (2,7) (3,8) (4,1) (5,5) (6,3)





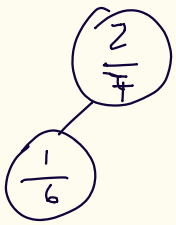


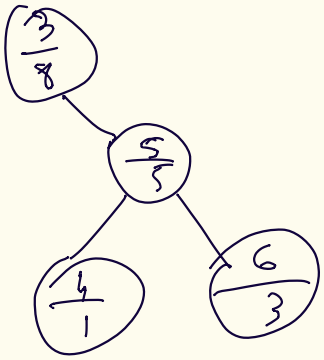


- show via splits, top and delete keys the x-component in range [3,6] and $y > 6$

- to determine points in $[3,6] \times [6,+\infty]$ we do:

  split $(T \leq 2,\ T > 2)$

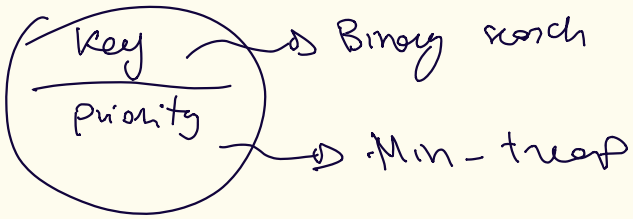


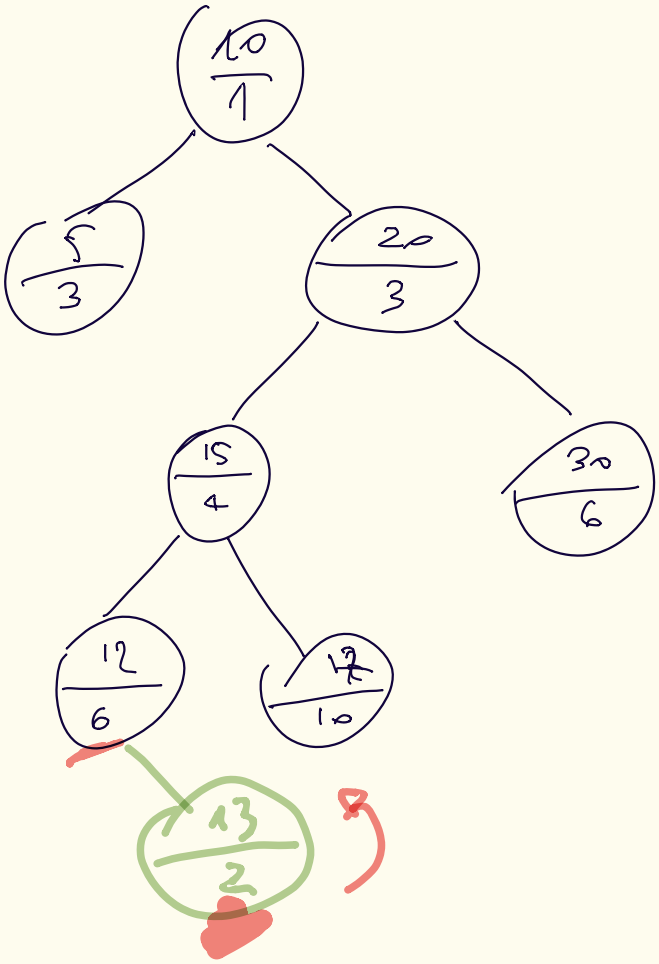


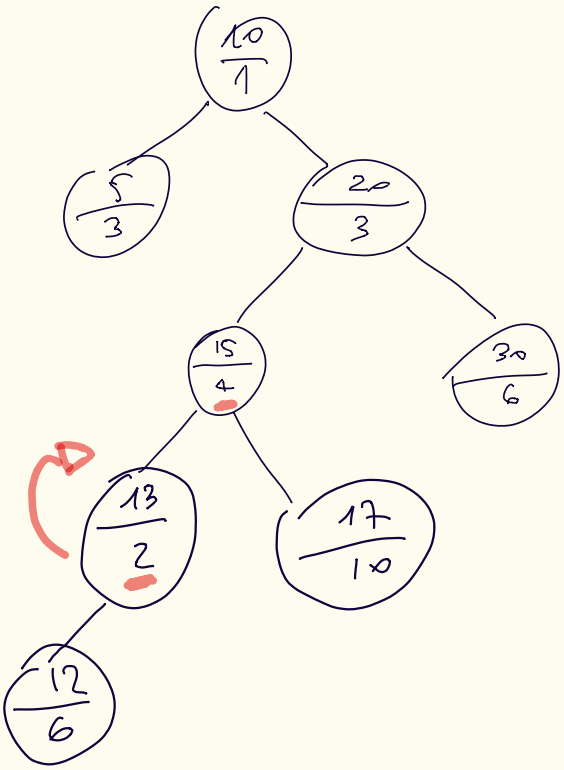
$T > 2, \leq 6$ $\Rightarrow$ (3,8)

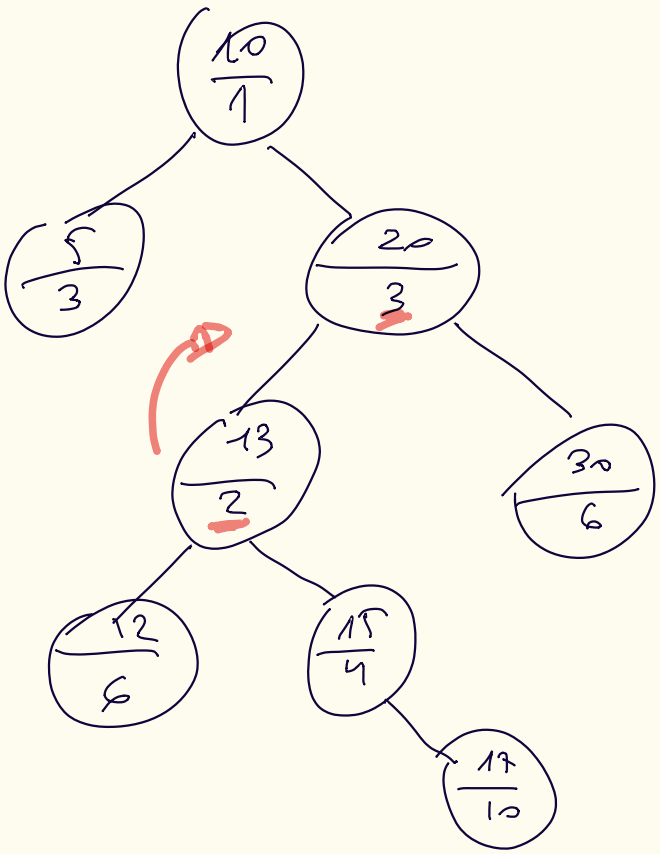
BUILD THE TREAP

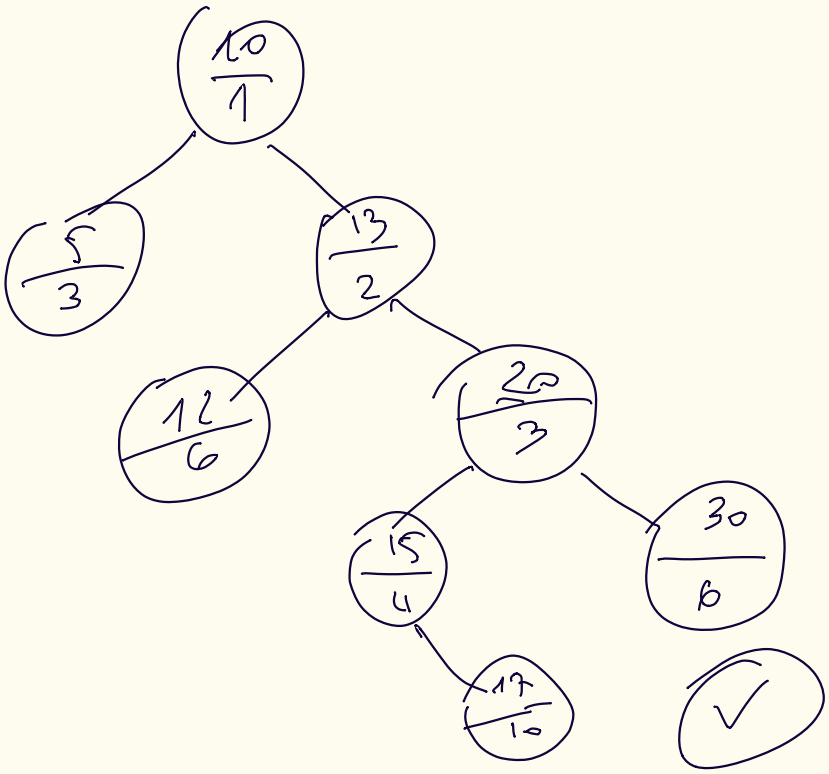
- Min-treap
- insert in the treap
  the key: <13, 2





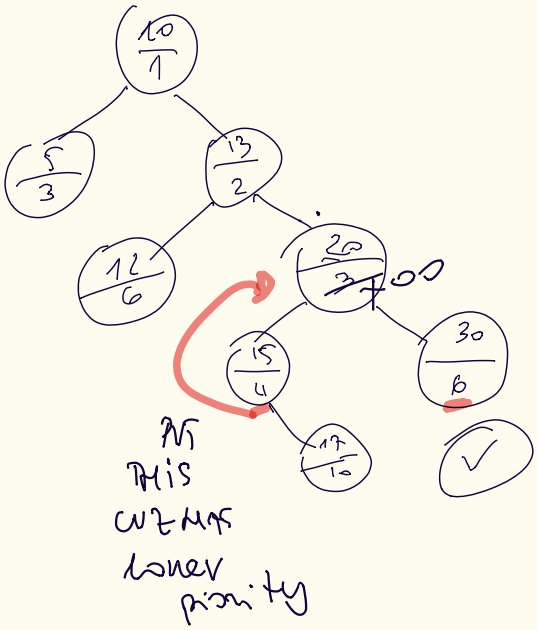


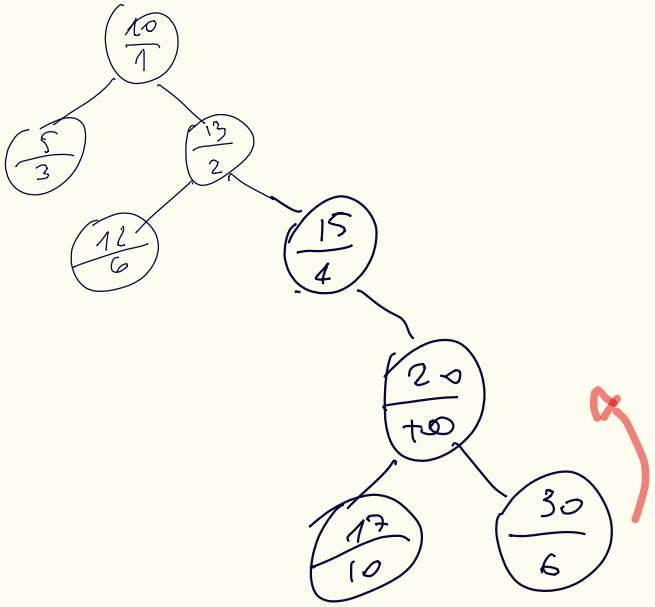


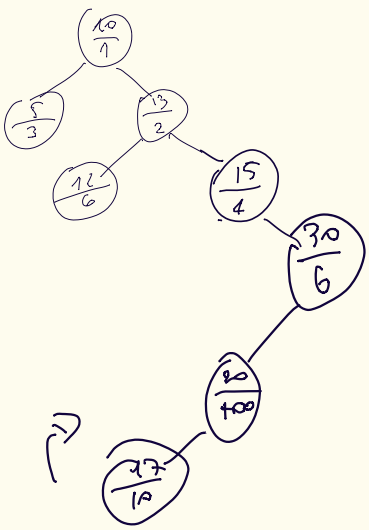


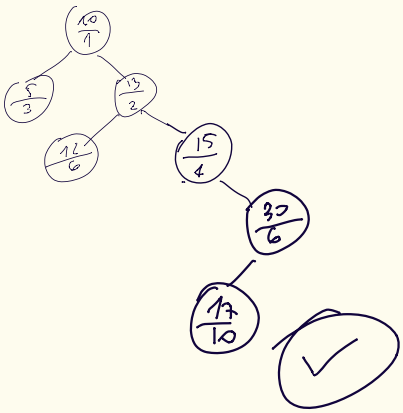
- Delete key: 20

  - set its priority of $\frac{20}{3}$ to $+\infty$ cuz is a min heap



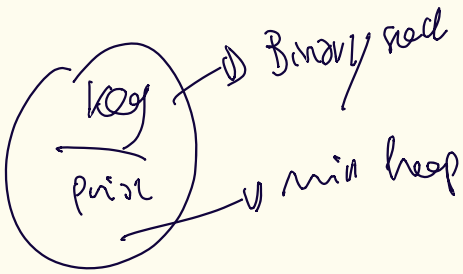




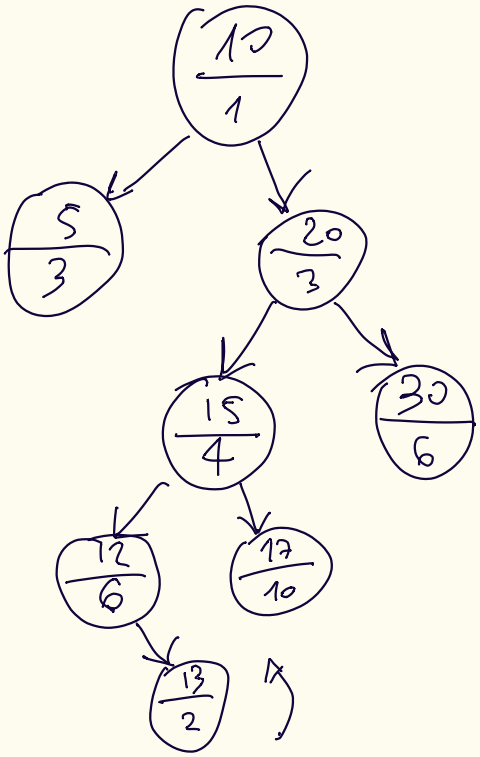


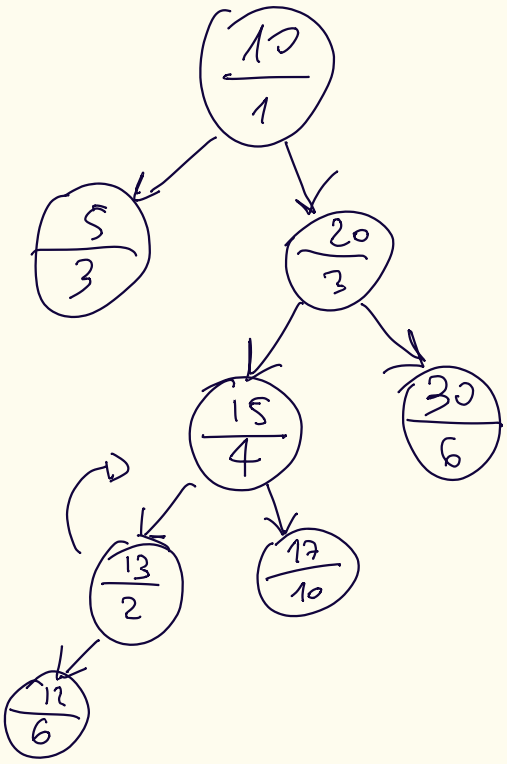
• CONSTRUCT TREAP <10,1> <5,3> <20,3> <15,4> <30,6> <12,6> <17,10>

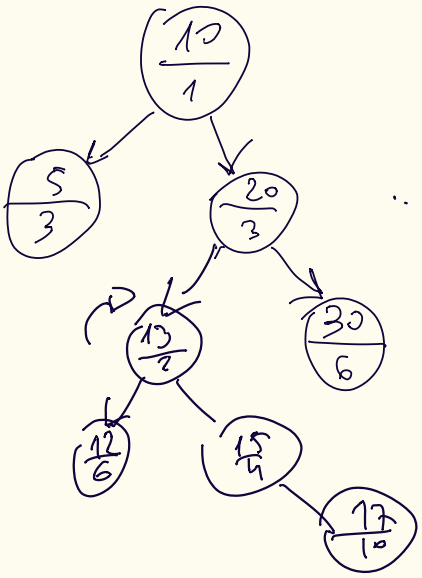
MIN HEAP

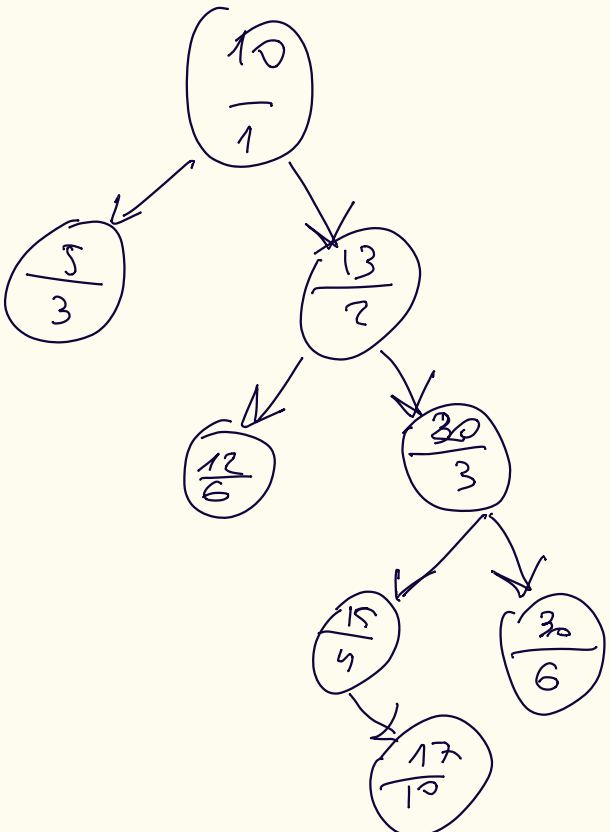


Insert <13,2>



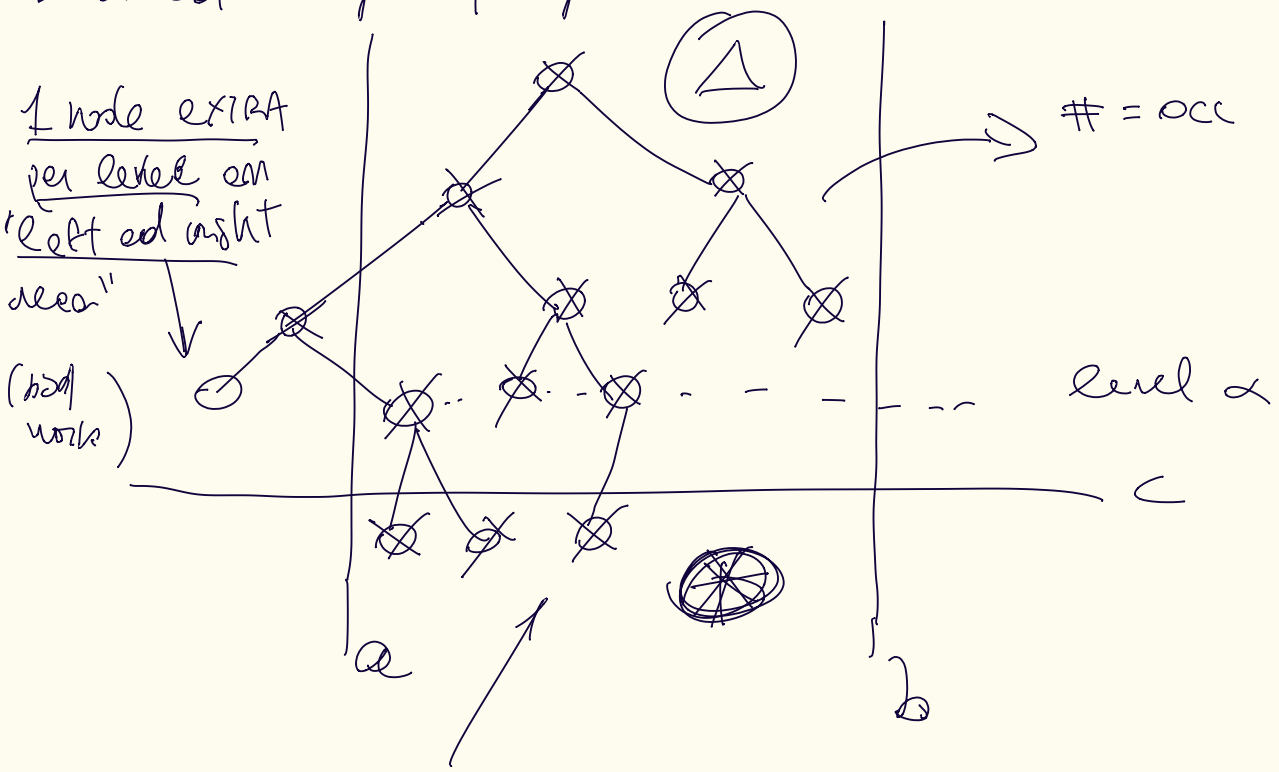






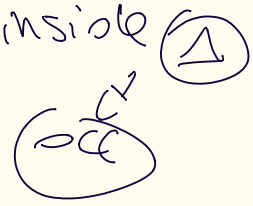
3 sided range query

X: checked



- in W.C. for each of node in level α I've 2 children
- For sure the node ~~reported~~ are < then node inside △

at most $2 \cdot occ$



Total cost of visiting tree

- cost inside = occ
- cost left = h
- cost right = h
- cost BELOW = $2\,occ$

$\Rightarrow O(h + occ)$

extra optimum

I'd to return it

but if MEAP is Balanced,

$\Rightarrow$ cost $O(\log_2 n + occ)$

$\#$ of points

- THEO : TREAP with random priorities has height $h = O(\log_2 n)$

PROOF:

- random variable $A_k^i = \begin{cases} 1 & \text{se } x_i \text{ è proper ancestor di } x_k \\ 0 & \text{altrimenti} \end{cases}$

- COMPUTE depth$(x_k) = \sum_{i=1}^{n} A_k^i$

di ancestors of x_k

- AVG DEPTH :

$$\mathbb{E}[\text{depth}(x_k)] = \mathbb{E}\left[\sum_{i=1}^{n} A_{ik}^i\right] = \sum_{i=n}^{n} \mathbb{E}[A_k^i] =$$

$$= \sum_{i=1}^{n} \mathbb{P}(A_k^i = 1)$$

- let $X(i,k)$ subset of treaps either $\{x_i, x_{i+1}, \dots, x_k\}$ or $\{x_k, x_{k+1}, \dots, x_i\}$

THEO : $\forall i \neq k$, x_i is a proper ancestor of x_k if x_i has the min priority in $X(i,k)$

DIM TEO

① x_i is the root => has smallest priority

② x_k is the root => x_i is ancestor, x_k has smallest priority

③ another x_j is ROOT => x_k and x_j are in different subtrees

$i < j < k \quad || \quad i > j > k$

$\Rightarrow$ x_i is not ancestor of x_k

↳ but has smallest priority

$\Rightarrow$ x_j HAS IT

④ if x_i and x_k are in the same subtree

$\Rightarrow$ THE THEOREM IS RECURSIVE

- Since each node in $X(i,k)$ is equally liked to have smallest priority:

$$P[A'_k = 1] = \begin{cases} \frac{1}{k-i+1} & \text{if } k > i \\ 0 & \text{if } k = i \\ \frac{1}{i-k+1} & \text{if } k < i \end{cases} =$$

$$= \sum_{i<k} \frac{1}{k-i+1} + \sum_{i>k} \frac{1}{i-k+1}$$

ARMONIC SUM $\leq 2\ln(n) - 2$

$O(\log_2 n)$

State and proof that treap with random priority has height $h = O(\lg_1 n)$

proof: $A_k^i = \begin{cases} 1 & \text{if } x_i \text{ is a proper ancestor of } x_k \\ 0 & \text{otherwise} \end{cases}$

- compute depth of x_k = depth(x_k) = # of ancestors

$$\text{depth}(x_k) = \sum_{i=1}^{n} A_k^i$$

$$E[\text{depth}(x_k)] = E\left[\sum_{i=1}^{n} A_k^i\right] = \sum_{i=1}^{n} E[A_k^i] =$$

$$= \sum_{i=1}^{n} P(x_i \text{ is a proper ancestor of } x_k) =$$

$$= \sum_{i=1}^{n} P[A_k^i = 1]$$

- let be $X(i,k)$ either $\{x_k, x_{k+1}, \dots, x_1\}$ or $\{x_i, x_{i+1}, \dots, x_n\}$

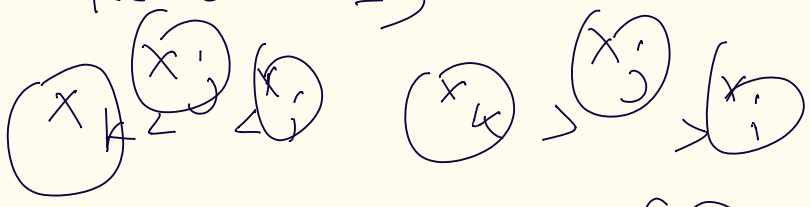
lema: (x_i) is a proper ancestor of (x_k) if it has the smallest priority in $X(i,k)$

(1) (x_i) is the root $\Rightarrow$ has smallest priority $\Rightarrow (x_i)$ proper ancestor of (x_k)

(2) (x_k) is the root $\Rightarrow (x_i)$ cannot be an ancestor of (x_k) and (x_k) has smallest

priority

(3) x_j is the root $\Rightarrow$



$\Rightarrow$ x_k and x_i are in different subtrees $\Rightarrow$ x_j has smallest priority

(4) x_i and x_j are in the same subtree

$\Rightarrow$ Recurse on the theorem

so

$$P(A^i_k = 1) = \begin{cases} \frac{1}{i-k+1} & \text{if } i > k \\ 0 & \text{if } i = k \\ \frac{1}{k-i+1} & \text{if } k > i \end{cases}$$

$$= \sum_{i \geq k} \frac{1}{i-k+1} + \sum_{k>i} \frac{1}{k-i+1}$$ harmonic sum $\approx 2\ln(n) - 2$

$\Rightarrow O(\log_2 n)$