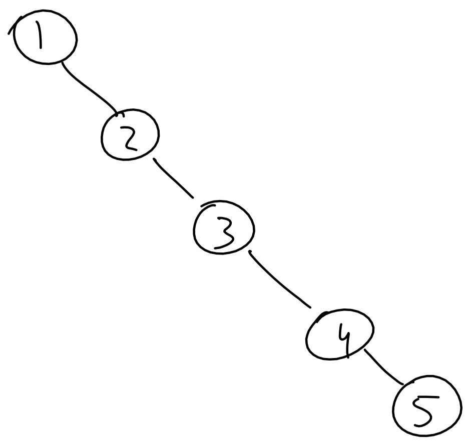
Randomized Search Trees

1, 2, 3, 4, 5



Treaps

- Binary Search Tree
- Balanced based on priority

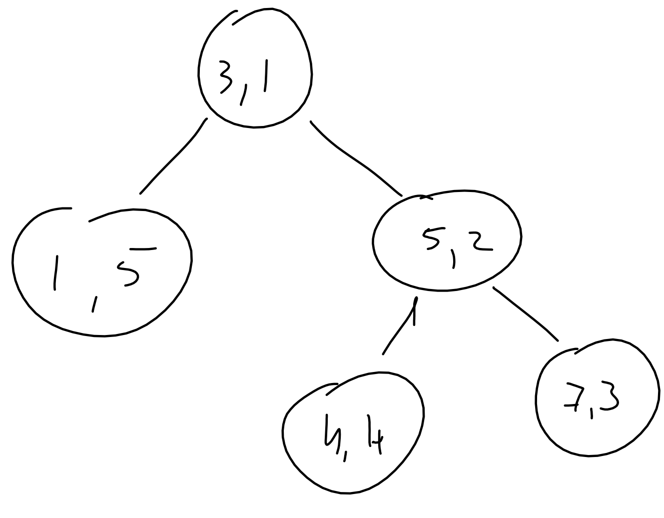
v.key

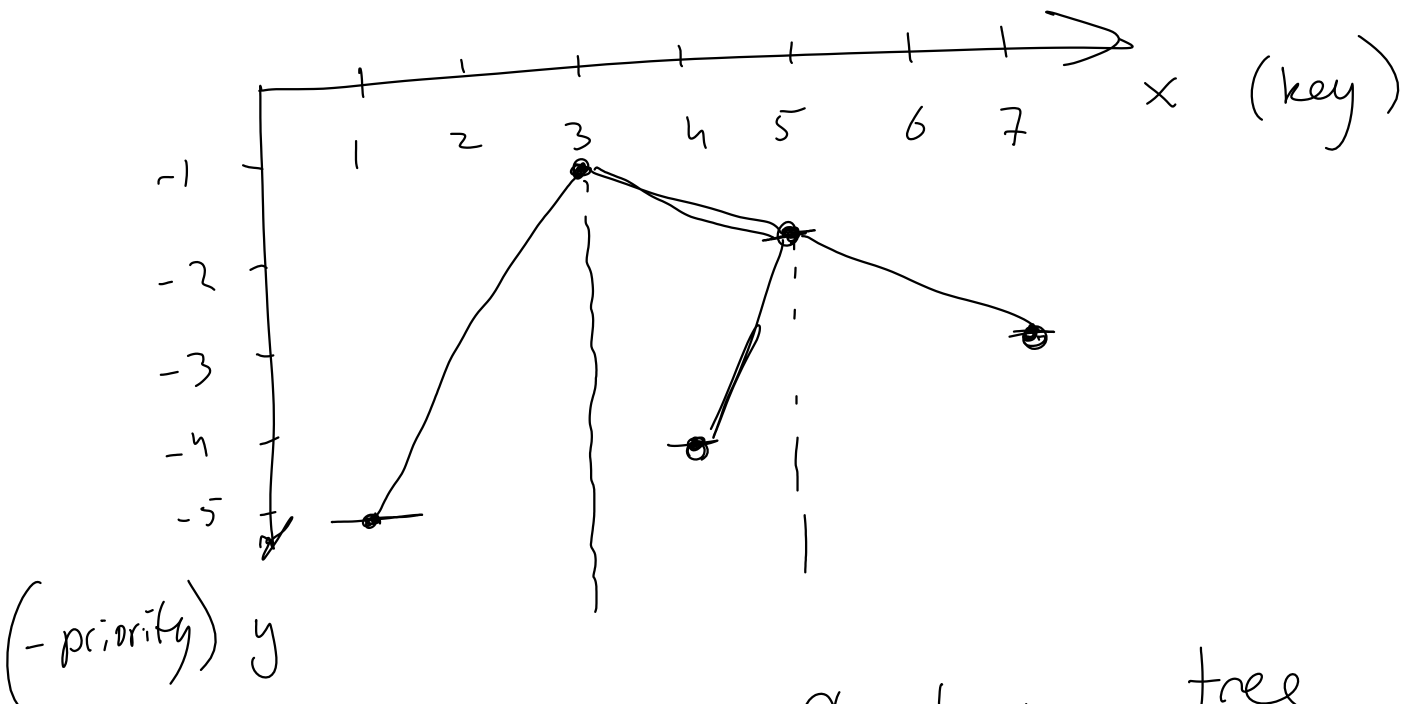
v.priority

assigned uniformly at random

key priority

(1, 5) (7, 3) (5, 2) (3, 1) (4, 4)

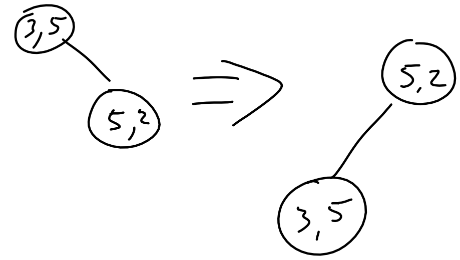




Cartesian tree

(used in geometric algorithms)

BST-Insert (v)



Treap Insert (v)

BST-Insert(v)

while (v.parent ≠ nil & v.priority < v.parent.priority)

rotate(v)

Treap Delete (v)

Treap-Increase-Priority (v, +∞)

Delete (v)

Treap-Increase Priority (v, new-Priority)

v.priority = new-priority

if <u>v.left ≠ nil</u> & (v.right ≠ nil & v.right.priority > v.left.priority)

min = v.left

else

min = v.right

while min $\neq$ nil
    rotate (min)
    if v.left $\neq$ nil & (v.right $\neq$ nil & v.right.priority > v.lef.priority)
        min = v.left
    else
        min = v.right

Treap-Split (T, key)
    v = new TreapNode (key, $-\infty$)
    Treap-Insert (T, v)
    $T_<$ = v.left
    $T_>$ = v.right
    Delete (v)
    return ($T_<$, $T_>$)

Treap Merge ($\underline{T_<}$, $\underline{T_>}$)

v = new TreapNode $\left(\frac{\max(T_<) + \min(T_>)}{2}, -\infty \right)$

v.left = $T_<$.root

v.right = $T_>$.root

Treap-Delete (v)

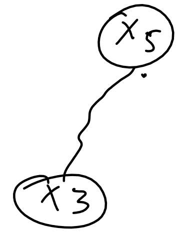
Treap Analysis

Runtime of an operation on node v
$O(\text{depth}(v))$

$x_k \leftarrow k^{th}$ smallest key in treap

Let $Y_{ij} = I\{\underline{x_i \text{ is a proper ancestor of } x_j}\}$

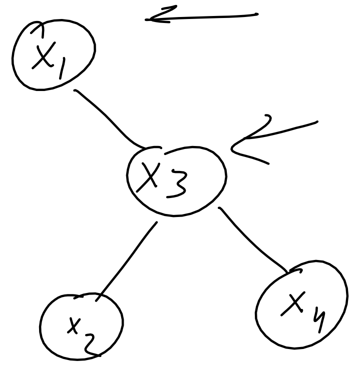
$Y_{ii} = 0$



$Y_{53} = 1$

$Y_{35} = 0$

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



$$\text{depth}(x_3) = Y_{13} + Y_{23} + Y_{33} + Y_{43}$$

$$1 + 0 + 0 + 0 = 1$$

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

Linearity of Expectations

$$E[A+B] = E[A] + E[B]$$

$$= \sum_{i=1}^{n} E[Y_{ik}]$$

$$= \sum_{i=1}^{n} \Pr\{Y_{ik} = 1\}$$

$$E[A] = \sum_{a} a \cdot \Pr[A=a]$$

If A is an i.r.v.

$$\Rightarrow E[A] = \cancel{0 \cdot \Pr[A=0]} + 1 \cdot \Pr[A=1] = \Pr[A=1]$$

$$X(i,j) = \begin{cases} \{x_i, x_{i+1}, \ldots, x_j\} & \text{if } i \le j \\ \{x_j, x_{j+1}, \ldots, x_i\} & \text{if } j \le i \end{cases}$$

Lemma: x_i is a proper ancestor of x_j in a treap T of size n, $(Y_{ij} = 1)$

iff x_i has smallest priority among $X(i,j)$

Proof: Cases:

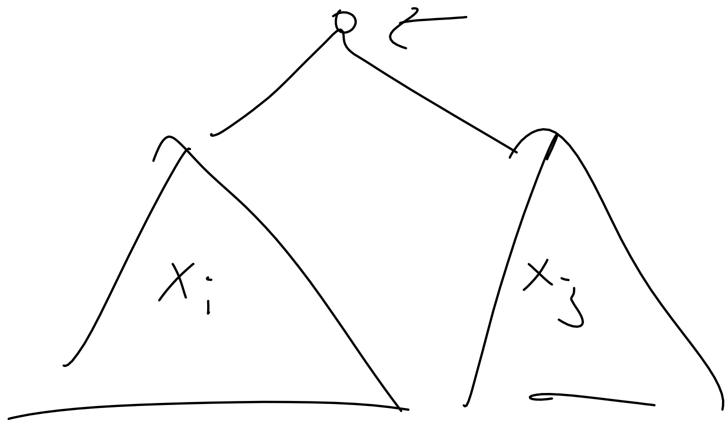
1) x_i is the root of Treap T

$\Rightarrow$ x_i has smallest priority among keys in T & it's a proper ancestor of x_j

2) x_j is the root of T

$\Rightarrow$ x_i doesn't have smallest priority & x_i is not ancestor of x_j

3) x_i & x_j are in separate subtrees of T



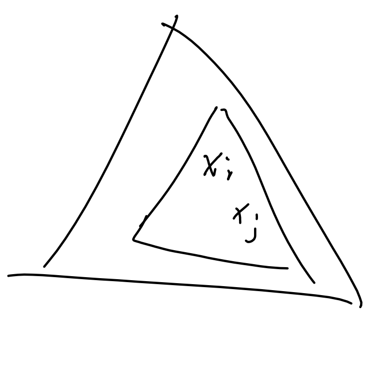
$\Rightarrow$ There exists key k s.t.

$x_i < k < x_j$ or $x_j < k < x_i$

x_i is not a proper ancestor of x_j

& priority of x_k is smaller than x_i

4) x_i & x_j are in the same subtree of T



$\Rightarrow$ By Induction the claim is true

$$\underline{\Pr[Y_{ik} = 1]} \quad ?$$

$\iff \Pr[x_i$ has smallest priority among $X(i,k)]$

$$\Pr[Y_{ik} = 1] = \frac{1}{|i-k|+1}$$

$$E[\text{depth}(\cancel{x}_k)] = \sum_{i=1}^{n} \Pr[Y_{ik} = 1]$$

$$= \sum_{i=1,\ i \neq k}^{n} \frac{1}{|i-k|+1}$$

$$= \sum_{i=1}^{k-1} \frac{1}{k-i+1} + \cancel{\frac{1}{k-k+1}} + \sum_{i=k+1}^{n} \frac{1}{i-k+1}$$

$$= \sum_{i'=2}^{k} \frac{1}{i'} + \cancel{1} + \sum_{i'=2}^{n-k+1} \frac{1}{i'}$$

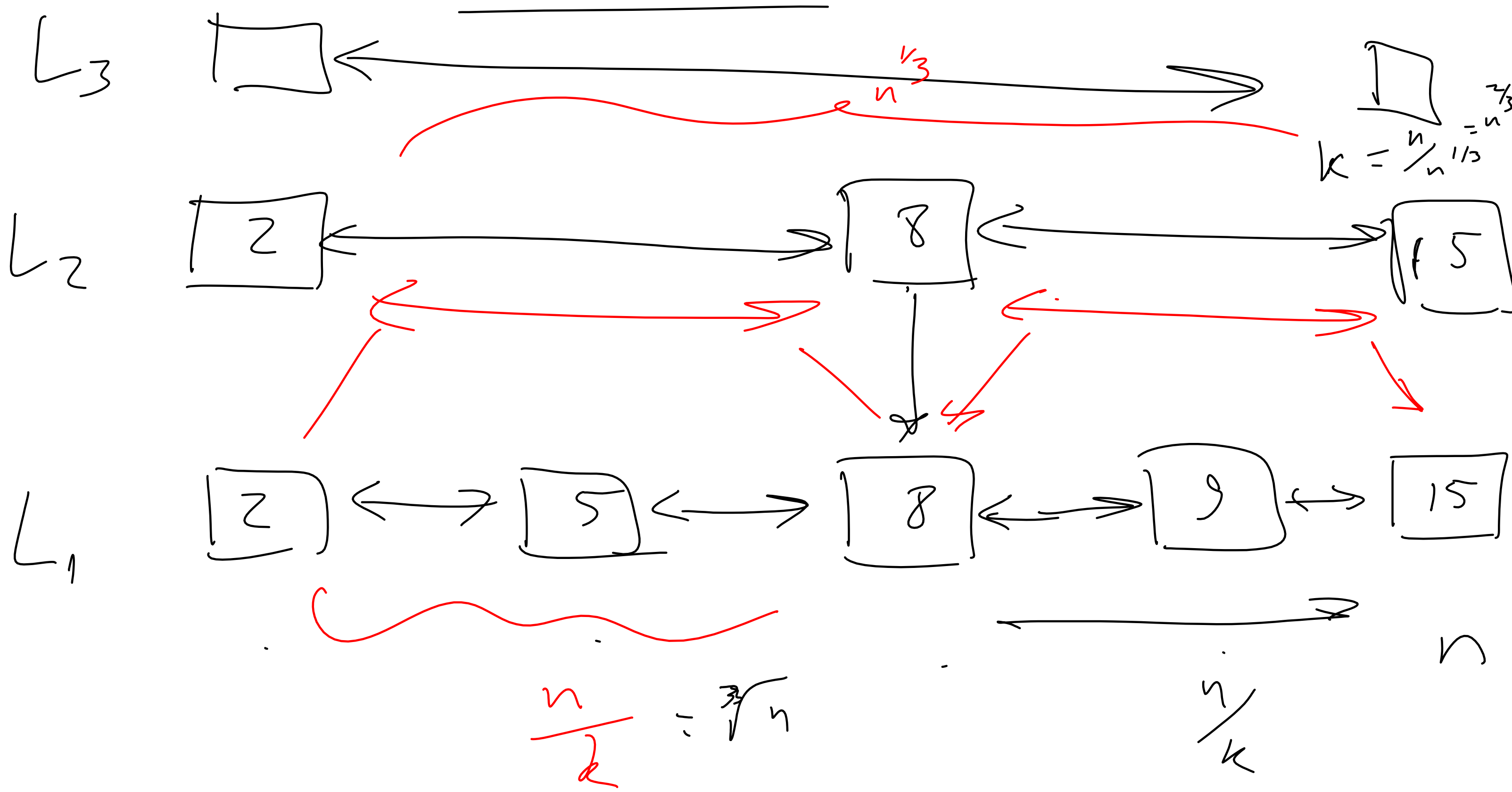
$$\le \sum_{i'=2}^{n} \frac{1}{i'} + \cancel{1} + \sum_{i'=2}^{n} \frac{1}{i'}$$

$$\le 2 \cdot \sum_{i'=2}^{n} \frac{1}{i'} \le 2 \cdot \int_{1}^{n} \frac{1}{x}\, dx$$

$$E[depth(x_k)] = 2 \ln n = O(\log n)$$

Skip Lists

$$k' = \frac{n^{2/3}}{n^{1/3}} = n^{1/3}$$



search time: $f(k) = k + \frac{n}{k} \Rightarrow f(\sqrt{n}) = 2\sqrt{n}$

$$f'(k) = \frac{df}{dk} = 1 + \left(-\frac{n}{k^2}\right) = 0$$

$$\underline{\underline{k = \sqrt{n}}}$$

$$f''(k) = \frac{1}{2}\frac{n}{k^3} > 0$$

<u>3 Lists:</u>

$$k = \sqrt[3]{n^2}$$

$$\Rightarrow \text{search time} = \underline{\underline{3 \cdot \sqrt[3]{n}}}$$

$$k' = \sqrt[3]{n}$$

<u>log n lists:</u>

$$\Rightarrow \text{search time: } \log n \cdot \underline{\underline{n^{\frac{1}{\log n}}}}$$

$$= 2 \log n$$

$$n^{\frac{1}{\log n}} = x = 2$$

$$\log\left(n^{1/\log n}\right) = \underline{\log x} = \log\left(n^{1/\log n}\right) = \frac{1}{\log n} \cdot \log n$$

$$= 1$$