Tools
- Chernoff Bound
- level ancestor in Trees.

Chernoff Bound:

random variables.

$X_1, X_2, \ldots, X_n$ are 0/1 independent r.v.

$Prob(X_i = 1) = p_i$

$Prob(X_i = 0) = 1 - p_i$

$X = \sum_{i=1}^{n} X_i \leftarrow \{0, 1, 2, \ldots, n\}$

$E(X) = \sum_{i=1}^{n} E(X_i) = \sum_{i=1}^{n} p_i = (\mu)$

For any $0 \leq \delta \leq 2e-1$

$Prob[X > (1+\delta)\mu] < e^{\frac{-\mu\delta^2}{4}}$

$Prob[X < (1-\delta)\mu] < e^{\frac{-\mu\delta^2}{4}}$

(*) $\delta > 2e-1$

$Prob[X > (1+\delta)\mu] < 2^{-(1+\delta)\mu}$

Problem: You are tossing a coin n times, and

$$\text{Prob}(\text{Heads}) = 1/3$$
$$\text{Prob}(\text{Tails}) = 2/3$$

(a) What is μ = expected "Heads" count?

(b) What is prob X = no of heads count (observed) $> 2\mu$

Soln For $i = 1$ to n,

$$X_i = \begin{cases} 1 & \text{if } i^{th} \text{ toss gets heads} \\ 0 & \text{o/w} \end{cases}$$

$$\text{Prob}(X_i = 1) = 1/3$$

$$E(X_i) = 1 \cdot \text{Prob}(X_i = 1) + 0 \cdot \text{Prob}(X_i = 0)$$
$$= \frac{1}{3}$$

$$X = \sum_{i=1}^{n} X_i \quad (\text{Total number of head counts})$$

$$E(X) = \sum_{i=1}^{n} E(X_i) = n * \left(\frac{1}{3}\right)$$

Note: X_i's are independent.

$$\text{Prob}(X > 2\mu) = \text{Prob}(X > (1+\delta)\mu) \quad \delta = 1$$
$$\leq e^{-\frac{\mu \delta^2}{4}}$$
$$= e^{-\frac{\mu}{4}} = e^{-\frac{n}{12}} \quad \left(\mu = \frac{n}{3}\right)$$

(c) What is prob X = no of heads count (observed) $> \mu + 10\sqrt{n \log n}$

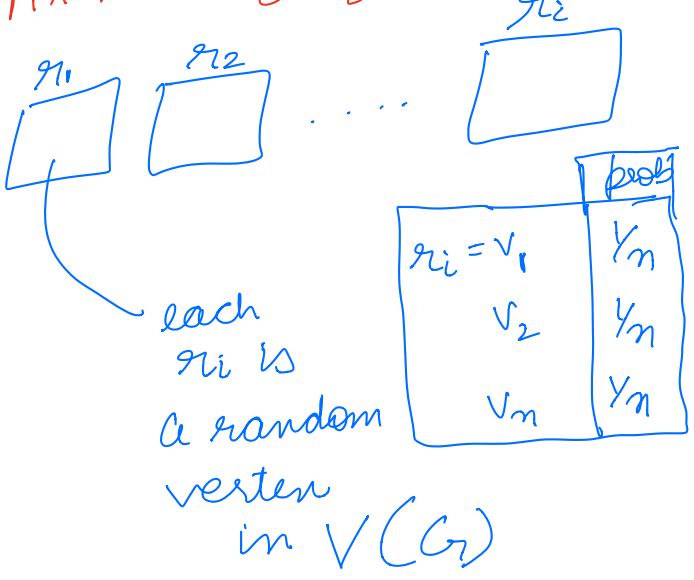
$$\delta = \theta\left(\frac{\sqrt{\log n}}{\sqrt{n}}\right)$$

$$e^{-\frac{\mu\delta^2}{4}} = e^{-c \log n} = \frac{1}{n^c}$$

inverse polynomial

Random construction 1

Fix the size of set $= t$



each r_i is a random vertex in $V(G)$

Size = Fixed

Eg. Used in Hitting set construction

Random construction 2.

You create a set W s.t.

each $v_i \in W$ w.p. "p"

Expected size $= pn$

Using chernoff bound you can show that

- Prob $(|W| > 2np) =$ very small
- Prob $(|W| < \frac{np}{2}) =$ very small

eg. Used in r-FT-k-Spanner

Lemma: Create a subset J of $V(G_1)$ s.t.

$$\text{Prob}(v_1 \in J) = 1 - \frac{1}{r}$$

$$\vdots$$

$$\text{Prob}(v_n \in J) = 1 - \frac{1}{r}$$

Then (a) $E(|V(G_1) \setminus J|) = \frac{n}{r}$

(b) What is $\text{prob}\left(|V(G_1) \setminus J| \leq \frac{2n}{r}\right)$?

Defn $x_i = \begin{cases} 1 & \text{if } v_i \in V(G_1) \setminus J \\ 0 & \text{o/w} \end{cases}$

$$E(x_i) = \text{Prob}(x_i = 1) = \frac{1}{r}$$

$$X = \sum_{i=1}^{n} x_i = \text{No of vertices in } V(G_1) \setminus J$$

$$E(X) = \sum_{i=1}^{n} E(x_i) = \frac{n}{r}$$

$$\text{Prob}\left(X \leq \frac{2n}{r}\right) = \text{Prob}\left(X \leq (1+\delta)\mu\right) \overset{(\delta=1)}{=} e^{-\frac{\mu\delta^2}{4}}$$

$$= e^{-\frac{n}{4r}} = \text{very small for } r \leq \sqrt{n}$$

Application

Recall r-FT- k-spanner problem.

① $H = (V, \emptyset)$

② For $i = 1$ to $\alpha = 16 r^3 \log(n)$

- Compute set J_i s.t. each vertex is included in J_i with prob. $= \left(1 - \frac{1}{r}\right)$
- $H_i \leftarrow$ k-spanner of $(G \mid J_i)$
- Add edges of H_i to H.

$$\#\text{ of vertices in } V(G) \setminus J_i \leq \frac{2n}{r} \quad \left(\text{w.p. } e^{-\frac{n}{4r}}\right)$$

$$\#\text{ of edges in } H_i = \left(\frac{2n}{r}\right)^{1 + \frac{2}{k+1}}$$

$$\#\text{ of edges in } H = r^3 \log n \cdot \left(\frac{2n}{r}\right)^{1 + \frac{2}{k+1}}$$

In previous calculation we use following result taught in Lecture-1.

odd k. $\quad$ size of k-spanner

$2t-1$ $\quad$ $n^{1+1/t}$

$t = \frac{k+1}{2}$ $\quad$ $n^{1+\frac{1}{t}} = n^{1+\frac{2}{k+1}}$

(*) We still require to show that with high probability

$V(G) \setminus J_1, \ldots, V(G) \setminus J_\alpha$

all of them have $\leq \frac{2n}{r}$ vertices.

Proof:

$$\text{Prob}\left(|V(G) \setminus J_i| > \frac{2n}{r}\right) \leq e^{-\frac{n}{4r}}$$

By union bound,

$$\text{Prob}\Big(\exists\, i \text{ s.t. } |V(G) \setminus J_i| > \frac{2n}{r} \text{ vertices}\Big)$$

$$\leq \alpha \cdot e^{-\frac{n}{4r}}$$

$$\leq r^3 \log n \cdot e^{-\frac{n}{4r}}$$

$$\leq n^4 \cdot e^{-\frac{n}{4r}}$$

very small.

Level Ancestor Problem

We are given a rooted tree T on n nodes. Let r denote the root of the tree. A node v is said to be the level ancestor of node u at k (denoted by $LA(u, k)$) if it is the ancestor of u and is at height k in the tree. The aim is to maintain a data structure for rooted trees that can answer LA queries efficiently.

The following notations will be used.

- T: The input tree rooted at r.
- $T(v)$: The sub-tree of T rooted at v (note $T(v)$ comprises of v and all descendants of v).
- $height_T(v)$: One plus the distance from v to farthest leaf node lying in $T(v)$.
- $depth_T(v)$: One plus the distance from r to node v.
- S: The set of leaf nodes of tree T.

As S is the set of leaves of tree T, it is sufficient to solve the level ancestor problem for set S. This is because any non-leaf node in the tree can store a pointer to one of the leaf nodes in its subtree.

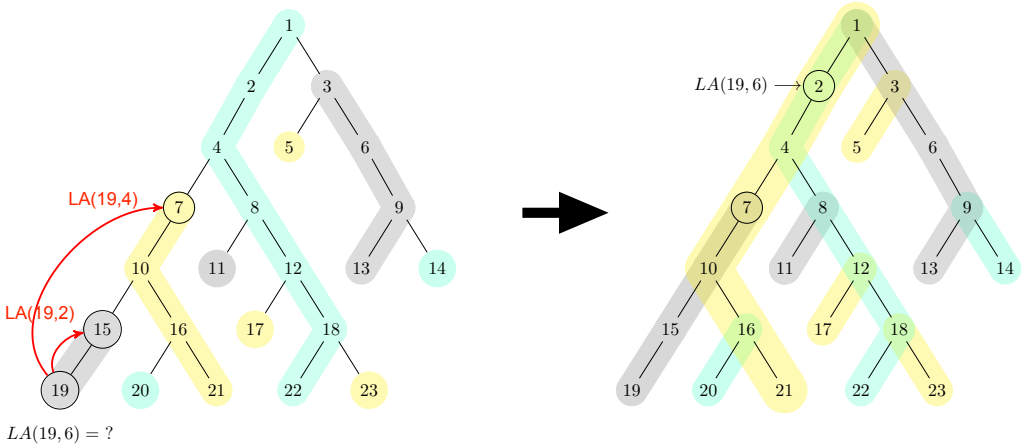


Figure 1: (i) Tree is partitioned into "ladders" (shown on left) such that each node u is contained in a ladder of length at least $height_T(u)$. (ii) Extended-Ladders are constructed (shown on right) by doubling length of each ladder.

To report the answer in $O(1)$ time the following data structures are used -

1. **Jump Pointers -** Each leaf node $v \in S$ keeps an array A_v such that the i^{th} element of the array ($A_v[i]$) stores a pointer to the ancestor of v at height 2^i, for $i \in [0, \log_2 n]$.

2. **Ladders -** The ladders is a partitioning of the tree T into a set of disjoint-paths $\mathcal{L}$ obtained as follows.

   A path from the root of the tree to one of its farthest leaf nodes is removed from the tree and added to the set $\mathcal{L}$. And then the same procedure is recursed on the trees formed on the removal of this path.

Note that finally T is partitioned into disjoint-paths (referred to as ladders) such that for each node v the ladder containing v (denote by L_v) has length greater than equal to the height of v in T.

Observation 1. *If a node v has height h, then the ladder L_v satisfy $|L_v| \geq h$.*

3. **Extended Ladders -** The extended-ladders are constructed by doubling each ladder, i.e. by adding to each ladder as many ancestors as the number of nodes in it. For each $v \in T$, let EL_v denote the extended-ladder obtained by doubling the ladder L_v.

   Observation 2. *If a node v has height h, then EL_v stores at least h immediate ancestors of v.*

   Observation 3. $\sum_{v \in S} |EL_v| \leq \sum_{v \in S} 2|L_v| = 2n$.

4. **LA within an Extended Ladder -** For each extended-ladder EL_v corresponding to a leaf $v \in S$, we store answers to $LA(v, k)$ for $k \leq |EL_v|$.
   By Observation 3, the total space used for storing this information is $O(n)$.

Now we see that the LA queries for leaf nodes can be answered in worst-case constant time by using the extended-ladders and jump pointers.

Theorem 1. *The LA queries for a rooted tree of size n can be answered in $O(1)$ time by using $O(n + |S| \log n)$ space data structures, where S is the set of leaf nodes of the tree.*

Proof. To report the ancestor of a leaf node w at height h, by jump pointers first we go to its ancestor u at height 2^i such that $2^i \leq h < 2^{i+1}$. Now by Observation 2, the extended-ladder EL_u contains at least 2^i immediate ancestors of u, and hence $LA(w, h)$ can be retrieved easily. Therefore, the ancestor of w at height h can be reported in $O(1)$ time.

For example consider Figure 1. The depth of node '19' is seven, there are jump pointers from it to its ancestor at height 2 and 4; To compute $LA(19, 6)$ first we go to node '7' which is at height four using the jump pointers; Now the extended-ladder corresponding to node '7' (shown in yellow color) will have at least four ancestors of '7', or till root is reached. Thus $LA(19, 6)$ can be computed using the extended-ladder corresponding to node '7'.

The total space used is $O(n + |S| \log n)$. □

Theorem 2. *The LA queries for a rooted tree of size n can be answered in $O(1)$ time by using an $O(n \log n)$ space data structure.*