CS5330 Randomized Algorithms

Deterministic Algo: for every input, algo gives correct output quickly   input $\rightarrow \boxed{\text{Algo}} \rightarrow$ output

Randomized Algo: for every input, for most choices of randomness, the algo gives correct output quickly   input $\rightarrow \boxed{\text{Algo}} \rightarrow$ output, random number $\nearrow$
(i.e. occasionally it may run slowly or give the wrong answer)

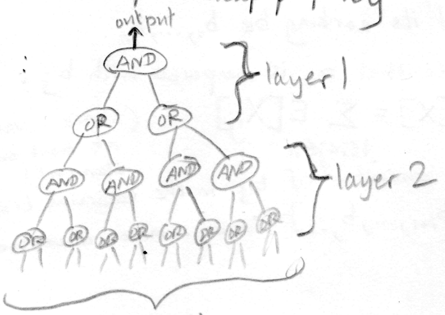
Running time measurement → expected runtime
→ high probability bound on runtime

Fact: $1-x \leq e^{-x}$

Treap (with randomized priorities) building $\cong$ randomized quicksort
→ elements of $\langle X, \overset{\text{random}}{Y} \rangle$

where X satisfies BST property : left $< X <$ right
and Y satisfies heap property : $Y >$ left and $Y >$ right

AND/OR tree :



$n = 2^{2k}$ bits input

$f : \{0,1\}^{16} \rightarrow \{0,1\}$

randomised algo: from root, query random subtree, if the answer from that subtree is insufficient then query the other subtree

Karger min cut (multigraph): While there are more than two vertices, pick a random edge and identify the two endpoints. Output the number of edges remaining. $P(\text{answer is correct}) \geq \frac{2}{n(n-1)}$

Proof via properties: ① the min cut never decreases
② if the multigraph has n vertices and min-cut size k, then it has at least $\frac{nk}{2}$ edges

Cor: there cannot be more than $\frac{n(n-1)}{2}$ distinct min cuts.

Probability space
- (countable) sample space Ω (i.e. a set of countably many elements) : set of possible outcomes of a random process
- probability measure $Pr : \Omega \rightarrow [0,1]$

Event: any subspace of the sample space Ω **Independent**: $Pr(E \cap F) = Pr(E) \cdot Pr(F)$
- $Pr(E) = \sum_{w \in E} Pr(w)$
- union bound : $Pr(E_1 \cup E_2) \leq Pr(E_1) + Pr(E_2)$

Freivald's algorithm : Given $n \times n$ matrices A, B, C, does $AB = C$?
- Algorithm : pick random $r \in \{0,1\}^n$ and check that $A(Br) = Cr$
- $O(n^2)$ time, and $AB \neq C \Rightarrow Pr(ABr \neq Cr) \geq \frac{1}{2}$ (so we can just repeat $\lceil \log_2 \frac{1}{\delta} \rceil$ times
  - Proof : there must be a entry d_{ij} in $D = AB - C$ that is nonzero. consider the r_j that will be multiplied to d_{ij}. assuming everything else has been fixed, at least one of $\left\{ \begin{matrix} r_j = 1 \\ r_j = 0 \end{matrix} \right\}$ will imply that $Dr \neq 0$.

Random variable: a function $X : \Omega \to \mathbb{R}$.

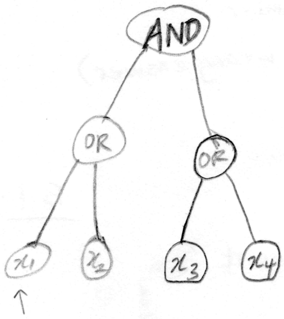
Expectation: $E[X] = \sum_i i \cdot Pr(X=i)$ (could be unbounded, in which case $E[X]=\infty$)

Linearity of expectation: $E\left[\sum_i X_i\right] = \sum_i E[X_i]$

Variance: $Var[X] = E\left[(X-E[X])^2\right] = E[X^2] - E[X]^2$

Conditional expectation: $E[X|E] = \sum_x x \cdot Pr(X=x|E)$

AND-OR tree evaluation:



E_{k-1} := expected running time

AND

$(0,0)$: $E_k \le 2 \cdot E_{k-1} + O(1)$

$(0,1)$: $E_k \le 2.75 \cdot E_{k-1} + O(1)$

$(1,1)$: $E_k \le 3 \cdot E_{k-1} + O(1)$

OR

$(1,1)$: $E[F'_k(x,y)] \le E_{k-1} + O(1)$

$(0,1)$: $E[F'_k(x,y)] \le 1.5 \cdot E_{k-1} + O(1)$

$(0,0)$: $E[F'_k(x,y)] \le 2 \cdot E_{k-1} + O(1)$

Indicator random variables: $X(\omega) := \begin{cases} 1 & \text{if } \omega \in E \\ 0 & \text{otherwise} \end{cases}$

- $E[X] = Pr(E)$

Randomized quicksort analysis: Let the input be $a_1, \dots, a_n$ and its sorting be $b_1, \dots, b_n$.

Let X_{ij} be the indicator ran. var. that b_i is compared with b_j.

Then: expected # of comparisons: $E[X] = \sum_{1 \le i < j \le n} E[X_{ij}]$ (since every element is compared at most once with each other element)

Observe: ① $X_{ij} = 1$ if b_i is an ancestor of b_j in the recursion tree or vice versa

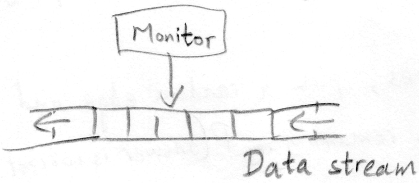
② $X_{ij} = 0$ if any of $\{b_{i+1}, \dots, b_{j-1}\}$ is a common ancestor of b_i & b_j in the recursion tree

- Hence $Pr(X_{ij}=1) = \frac{2}{j-i+1}$

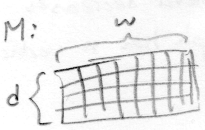
Hence $E[X] \in O(n \log n)$

Markov's inequality: If X is a non-negative ran. var then $\forall k > 0$: $Pr(X \ge k) \le \frac{E[X]}{k}$

Count-min sketch:



- Stream contains m elements from universe $\{1, \dots, n\}$
- f_i = number of occurences of f_i
- Algorithm: choose d independent hash functions $h_1, \dots, h_d : [n] \to [w]$ (note: $[n] := \{1, \dots, n\}$)



- on each input a, For each hash function $j \in [d]$, increment $h_j(a)$-th column of j-th row.
- given a query i, output $\hat{f}_i = \min_{j \in [d]} M[j][h_j(i)]$

Thm: If $d = \log\left(\frac{1}{\delta}\right)$ and $w = \frac{2}{\varepsilon}$, then $\forall i \in [n]$, $f_i \le \hat{f}_i \le f_i + \varepsilon m$ with probability at least $1-\delta$.

Space needed: $O\left(\frac{1}{\varepsilon} \log m \log\left(\frac{1}{\delta}\right)\right)$ + cost of storing hash functions

Chebyshev's inequality: If X is a ran. var. then $\forall k > 0$: $Pr(|X - E[X]| \ge k) \le \frac{Var[X]}{k^2}$

Facts about variance: $Var[X] = E[(X-E[X])^2] = E[X^2] - E[X]^2$

If X and Y are independent:
- $E[XY] = E[X] \cdot E[Y]$
- $Var[X+Y] = Var[X] + Var[Y]$

Not necessarily independent: $Var\left[\sum_i X_i\right] = \sum_i Var[X_i] + 2 \cdot \sum_{i<j} Cov(X_i, X_j)$

where $Cov(X,Y) = E[X \cdot Y] - E[X] \cdot E[Y]$

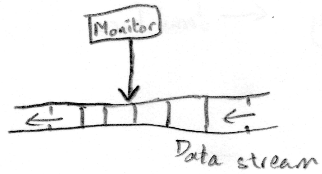
Binomial: $E[X] = np$

$Var[X] = np(1-p)$

Bernoulli: $E[X] = p$

$Var[X] = p(1-p)$

Count sketch :



- Given $i \in [n]$, want to estimate f_i.
- Alg: choose a hash function $h : [n] \to [w]$, and a random signs $\sigma_1, \dots, \sigma_n \in \{-1, 1\}$
  - on encountering a, update $C[h(a)] \leftarrow C[h(a)] + \sigma_a$
  - given a query i, output $\hat{f}_i = \sigma_i \cdot C[h(i)]$

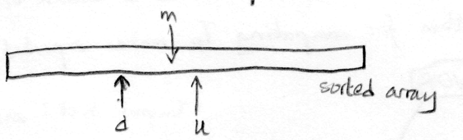
Thm : If $w = \frac{3}{\varepsilon^2}$ then $\forall i \in [n]$, $|\hat{f}_i - f_i| \le \varepsilon \sqrt{\sum_i f_i^2}$ with probability at least $\frac{2}{3}$.

Median finding :

- Median: the $\lceil \frac{n}{2} \rceil$'th element in the sorted order
- Best deterministic algorithm must make at least $2n$ comparisons
- Randomised algorithm taking $1.5n + o(n)$ comparisons:

Lazy median:

- Idea:



want to pick d and u

s.t. $d < m < u$ and there are n^c elements between d and u for $c < 1$.

- it takes $\frac{3n}{2} + o(n)$ to determine the set of elements between d and u
- the n^c elements can be sorted normally with $o(n)$ comparisons

- Algorithm:
  - Pick $\lceil n^{\frac{3}{4}} \rceil$ elements in S at random with replacement into set R
  - Sort R
  - Let d be the $\lfloor \frac{1}{2} n^{\frac{3}{4}} - \sqrt{n} \rfloor$ smallest element of R $\rightsquigarrow \mathcal{E}_1 : |\{r \in R \mid r \le m\}| < \frac{1}{2} n^{\frac{3}{4}} - \sqrt{n}$
  - Let u be the $\lfloor \frac{1}{2} n^{\frac{3}{4}} + \sqrt{n} \rfloor$ smallest element of R $\rightsquigarrow \mathcal{E}_2 : |\{r \in R \mid r \ge m\}| < \frac{1}{2} n^{\frac{3}{4}} - \sqrt{n}$
  - By comparing every element in S to d and u, compute $C = \{x \in S \mid d \le x \le u\}$ and $\ell_d = |\{x \in S \mid x < d\}|$ and $\ell_u = |\{x \in S \mid x > u\}|$
  - If $\ell_d > \frac{n}{2}$ or $\ell_u > \frac{n}{2}$ then FAIL
  - If $|C| > 4n^{\frac{3}{4}}$ then FAIL $\longrightarrow \mathcal{E}_3 : |C| > 4n^{\frac{3}{4}}$
  - Sort C and output the $(\lfloor \frac{n}{2} \rfloor - \ell_d + 1)$'th element in the sorted order.

Proof: $\mathcal{E}_1$ and $\mathcal{E}_2$: $E[|r \in R : r \le m|] = \frac{1}{2} n^{\frac{3}{4}}$ (since each element in R has $\frac{1}{2}$ probability of being $\le m$)

$\therefore$ By Chebyshev, $\Pr(\mathcal{E}_1) \le \frac{n^{\frac{3}{4}}/4}{n} \le O(n^{-\frac{1}{4}})$

$\mathcal{E}_3$:
- Suppose $> 2n^{\frac{3}{4}}$ elements of C are smaller than m
- Then, $< \frac{1}{2} n - 2n^{\frac{3}{4}}$ elements are smaller than d
- But R has $\frac{1}{2} n^{\frac{3}{4}} - \sqrt{n}$ elements smaller than d
- Similar argument using Chebyshev ... $\Pr(\mathcal{E}_3) = O(n^{-\frac{1}{4}})$

- Since $O(n^{-\frac{1}{4}}) \in o(1)$, can repeat the algorithm until it does not fail.

Las Vegas algorithm: always gives the correct solution, but runtime may vary ← focus here

Monte Carlo algorithm: solution might be incorrect, but running time fixed.

Minimax lemma: $\min_i \max_j C(i,j) \geq \max_j \min_i C(i,j)$

Minimax lemma for mixed strategy: Row player fixes prob p_i for row i, Col player fixes prob q_j for col j

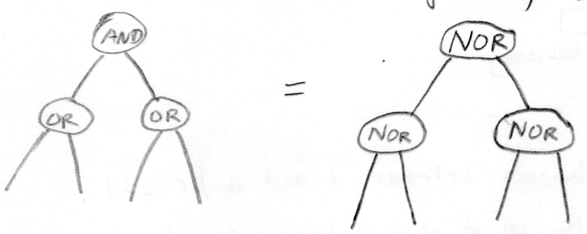
$$\min_{p_1,\dots} \max_j \sum_i p_i \cdot C(i,j) \geq \max_{q_1,\dots} \min_i \sum_j q_j \cdot C(i,j)$$

Von Neumann's minimax thm: this is equality

Yao's minimax principle: To show that the expected cost for any randomised algorithm is $\geq T$, it suffices to show that one distribution Q over inputs has that:

For any $\underbrace{\text{deterministic algorithm } A}_{\text{usually modelled as a search tree}}$, $\mathbb{E}_{x\sim Q}[\text{cost of running } A \text{ on } x] \geq T$

AND-OR tree minimum: any randomized algorithm for computing T_k makes expected at least $n^{0.694}$ queries ($n = 2^{2k}$)



- Input dist: each leaf independently set to 1 with prob $p = \frac{3-\sqrt{5}}{2}$
- every node in the NOR tree is 1 with prob p, since $(1-p)^2 = p$
- $T(h) = T(h-1) + (1-p)T(h-1) \geq 1.618 \cdot T(h-1)$, so $T(2k) \geq 1.618^{2k} = n^{0.694}$

Hoeffding bound: Let $X_1, \dots, X_n$ be independent rand. vars, and assume that $X_i \in [a_i, b_i]$ for every i, and $\mu_i = \mathbb{E}[X_i]$.
Then for any $t > 0$, $\Pr\left(\left|\sum_i (X_i - \mu_i)\right| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_i (b_i - a_i)^2}\right)$

Chernoff bound: Let $X_1, \dots, X_n$ be independent Bernoulli rand. vars.
Let $\mu_i = \mathbb{E}[X_i] = \Pr(X_i = 1)$ and $\mu = \sum_i \mu_i$.
Then for any $\delta \in (0,1)$, $\Pr\left(\left|\sum_i X_i - \mu\right| \geq \delta\mu\right) \leq 2\exp\left(-\frac{\mu\delta^2}{3}\right)$

Chernoff is better when variables are skewed (i.e. $\mathbb{E}[X_i] \ll \frac{1}{2}$)

Chernoff bound (large deviations): For any $K \geq 6\mu$, $\Pr\left(\sum_i X_i \geq K\right) \leq 2^{-K}$

$G(n,p)$: random graph on n vertices that connects each pair of distinct vertices with probability p.

Thm: let $G \sim G(n, \frac{d}{n-1})$ where $d \geq C \log n$ for some $C > 0$. Then, with probability 99%, every vertex of G has $0.9d \leq \deg \leq 1.1d$

Proof: use Chernoff bound to show that $\Pr(|\deg(X_i) - d| > 0.1d) \leq 2\exp\left(-d \cdot \frac{(0.1)^2}{3}\right) \leq 2\exp\left(-\frac{C\log n}{300}\right) \leq \frac{1}{100n}$

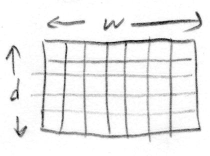
then use union bound.

Thm: n balls thrown randomly into n bins. With probability $\geq 99\%$, $\max_i X_i \leq 7 + \log n$

$X_i :=$ number of balls in bin i

Proof using Chernoff bound (large deviations) followed by union bound.

Count sketch with median:



- hash functions $h_1, \dots, h_d : [n] \to [w]$
- signs $s_1, \dots, s_d : [n] \to \{-1, +1\}$
- on each $a \in [n]$: for each $j \in [d]$, $C[j, h_j(a)] \leftarrow C[j, h_j(a)] + s_j(a)$
- $\hat{f}_i = \text{median}(s_1(i) \cdot C[1, h_1(i)], \dots, s_d(i) \cdot C[d, h_d(i)])$

So set $w = O(\frac{1}{\varepsilon^2})$ and $d = O(\log \frac{1}{\delta})$
then $\forall i \in [n]$, $|\hat{f}_i - f_i| \leq \varepsilon \sqrt{\sum_i f_i^2}$ with prob at least $1-\delta$.
(space complexity: $O(\varepsilon^{-2} \log m \log \frac{1}{\delta})$ + cost for hashfunctions & signs)

Pf: each row has at most $\frac{1}{3}$ chance of being bad, i.e. $|\hat{f}_{ij} - f_i| > \varepsilon \cdot \|f\|_2$

$X_j :=$ j is bad

$\mathbb{E}[X_j] \leq \frac{d}{3}$

$$\Pr\left(\sum_j X_j \geq \frac{d}{2}\right) \leq \Pr\left(\left|\sum_j X_j - \mathbb{E}\left[\sum_j X_j\right]\right| \geq \frac{d}{6}\right) \overset{\text{Hoeffding}}{\leq} 2\exp\left(-\frac{2(\frac{d}{6})^2}{d}\right) = 2e^{-d/18}$$

So $d = \Theta(\log \frac{1}{\delta})$ to make error prob δ.

Discrepancy: · n subsets $S_1, \dots, S_n \subseteq \{1, \dots, m\}$

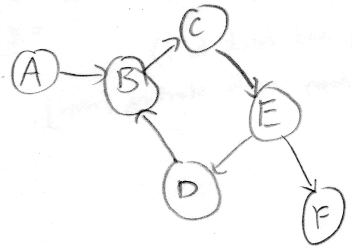
· want a vector $x \in \{-1, 1\}^m$ s.t. $\forall\, i \in [n]$, $\left|\sum_{j \in S_i} x_j\right| \le D$. ← called the discrepancy

(i.e. a vector that roughly equally partitions all sets)

Algorithm: choose x randomly. Then $\Pr\left(\max_i \left|\sum_{j \in S_i} x_i\right| > \sqrt{2m \ln n}\right) < 1$

(since, for each S_i, by Hoeffding, $\left|\sum_{j \in S_i} x_j\right| > \sqrt{2m \ln n}$ with probability $< \frac{1}{n}$)

Packet routing:



(see slides).

Hash functions should be: $h : U \to [n]$

- Pseudorandom
- Compactly representable $\rightsquigarrow$ $\mathcal{H}$: family of hash functions
- Efficiently computable

$h \in \mathcal{H}$ be uniformly chosen

number of bits to represent h is $\log_2 |\mathcal{H}|$.

E.g. $\mathcal{H} = \{h\}$ where $h(x) = x \bmod n$: not pseudorandom if x can be arbitrary

$\mathcal{H} = \{h_a : a = 1, \dots, n\}$ where $h_a(x) = ax \bmod n$: not pseudorandom if n divides x

Collision: $i, j \in U$ where $i \neq j$ and $h(i) = h(j)$

k-Universal $\mathcal{H}$: for all distinct $i_1, \dots, i_k$, $\Pr_{h \leftarrow \mathcal{H}}[h(i_1) = \dots = h(i_k)] \le \frac{1}{n^{k-1}}$

("universal" usually means 2-universal)

Strong k-universal $\mathcal{H}$: for all distinct $i_1, \dots, i_k \in U$ and all $y_1, \dots, y_k \in [n]$, $\Pr_{h \leftarrow \mathcal{H}}[h(i_1) = y_1, \dots, h(i_k) = y_k] = \frac{1}{n^k}$

Strong k-universal $\Rightarrow$ k-universal

Strong k-universal $\Rightarrow$ strong (k-1)-universal

Consequences: if h is drawn from a 2-universal family, the expected number of collisions amongst s items is $O(\frac{s^2}{n})$.

Stochastic processes: $X_0, X_1, X_2, \dots$ $\quad X_t$: state of process at time t

(X_0 ↑ initial state)

Markov chain: state X_t is only dependent on state X_{t-1}.

(X_t is independent of $X_1, \dots, X_{t-2}$, conditioned on the value of X_{t-1})

Transition matrix: $P_{i,j} = \Pr[X_t = j \mid X_{t-1} = i]$ (i: from, j: to)

$$P = \begin{pmatrix} P_{0,0} & P_{0,1} & \cdots & P_{0,j} & \cdots \\ P_{1,0} & P_{1,1} & \cdots & P_{1,j} & \cdots \\ \vdots & & \ddots & \vdots & \\ P_{i,0} & P_{i,1} & \cdots & P_{i,j} & \\ & & & & \ddots \end{pmatrix}$$

Stationary distribution: $\pi = (\pi_1, \dots, \pi_n)$ where $\pi P = \pi$

· every finite state Markov chain has a stationary distribution

Irreducible: For any two states i and j, both $i \to j$ and $j \to i$ (where $i \to j$: $P_{i,j}^t > 0$ for some $t \ge 0$) (i.e. can get from i to j in t steps)

* Every finite state **irreducible** Markov chain has **exactly one** stationary distribution: $\boxed{\pi_i = \frac{1}{h_{i,i}}}$ — we usually use π_i to find $h_{i,i}$.

Hitting time: $h_{i,j}$: expected time to reach state j from state i. $h_{i,j} = E[\min\{t > 0, X_t = j\} \mid X_0 = i]$

Aperiodic: $\forall i$, $\gcd(\{t : P_{i,i}^t > 0\}) = 1$ $\quad$ Equiv aperiodic: $\forall i, \exists t_0 > 0$ s.t. $\forall t \ge t_0$, $P_{i,i}^t > 0$.

6

Thm: Given a finite state irreducible aperiodic Markov chain with stationary distribution π, for any state i,
$\lim_{t\to\infty} Pr[X_t = i] = \pi_i$ (i.e. the distribution converges to the stationary distribution)

Time reversible: There exists a distribution π s.t. $\forall i,j$: $\pi_i P_{i,j} = \pi_j P_{j,i}$
(i.e. the edge probability backward is the same as forward, it is like running the Markov chain backwards)

- If a Markov chain is time-reversible, then π is a stationary distribution (since $(\pi P)_j = \sum_i \pi_i P_{i,j} = \sum_i \pi_j P_{j,i} = \pi_j \sum_i P_{j,i} = \pi_j$)

Commute time between two vertices i and j : $C_{ij} = h_{ij} + h_{ji}$ (time to go from i to j and back to i)

Cover time: max over all vertices j of the expected time to visit all vertices in a random walk starting from j.

Random walk: $P_{i,j} = \frac{1}{deg(i)}$ for each neighbour j of i.

- stationary distribution: $\pi_i = \frac{deg(i)}{2|E|}$, $h_i = \frac{2|E|}{deg(i)}$
- If $G = K_n$, then $h_{i,j} = n$ and cover time $= O(n \log n)$ (coupon collector problem)
- If G is the path of length n, then the cover time is $O(n^2)$
- If G is the "lollipop graph" with a vertex u participating in both a clique of size $\frac{n}{2}$ and a disjoint path of length $\frac{n}{2}$ ending in v, then $h_{v,u} = O(n^2)$, $h_{u,v} = O(n^3)$, cover time $= O(n^3)$
- **Thm:** $\underset{\text{cover time}}{C(G)} \leq 2|E| \cdot (n-1)$ → can be shown by considering Markov chain on edges:
  - if i is adjacent to j then $C_{ij} \leq 2|E|$.

Markov chain Monte Carlo: for sampling something from a desired distribution.

- Design a Markov chain whose stationary distribution π is the desired distribution
- Make sure that each step can be executed efficiently, and convergence to π is "fast".

Metropolis-Hastings step: to make an arbitrary Markov chain on state space Ω with transition matrix Q have a desired stationary distribution π.

Algorithm:

① Pick Y by taking a step from X_t using Q

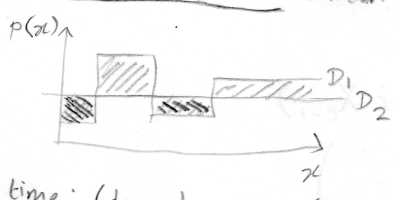
② $\alpha := \frac{1}{2} \min\left\{1, \frac{\pi(Y) \cdot Q_{X_t,Y}}{\pi(X_t) \cdot Q_{Y,X_t}}\right\}$

③ With probability α, $X_{t+1} \leftarrow Y$ (transit), else $X_{t+1} \leftarrow X_t$ (stay the same)

- can be shown that $\pi_i P_{i,j} = \frac{\pi_j Q_{j,i}}{2} = \pi_j P_{j,i}$ where P is the new transition matrix, so the new stationary distribution is π.

Rapid mixing & approximate counting:

Total variation (TV) distance between distributions D_1 & D_2: $\|D_1 - D_2\| := \max_{A \subseteq \Omega} \left(\Pr_{x \sim D_1}[x \in A] - \Pr_{x \sim D_2}[x \in A] \right)$
(something like the L_∞ norm... over subsets)



← intuitively, it is all the top shaded areas, or equiv. all the bottom shaded areas.

Mixing time: (depends on ε (how well we need it mixed)): $d(t) := \max_{x \in \Omega} \|P^t(x) - \pi\|$ where π is the stationary distribution.
$t_{mix}(\varepsilon) = \min\{t \mid d(t) \leq \varepsilon\}$

- rapidly mixing: $t_{min}(\varepsilon) \in poly(\text{problem size}, \varepsilon)$

Coupling between distributions D_1 & D_2 : (X, Y) where $X \sim D_1$ and $Y \sim D_2$.
(X and Y are generally not independent)

Coupling lemma: $\|D_1 - D_2\| \leq Pr[X \neq Y]$ where (X,Y) is a coupling between D_1 & D_2.

Coupling of Markov chains: same transition matrix, but possibly different starting states

Markov chain coupling lemma: If (X_t, Y_t) is a coupling of a Markov chain, and T is such that for every $x, y \in \Omega$, $Pr[X_T \neq Y_T \mid X_0 = x, Y_0 = y] \leq \varepsilon$, then $t_{mix}(\varepsilon) \leq T$.

- E.g. card shuffling:
  - states: permutations of $\{1, \dots, n\}$
  - transition: moving a uniformly random card to the top
  - chains couple once all cards have moved to the top at least once
  - $Pr[\exists\, C : C$ not moved to top in T steps$] \underset{\text{union bound}}{\leq} n \cdot (1 - \frac{1}{n})^T \leq n \cdot e^{-\frac{T}{n}}$
  - When $T \in O(n \ln \frac{n}{\varepsilon})$, we have $Pr[X_T \neq Y_T] \leq \varepsilon$
- E.g. lazy walk on a cycle:
  - stay at current node with probability $\frac{1}{2}$, move clockwise with probability $\frac{1}{4}$, move anticlockwise with probability $\frac{1}{4}$.
  - coupling of (X_T, Y_T): with probability $\frac{1}{2}$, move X_t one step in a random direction, otherwise, move Y_t one step in a random direction
  - $d_t := X_t - Y_t \pmod n$ increases or decreases by 1 with probability $\frac{1}{2}$ each, so $d_t \equiv 0 \pmod n$ within $O(n^2)$ steps in expectation.
  - By Markov ineq: when $\underset{\text{mixing time}}{T} \geq \frac{n^2}{\varepsilon}$, $Pr[X_T \neq Y_T] = Pr[\text{coupling time} \geq T] \underset{\text{Markov}}{\leq} \frac{E[\text{coupling time}]}{T} = \varepsilon$
- Eg. random spanning trees:
  - consider random rooted arborescences instead (directed tree where all edges point away from root)
  - with probability $\frac{1}{2}$, keep current arborescence. otherwise, pick a random edge from some vertex to the current root, and re-root the trees at the new root.
  - takes $O(n^6)$ time for roots to agree
  - after roots agree, takes another $O(n^3)$ to visit all vertices (and then entire arborescence will agree)
  - by Markov ineq: mixing time is $O(\frac{n^6}{\varepsilon})$

Fully polynomial randomised approximation scheme (FPRAS): given input x and parameters $\varepsilon, \delta \in (0,1)$, the algorithm outputs an approximation $\hat{V}$ such that $Pr[|\hat{V} - V(x)| > \varepsilon \cdot V(x)] < \delta$, and the algorithm runs in $poly(|x|, \frac{1}{\varepsilon}, \log(\frac{1}{\delta}))$.

(note: $V(x)$ is the true answer, so it means $\hat{V}$ is a $(1 \pm \varepsilon)$-multiplicative approximation of $V(x)$.)

- FPRAS for size estimation:
  - suppose $S \subseteq \Omega$, and we can efficiently generate uniform samples from Ω, and efficiently check if a sample is in S.
  - then there is an FPRAS for computing $\mu = \frac{|S|}{|\Omega|}$ that makes $\frac{3 \ln(\frac{2}{\delta})}{\varepsilon^2 \mu}$ samples from Ω.

Fully polynomial almost uniform sampler (FPAUS): (for sampling almost uniformly from Ω, usually as part of an FPRAS) given input x and parameter $\varepsilon > 0$, output distribution $W(x)$ satisfies

$$\|\underbrace{W(x)}_{\text{output distribution}} - \underbrace{U(x)}_{\text{uniform distribution}}\| \leq \varepsilon$$

and the algorithm runs in $poly(|x|, \frac{1}{\varepsilon})$.

- E.g. k-colourings: Finding an approximation for the number of k-colourings in a given graph G.
  - G has n vertices and m edges $= e_1, \dots, e_m$
  - Let G_i be the graph with all n vertices and edges $\{e_1, \dots, e_i\}$
  - Let C_i be the number of k-colourings for G_i. (C_m is what we want, and $C_0 = k^n$)
  - $C_m = \frac{C_m}{C_{m-1}} \times \frac{C_{m-1}}{C_{m-2}} \times \dots \times \frac{C_1}{C_0} \times C_0$
  - want to approximate each $\frac{C_i}{C_{i-1}}$ up to multiplicative approximation $(1 \pm \frac{\varepsilon}{2m})$ with probability $\frac{\delta}{m}$, then by union bound, we can approximate $\frac{C_m}{C_0}$ up to multiplicative approximation $(1 \pm \varepsilon)$, so we have an FPRAS for C_m.
  - How to approximate $\frac{C_i}{C_{i-1}}$? Two sources of error: our sampler for C_{i-1} is only approximately uniform; we are sampling a finite amount of times
    - can be shown that $\frac{C_i}{C_{i-1}} \geq \frac{3}{4}$ when $k > 2\Delta$
    - use FPAUS on G_{i-1} to produce a colouring A s.t. $\left| Pr[A \text{ colours } G_i] - \frac{C_i}{C_{i-1}} \right| \leq \frac{\varepsilon}{8m}$
    - if we sample r colourings from FPAUS on G_{i-1}, let Z_i be how many colour G_i. then:
      $r\frac{C_i}{C_{i-1}}\left(1 - \frac{\varepsilon}{6m}\right) \leq r \cdot \left(\frac{C_i}{C_{i-1}} - \frac{\varepsilon}{8m}\right) \leq E[Z_i] \leq r \cdot \left(\frac{C_i}{C_{i-1}} + \frac{\varepsilon}{8m}\right) \leq r\frac{C_i}{C_{i-1}}\left(1 + \frac{\varepsilon}{6m}\right)$
    - since $E[Z_i] \geq \frac{r}{2}$, by Chernoff bound, with $r = \frac{m^2}{\varepsilon^2} \log\left(\frac{m}{\delta}\right)$ samples, with probability $1 - \frac{\delta}{m}$, can get a $(1 \pm \frac{\varepsilon}{6m})$-multiplicative approximation of Z_i
    - combining the two sources of error, we have a $(1 \pm \frac{\varepsilon}{6m})^2 \leq (1 \pm \frac{\varepsilon}{2m})$-multiplicative approximation of $\frac{C_i}{C_{i-1}}$.

Conditional expectation: $E[X|Y]$ is a random variable $f(Y)$ where $f(y) = E[X|Y=y]$

- Lemma: $E[E[X|Y]] = E[X]$
  - $E[E[X|Y,Z]|Z] = E[X|Z]$

Martingale: $Z_0, \dots, Z_n$ is a martingale w.r.t. $X_0, \dots, X_n$ if for all $i \geq 0$:

- Z_i is a function of $X_0, \dots, X_i$
- $E[|Z_i|] < \infty$
- $E[Z_{i+1}|X_0, \dots, X_i] = Z_i$

Azuma-Hoeffding bound (like Hoeffding bound, but for martingales): If $Z_0, \dots, Z_n$ is a martingale s.t. $B_k \leq Z_k - Z_{k-1} \leq B_k + c_k$ where B_k could depend on $Z_0, \dots, Z_{k-1}$, then for all $t \geq 1$ and any $\lambda > 0$, $Pr[|Z_t - Z_0| \geq \lambda] \leq 2\exp\left(-\frac{2\lambda^2}{\sum_{k=1}^{t} c_k^2}\right)$

Doob martingale: want to find an approximation for $Z = f(X_1, \dots, X_n)$

- for any $i \geq 0$, $Z_i = E[f(X_1, \dots, X_n)|X_1, \dots, X_i]$
- then $Z_0, \dots, Z_n$ forms a martingale w.r.t. $1, X_1, \dots, X_n$ and $Z_n = Z$

- E.g.: pattern matching: suppose $X_1, \dots, X_n$ are random chars from an alphabet of size s, and we want to find $Z :=$ number of occurences of a particular substring B of size k
  - Let $Z_i = E[Z|X_1, \dots, X_i]$, so $Z_n = Z$ and $Z_0 = E[Z] = (n-k+1) \cdot s^{-k}$
  - Note that $|Z_{k+1} - Z_k| \leq k$ (one char can contribute at most k occurences of B
  - By Azuma-Hoeffding, $Pr[|Z - E[Z]| \geq \varepsilon] \leq 2\exp\left(-\frac{\varepsilon^2}{2nk^2}\right)$

- E.g. m balls, n bins, let X_i be the bin the i^{th} ball lands in
  - want to find $Z :=$ number of empty bins
  - let $Z_i = E[Z \mid X_1, \dots, X_i]$ (Doob martingale)
  - note that $|Z_{k+1} - Z_k| \leq 1$, $Z_n = Z$, $Z_0 = E[Z]$
  - By Azuma-Hoeffding, $Pr[|Z - E[Z]| \geq \varepsilon] \leq 2\exp\left(-\frac{\varepsilon^2}{2m}\right)$
    (note: we don't actually need to know $E[Z]$)

c-Lipschitz : $f(X_1, \dots, X_n)$ is c-Lipschitz if for all i and all possible values $x_1, \dots, x_n$ and y_i:

$$|f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) - f(x_1, \dots, x_{i-1}, y_i, x_{i+1}, \dots, x_n)| \leq c$$

(i.e. each coordinate doesn't contribute too much to the final result)

- If f is c-Lipschitz and $X_1, \dots, X_n$ are independent, then $|Z_k - Z_{k-1}| \leq c$ for the Doob martingale $Z_0, \dots, Z_n$.

McDiarmid's inequality (special case of Doob + Azuma-Hoeffding):

If f is a function on n variables and is c-Lipschitz, and $X_1, \dots, X_n$ are independent random variables,
then $Pr[|f(X_1, \dots, X_n) - E[f(X_1, \dots, X_n)]| \geq \lambda] \leq 2\exp\left(-\frac{2\lambda^2}{nc^2}\right)$