CSCI 6220: Randomized Algorithms

Chunheng Jiang

December 20, 2018

1 Asymptotic Analysis

Asymptotic analysis concentrates on the change in running time when increasing the size of an algorithm's inputs, e.g. double the input size. It avoids the perturbation from machines in analysis.

Definition 1.1 (Asymptotic Notation)**.** *For a function $f(n)$, three sets of functions*

$$\begin{aligned} O(f(n)) &= \left\{ g(n) \middle| \exists \epsilon > 0, N > 0, s.t. |g(n)| \geq \epsilon |f(n)|, \forall n \geq N \right\}, \\ \Omega(f(n)) &= \left\{ g(n) \middle| \exists \epsilon > 0, N > 0, s.t. |g(n)| \leq \epsilon |f(n)|, \forall n \geq N \right\}, \\ \Theta(f(n)) &= \left\{ g(n) \middle| \exists \epsilon > 0, N > 0, s.t. |g(n)| = \epsilon |f(n)|, \forall n \geq N \right\}, \end{aligned}$$

are defined to express the asymptotic behavior of an algorithm's running time in terms of upper bound, lower bound, both the upper bound and the lower bound, respectively.

To be clear, $p(n) \in o(n)$ implies that $\forall \epsilon > 0$, $\exists N$, s.t. $0 \leq p(n) < \epsilon n, \forall n \geq N$.

Suppose an algorithm requires at least $\sqrt{\log n / n}$ steps, its complexity is $\Omega(\sqrt{\log n / n})$. Another algorithm requires at most n^2 rounds, therefore has complexity of $O(n^2)$.

2 Equalities and Inequality Bounds

- Binomial Coefficients:

$$\begin{aligned}(p+q)^n &= \textstyle\sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k}, \\ \binom{n}{k} &= \binom{n}{n-k} = \tfrac{n!}{k!(n-k)!}, \forall n \geq k \geq 0.\end{aligned}$$

- Power Series Expansions:

$$\begin{aligned}e^x &= 1 + x + \tfrac{x^2}{2!} + \tfrac{x^3}{3!} + \cdots \\ \ln(1+x) &= x - \tfrac{x^2}{2} + \tfrac{x^3}{3} - \tfrac{x^4}{4} + \cdots \\ (1-x)^{-1} &= 1 + x + x^2 + x^3 + x^4 + \cdots\end{aligned}$$

- Stirling's Formula:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \frac{1}{12n} + O\left(\frac{1}{n^2}\right)\right).$$

- Let $n \geq k \in \mathbb{N}^+$.

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \min\left\{\left(\frac{en}{k}\right)^k, \frac{n^k}{k!}\right\}.$$

- Let $m \geq \max\{n, n+k\}$.

$$\binom{m}{n}\binom{m-n}{k} = \binom{m}{n+k}\binom{n+k}{k}.$$

3 Randomized Algorithms

The design and analysis of a randomized algorithm is to show that the randomized behaviors in execution is *likely to be good*, on every input. A randomized algorithm makes random choices during execution and its behaviors vary even given the same inputs.

However, a *deterministic algorithm* will output the same outputs when given the same inputs. It follows a fixed procedure and provides deterministic output. Also, the randomized algorithms are different from the *probabilistic analysis of algorithms*, where the input is assumed to be drawn from a specific probability distribution, and show that the algorithm works for most inputs.

Randomized algorithms can be roughly categorized into two classifications: the *Las Vega* algorithms and the *Monte Carlo* algorithms.

3.1 Las Vegas Algorithms

A Las Vegas algorithm always produce the correct answer, but its running time is a random variable whose expectation is bounded.

3.2 Monte Carlo Algorithms

A Monte Carlo algorithm runs for a fixed number of steps, and produces an answer that is correct with a lower-bounded probability.

These probabilities and expectations are determined by the random choices and independency of the inputs. Therefore, the repetitions of Monte Carlo drives down the failure probability exponentially.

A Las Vegas algorithm can be converted into a Monte Carlo algorithm via early determination.

3.3 Applications

1. *data structures*: sorting, order statistics, searching
2. *graph algorithms*: minimum spanning trees, shortest paths, min-cut
3. *geometric algorithms and mathematical programming*: manipulation of geometric objects, faster algorithms for linear programming, rounding linear program solutions to integer linear program solutions
4. *probabilistic existence proofs*: show that a combinatorial object arises with non-zero probability among objects drawn from a suitable probability space

5. *derandomization*: first design a randomized algorithm, then argue that it can be derandomized to produce a deterministic algorithm

6. *algebraic identities*: polynomial and matrix identity verification, pattern matching, interactive proof systems

7. *number theoretic algorithms*: primality testing, polynomial roots and factors

8. *counting and enumerating*: matrix permanent, counting combinatorial structures

9. *parallel and distributed computing*: deadlock avoidance, distributed consensus

4 Randomized QuickSort

QuickSort algorithm is an efficient sorting method, it places the elements of an array in ascending order. It's a *divide and conquer* algorithm, it picks a pivot element in each step, and divides a larger set into two smaller subsets, s.t all elements in one subset is smaller than the pivot element, all elements in another subset is greater than the pivot element. It recursively applies the partition strategy to these subsets until all elements are in order. The time complexity of QuickSort is $O(n \log n)$, and an easier randomized QuickSort algorithm as indicated in Table 1 can obtain the same level complexity.

Theorem 4.1. *The expected number of comparisons of RandQS is at most* $2nH_n$.

Proof. The comparison operations happen in step 8 - 10. Let $S_{(i)}$ be the element in S with rank i, i.e. the ith smallest element. Let X_{ij} indicates whether there is a comparison between $S_{(i)}$ and $S_{(j)}$ in the execution of RandQS, where $j > i$. The execution result of RandQS is a binary tree, each node corresponds to a pivot element. Considering a permutation π of the elements: visiting in up-bottom order, and then from left to right within the same level. Then, π can be viewed as the order of picking pivot elements. If the pair has a comparison, one of them must be the ancestor of another one. Also, all

other elements with ranks between i and j should not be picked earlier than either $S_{(i)}$ or $S_{(j)}$. Otherwise, it will cause the partition of $S_{(i)}$ or $S_{(j)}$ to its two children, such that there is no chance to make comparison. Therefore, we should consider all elements with ranks from i to j, and compute the probability that either $S_{(i)}$ or $S_{(j)}$ are picked firstly, which is $Pr(X_{ij} = 1) = 2/(j-i+1)$.

The total number of comparison in the execution of RandQS is $X = \sum_{i=1}^{n} \sum_{j>i} X_{ij}$, and its expectation can be computed

$$\mathbb{E}[X] = \sum_{i=1}^{n} \sum_{j>i} \mathbb{E}[X_{ij}] = \sum_{i=1}^{n} \sum_{j>i} Pr(X_{ij} = 1) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \leq \sum_{i=1}^{n} \sum_{d=1}^{n} \frac{2}{d} = 2nH_n.$$

□

Algorithm 1 RandQS Algorithm

Input: A set S of n elements

Output: The elements of S in ascending order

```
1:  function QS(S)
2:      S_1 ← ∅, S_2 ← ∅                          ▷ S_1 and S_2 are disjoint
3:      Randomly pick a pivot element y
4:      C ← {y}
5:      if |S| = 1 then
6:          return C
7:      end if
8:      for i ← 1 to |S| do
9:          S_1 ← S_1 ∪ {S[i]} if S[i] ≤ y
10:         S_2 ← S_2 ∪ {S[i]} if S[i] > y
11:     end for
12:     if |S_1| ≠ ∅ then
13:         C ← QS(S_1) ∪ C                       ▷ recursively run QS on S_1
14:     end if
15:     if |S_2| ≠ ∅ then
16:         C ← C ∪ QS(S_2)                       ▷ recursively run QS on S_2
17:     end if
18:     return C                                  ▷ C = S_1 ∪ {y} ∪ S_2, both S_1 and S_2 are sorted
19: end function
```

5 Binary Planar Partitions

Given n non-intersecting line segments in the plane build a small linear decision tree that has (pieces of) at most one segment in each cell.

6 Cut Problem

Definition 6.1 (Cut). *Given $G = (V, E)$, a cut in G is a partition S, T of V, s.t. $S \cup T = V$ and $S \cap T = \emptyset$. Let denote it as $C(S, T) = \{(u, v) \in E | u \in S, v \in T\}$.*

Definition 6.2 (Min/Max-Cut Problem). *Given $G = (V, E)$, find a partition S, T of V, minimizing/maximizing the size of cut $C(S, T)$.*

Algorithm 2 Karger Algorithm

Input: multigraph $G = (V, E)$, supernodes $S \leftarrow \emptyset$

for each node $v \in V$ **do**
    set $S(v) = \{v\}$ ▷ each supernode is a new node generated from a contraction
end for
while $|V| > 2$ **do**
    choose an edge $e = (u, v)$ of G uniformly at random
    CONTRACT(e)
end while
return cut $(S(v_1), S(v_2))$ from $V = \{v_1, v_2\}$ of G
function CONTRACT(e)
    $u, v \leftarrow e$
    add a new vertex w to V, i.e. $V \leftarrow V \cup \{w\}$
    for each edge $e' \in E$ **do**
        if $e' = e$ **then**
            $E \leftarrow E - \{e'\}$ ▷ remove e from E
        else if e' share exact one end u or v with e **then**
            replace the end u or v with w
        end if
    end for
    $S(w) \leftarrow S(u) \cup S(v)$
end function

Min/Max-Cut problem is an *optimization problem*, and it's **NP-hard**. It's associated to an NPC *decision problem*: given a graph $G = (V, E)$ and an integer k, determine whether there is a cut of size at most/least k in G.

Karger Algorithm (a.k.a **Edge Contraction Algorithm**) provides an efficient randomized method to find the minimum cut. The method repeats and contracts $O(n^2 \ln n)$ times, retains the best cut with total cost $O(n^4 \ln n)$. It ends with non-zero probability to find a min-cut.

Theorem 6.1. *The Karger algorithm as shown in Table 2 returns a global min-cut of G with probability at least $1/\binom{n}{2}$, where n is the number of vertices in G.*

Proof. Suppose G has one unique global min-cut (A, B) of size k, i.e. there is a set C of k edges with one end in A and the other in B. In each step, the algorithm performs a contraction on an edge uniformly chosen at random. If an edge comes from C, it will fail immediately to find the min-cut.

To get a global min-cut of G, the chosen edges must come from the edges outside of C. To get an upper bound on the probability that an edge in C is contracted, we need to know the number of edges of G. Notice that if vertex v has degree less that k, the cut $(\{v\}, V - \{v\})$ would be of size less than k, a contradiction occurs according to the assumption. So $|E| \geq kn/2$.

Let ϵ_i be the event that an edge outside of C is contracted in step i, $1 \leq i \leq n-2$. The probability that the algorithm gives the global min-cut is

$$Pr(\epsilon_1 \cap \epsilon_2 \cap \cdots \cap \epsilon_{n-2}) = Pr(\epsilon_1) Pr(\epsilon_2|\epsilon_1) \cdots Pr(\epsilon_{n-2}|\epsilon_1 \cap \epsilon_2 \cap \cdots \cap \epsilon_{n-3}).$$

We note that at the first step $\bar{\epsilon}_1$ happens w/p at most $k/(kn/2) = 2/n$, then $Pr(\epsilon_1) \geq 1 - 2/n$. At ith step, there are $n - i$ (super)-nodes in the current graph. The probability that none of the edge in C is contracted is $Pr(\epsilon) \geq 1 - k/(k(n-i)/2) = 1 - 2/(n-i)$, $\forall i \leq n-2$. Plugging into the above equality, we get

$$Pr(\epsilon_1 \cap \epsilon_2 \cap \cdots \cap \epsilon_{n-2}) \geq (1 - \frac{2}{n})(1 - \frac{2}{n-1}) \cdots (1 - \frac{2}{3}) = \frac{2}{n(n-1)} = \binom{n}{2}^{-1}.$$

□

The algorithm fails to find a global min-cut w/p at most $1 - 1/\binom{n}{2}$. Repeatedly run it for $\binom{n}{2} \ln n$ times, the upper-bound of the failure probability will reduce to

$$\left(1 - 1/\binom{n}{2}\right)^{\binom{n}{2}\ln n} \leq e^{-\ln n} = \frac{1}{n}.$$

Lemma 6.1. *An undirected graph $G = (V, E)$ with n vertices has at most $\binom{n}{2}$ global min-cuts.*

Proof. Let $C_1, C_2, \ldots, C_r$ be all the global min-cuts of G. Let ϵ_i denote the event that C_i is returned by the algorithm, and let $\epsilon = \cup_{i=1}^{r} \epsilon_i$ denote the event that the algorithm returns any global min-cut. According to the earlier discussion, $Pr(\epsilon_i) \geq 1/\binom{n}{2}$, and each pair of ϵ_i and ϵ_j are independent. Therefore, we have

$$Pr(\epsilon) = Pr(\cup_{i=1}^{r} \epsilon_i) = \sum_{i=1}^{r} Pr(\epsilon_i) \geq r/\binom{n}{2}.$$

It's obvious, $Pr(\epsilon) \leq 1$ and then $r \leq \binom{n}{2}$. □

7 SAT Problem

Definition 7.1 (Boolean Satisfiability Problem)**.** *A boolean satisfiability problem, a.k.a SAT problem is a problem determining whether there is an assignment that satisfies a given Boolean formula. If such assignment exists, the Boolean formula is called satisfied. Otherwise, it's unsatisfied.*

Problem 7.1 (Sailor Problem)**.** *There are 40 sailors. After finish the tasks, they return back to their beds randomly. What the expected number of sailors who choose their original bunk beds? What the probability of no sailor chooses his or her original bunk?*

Solution 7.1. *(1) Let X_i is a binary indicator represents whether sailor i ($1 \leq i \leq n$) chooses his or her original bed. Assuming that sailors' choices are independent, i.e. multiple sailors*

are allowed to pick the same bed. Since each sailor has one different original bed, therefore X_i obeys Bernoulli distribution, i.e. $P(X_i = 1) = 1/n$ and $P(X_i = 0) = 1 - 1/n$. The number of sailors who choose their original beds could be written as $X = \sum_{i=1}^{n} X_i$. According to the linearity of expectation, we see that $\mathbb{E}(X) = \sum_{i=1}^{n} \mathbb{E}(X_i)$, and $\mathbb{E}(X_i) = P(X_i = 1) = 1/n$, then $\mathbb{E}(X) = \sum_{i=1}^{n} \mathbb{E}(X_i) = 1$.

(2) We know that $X = \sum_{i=1}^{n}$ obeys Binomial distribution, therefore we can compute the probability of $X = m$ where $0 \leq m \leq n$, i.e. $P(X = m) = \binom{n}{m}(1/n)^m(1 - 1/n)^{n-m}$. Therefore, the probability of no sailor chooses his or her original bunk bed is $P(X = 0) = 1 - \sum_{m=1}^{n} P(X = m) = \binom{n}{0}(1 - 1/n)^n = (1 - 1/40)^{40}$.

Problem 7.2. *There are 100 strings in a box. In each step, two string ends are picked at random, tied together and put back into the box. The process is repeated until there are no free ends. What is the expected number of loops at the end of the process?*

Solution 7.2. *Let X_t be the number of loops in the box at step t and Y_t be the number of free ends in the box, where $t = 0, 1, \ldots$. Therefore, the number of strings with free ends is $Z_t = Y_t/2$. When $t = 1$, $X_1 = 0$, $Y_1 = 2n$ and $Z_1 = n$. In each step, two string ends are randomly picked and tied together. The operation will produce two possible outcomes, (i) the two ends come from the same string, it will form a new loop; (ii) the ends come from different string, it will form a longer string with two free ends.*

Starting from step t, when case (i) occurs $X_{t+1} = X_t + 1$, and $X_{t+1} = X_t$ when case (ii) occurs. Meanwhile, we have $Y_{t+1} = Y_t - 2$ and $Z_{t+1} = Z_t - 1$, the number of free ends reduces by 2, and the number of strings with free ends reduces by 1. There are n free strings, and it terminates after n steps.

Let U_t indicates a binary indicator to represent whether the random operation in step t forms a new loop, i.e. $U_t = \{0, 1\}$. To obtain a new loop, the two free ends randomly chose in step should come from the same string. It's easy to compute $P(U_t = 1) = Z_t/\binom{Y_t}{2} = 2Z_t/[Y_t * (Y_t - 1)] = 1/(Y_t - 1)$, and $P(U_t = 0) = 1 - P(U_t = 0)$.

According to the above results, we can get the number of loops at the end of the process

$$X_n = X_1 + \sum_{t=1}^{n} U_t.$$

With the fact that $\mathbb{E}(U_t) = P(U_t = 1)$*, we have*

$$\begin{aligned}\mathbb{E}(X_n) &= X_1 + \sum_{t=1}^{n} \mathbb{E}(U_t) = \sum_{t=1}^{n} P(U_t = 1) = \sum_{t=1}^{n} 1/(Y_t - 1) \\ &= 1/(2n-1) + 1/(2n-3) + \cdots + 1/3 + 1/1 = H_{2n} - H_n/2.\end{aligned}$$

Problem 7.3. *Roll a standard dice to generate a sequence* $d_1, d_2, \ldots, d_R$*, where* R *is the first integer s.t.* d_R *is even. What's the expected sum* $d = \sum_i d_i$*?*

Solution 7.3. $\mathbb{E}(d) = \sum_{k=1}^{\infty} \mathbb{E}(\sum_i (d_i)|R = k)Pr(R = k)$*. Since there are* $p = 1/2$ *probability to get an even number in rolling a dice, i.e.* $Pr(R = k) = (1/2)^{k-1}(1/2) = (1/2)^k$*, then* $\mathbb{E}(d) = \sum_{k=1}^{\infty} \sum_i \mathbb{E}(d_i|R = k)Pr(R = k) = \sum_{k=1}^{\infty} \sum_i E(d_i|R = k)2^{-k}$*.*

Let's compute $\mathbb{E}(d_i|R = k)$*,* $i = 1, 2, \ldots, R$*. When* $i < R$*,* d_i *is odd, we get* $Pr(d_i = 1|R = k) = Pr(d_i = 3|R = k) = Pr(d_i = 5|R = k) = 1/3$ *and* $\mathbb{E}(d_i|R = k) = (1 + 3 + 5)/3 = 3$*. When* $i = R$*,* d_i *is even, and* $Pr(d_i|R = k) = 1/3$*. The possible values for* d_R *is* $\{2, 4, 6\}$ *and the expectation* $\mathbb{E}(d_i|R = k) = (2 + 4 + 6)/3 = 4$*. Therefore, we have*

$$\mathbb{E}(d) = \sum_{k=1}^{\infty} [3(k-1) + 4]2^{-k} = 3\sum_{k=0}^{\infty} k2^{-k} + 4\sum_{k=1}^{\infty} 2^{-k}.$$

The geometric distribution $G(p)$ *has expectation* $1/p$*, and* $\sum_{k=0}^{\infty} k2^{-k} = 2$*. As for* $\sum_{k=1}^{\infty} 2^{-k} = 1/2/[1 - (1/2)] = 1$*. As a result, we have* $\mathbb{E}(d) = 3 * 2 + 4 * 1 = 10$*.*

8 Coupon Collector Problem

Suppose there are n different types of coupons, and a coupon is chosen at random at each trial. Each random coupon is equally likely be of any of the n types, and the random choices of the coupons are mutually independent. To collect on of each type of coupon, at least how many trails are required?

Let X be the number of trials required to collect at least one of each type of coupon. We determine $\mathbb{E}[X]$. Let X_i be the number of additional trials for another new type of coupon while exactly $i-1$ different types of coupon have been collected, $1 \le i \le n$ and $X = \sum_{i=1}^{n} X_i$. It's obvious, X_i has geometric distribution with parameter p_i. To be success, it requires to collect a type of coupon which is different from previous $i-1$ types. The probability is $p_i = 1-(i-1)/n$.

At the beginning, no coupon is collected. To collect a new type of coupon, only one additional trail is required, and the probability of being success is $Pr(X_1 = 1) = p_1 = 1$.

Let's compute the expectation for X

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} \frac{1}{p_i} = \sum_{i=1}^{n} \frac{n}{n-i+1} = nH_n.$$

Lemma 8.1 (Bounded Harmonic Number)**.** *The harmonic number $H_n = \sum_{i=1}^{n} 1/i$ satisfies*

$$H_n = \ln n + \Theta(1).$$

Proof. Because $f(x) = 1/x$ is monotonically decreasing, we apply the geometric properties of an integral operation over $1/x$ and have

$$\sum_{k=2}^{n} \frac{1}{k} \le \ln n = \int_1^n \frac{1}{x} dx \le \sum_{k=1}^{n} \frac{1}{k},$$

hence $\ln n \le H_n \le \ln n + 1$, we end the proof. □

We analysis the probability that X derivates from its expectation $nH_n = n\ln n + \Theta(n)$ by amount of cn, where c is real constant. Let ϵ_i^r denote the event that coupon type i is not collected in the first r trials. Therefore,

$$Pr(\epsilon_i^r) = (1-\frac{1}{n})^r \le e^{-r/n}.$$

Suppose the number of trials X required is greater than $r = \beta n \ln n$, then at least one of the events $\{\epsilon_i^r\}_{i=1}^n$ must happen. Apply the union bound, we can write

$$Pr(X > r) = Pr(\cup_{i=1}^n \epsilon_i^r) \le \sum_{i=1}^{n} Pr(\epsilon_i^r) \le ne^{-r/n} = n^{-\beta+1}.$$

When $\beta = 2$, $Pr(X > 2n\ln n) \le 1/n$.

9 Randomized Selection

10 Randomized Median Algorithm

Analysis the probability that the randomized median algorithm fails, including 3 reasons:

- ε_1: $|\{r \in R | r \leq m\}| < 1/2n^{3/4} - \sqrt{n}$,
- ε_2: $|\{r \in R | r \geq m\}| < 1/2n^{3/4} - \sqrt{n}$,
- ε_3: $|C| > 4n^{3/4}$.

Therefore, $Pr(Fails) = Pr(\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3) = Pr(\varepsilon_1) + Pr(\varepsilon_2) + Pr(\varepsilon_3)$

11 Tail Bounds

Let $\{X_i\}_{i=1}^n$ be independent *Poisson trails* s.t. for $1 \leq i \leq n$, $Pr(X_i = 1) = p_i$, where $0 < p_i < 1$. Then, for $X = \sum_{i=1}^n X_i$, $\mu = \mathbb{E}(X) = \sum_{i=1}^n p_i$. If $p_i = p > 0, \forall 1 \leq i \leq n$, $\{X_i\}_{i=1}^n$ are a.k.a *Bernoulli trails*, and X is a *Binomial* r.v.

To bound the probability that a random variable deviates from its expectation, many useful techniques are developed, e.g. Markov's inequality and Chebyshev's inequality. There are several questions regarding the deviation of X from its expectation μ:

- given $\delta > 0$, what's the probability that X exceeds $(1+\delta)\mu$?
- given a small $\epsilon > 0$ (e.g. 0.01), how large need δ be s.t. $Pr[X \geq (1+\delta)\mu] \leq \epsilon$?

To answer these questions, the *Chernoff bounds* was proposed. They are derived using the *Moment Generating Function* (MGF), and extremely useful in designing & analyzing randomized algorithms.

Algorithm 3 Randomized Median Algorithm

Input: A set S of n elements are a totally ordered university

Output: The median element in S, denoted m

$R \leftarrow$ uniformly sample $\lceil n^{3/4} \rceil$ elements from S with replacement
Sort R
$d \leftarrow$ the $\lceil 1/2n^{3/4} - \sqrt{n} \rceil$th smallest element in R
$u \leftarrow$ the $\lceil 1/2n^{3/4} + \sqrt{n} \rceil$th smallest element in R
$C \leftarrow \emptyset$ ▷ $C = \{x \in S | d \leq x \leq u\}$
$n_d, n_u \leftarrow 0$ ▷ $n_d = |\{x \in S | x < d\}|$, $n_u = |\{x \in S | x > u\}|$
function COMPARE(d, u, S)
    for $i \leftarrow 1$ to $|S|$ **do**
        $n_d \leftarrow n_d + 1$, if $S[i] < d$
        $n_u \leftarrow n_u + 1$, if $S[i] > u$
        $C \leftarrow C \cup \{S[i]\}$, if $d \leq S[i] \leq u$
    end for
    if $n_d > n/2$ or $n_u > n/2$ or $|C| > 4n^{3/4}$ **then**
        return FAIL
    else if $|C| \leq 4n^{3/4}$ **then**
        Sort C
        return $m \leftarrow (\lfloor n/2 \rfloor - n_d + 1)$th element in C
    end if
end function

Definition 11.1 (Moment Generating Function)**.** *The expectation of e^{tX} is the moment generating function of X, and denoted as $M_X(t) = \mathbb{E}[e^{tX}], \forall t > 0$.*

Proposition 11.1. *If $\{X_i\}_{i=1}^n$ are independent, and $Y = \sum_{i=1}^n c_i X_i, \forall c_i \in \mathbb{R}$, then the PDF of Y is the convolution of the PDF of $\{X_i\}_{i=1}^n$, and its MGF*

$$M_Y(t) = \prod_{i=1}^{n} M_{X_i}(c_i t), \forall t \in \mathbb{R}. \tag{1}$$

Proposition 11.2. $\mathbb{E}[X^n] = M_X^{(n)}(0)$.

Theorem 11.1 (Markov's Inequality)**.** *If X is a random variable with expectation μ_X, then*

$$Pr(X \geq t) \leq \mu_X / t, \forall t > 0. \tag{2}$$

Proof. According to the definition of expectation of X, we have

$$\begin{aligned} \mu_X &= \mathbb{E}[X] = \sum_{\forall x} x Pr(X = x) \\ &= \sum_{\forall x \geq t} x Pr(X = x) + \sum_{\forall x < t} x Pr(X = x) \\ &\geq \textstyle\sum_{\forall x \geq t} t Pr(X = x) = t Pr(X \geq t). \end{aligned}$$

Divide both sides by t, we get $Pr(X \geq t) \leq \mathbb{E}[X]/t, \forall t > 0$. □

Theorem 11.2 (Chebyshev's Inequality)**.** *If X is a random variable with variance σ_X^2, then*

$$Pr(|X - \mu_X| \geq t\sigma_X) \leq 1/t^2, \forall t > 0. \tag{3}$$

Proof. Let $Y = (X - \mu_X)^2$, then $\mathbb{E}(X - \mu_X)^2 = \sigma_X^2$. Applying Markov's inequality, we get

$$Pr(|X - \mu_X| \geq t) = Pr(Y \geq t^2) \leq \mu_Y / t^2 = \mathbb{E}(X - \mu_X)^2 / t^2 = (\sigma_X / t)^2,$$

then $Pr(|X - \mu_X| \geq t\sigma_X) \leq 1/t^2$. □

Problem 11.1. *Votes are misrecorded w/p $p = 0.02$, and there are 10,000 votes, what's an upper bound on the probability that 4% or more of the votes are misrecorded?*

Solution 11.1. *Let $X_i = 1$ if vote i is misrecorded, $X_i = 0$ otherwise. It's a Bernoulli r.v with parameter $p = 2\%$. Let $X = \sum_{i=1}^{n} X_i$ be a binomial r.v, i.e. $X \sim B(n,p)$ where $n = 10,000$. Therefore, $\mathbb{E}[X] = np = 200$, $Var[X] = np(1-p) = 196$. Then, $Pr(X \geq 4\%n) = Pr(X \geq 400) = Pr(X - \mathbb{E}[X] \geq 200)$. According to the Chebyshev's inequality,*

$$Pr(X - \mathbb{E}[X] \geq 200) \leq \frac{Var[X]}{200^2} = \frac{196}{200^2} = 0.0049.$$

Problem 11.2. *Let X_i be i.i.d r.v.s, $1 \leq i \leq n$. Given $\mu = \mathbb{E}(X_i)$ and $\sigma^2 = Var(X_i)$. Suppose $X = \sum_{i=1}^{n} X_i/n$, show the best upper bound for $Pr(X \geq \delta)$, if $\delta > \mu$.*

Proof. Because X_i are iid, then $\mathbb{E}(X) = \sum_{i=1}^{n} \mathbb{E}(X_i)/n = \mu$, $\sigma_X^2 = Var(X_i)/n = \sigma^2/n$. Therefore, we have

$$\begin{aligned} Pr(X \geq \delta) &= Pr(X - \mu_X \geq \delta - \mu_X) \leq Pr(|X - \mu_X| \geq (\delta - \mu_X)) \\ &\leq \mathrm{Var}(X)/(\delta - \mu_X)^2 = \tfrac{\sigma^2}{n(\delta-\mu)^2}. \end{aligned}$$

□

11.1 Chernoff Bounds

Theorem 11.3 (Upper Chernoff Bounds on Poisson Trials)**.** *Given n independent Poisson trials $\{X_i\}_{i=1}^{n}$ with $Pr[X_i = 1] = p_i \in (0,1)$, $1 \leq i \leq n$, and the sum $X = \sum_{i=1}^{n} X_i$ has its expectation $\mu = \mathbb{E}(X) = \sum_{i=1}^{n} p_i$. These bounds on the tail probabilities*

$$Pr[X \geq (1+\delta)\mu] \leq \left[\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right]^{\mu}, \forall \delta > 0 \tag{4}$$

$$Pr[X \geq (1+\delta)\mu] \leq e^{-\mu\delta^2/3}, \forall \delta \in (0,1] \tag{5}$$

$$Pr[X \geq R] \leq 2^{-R}, R \geq 6\mu \tag{6}$$

hold.

The first bound is the strongest one, and from it we derive the others, which are easier in statement and computing.

Proof. It's obvious, $Pr[X \geq (1+\delta)\mu] = Pr[e^{tX} \geq e^{t(1+\delta)\mu}]$, $\forall t > 0$. According to Markov's inequality, $Pr[e^{tX} \geq e^{t(1+\delta)\mu}] \leq \mathbb{E}[e^{tX}]/e^{t(1+\delta)\mu}$. Because $\{X_i\}_{i=1}^n$ are independent, therefore $\{e^{tX_i}\}_{i=1}^n$ are independent as well. Therefore,

$$\mathbb{E}[e^{tX}] = \prod_{i=1}^{n} \mathbb{E}[e^{tX_i}] = \prod_{i=1}^{n} [p_i e^t + (1-p_i)] = \prod_{i=1}^{n} [1 + p_i(e^t - 1)].$$

With the fact that $e^x \geq 1 + x$ and set $x = p_i(e^t - 1)$, we have

$$\prod_{i=1}^{n} [1 + p_i(e^t - 1)] \leq \prod_{i=1}^{n} \exp\Big(p_i(e^t - 1)\Big) = \exp\bigg(\sum_{i=1}^{n} p_i(e^t - 1)\bigg) = \exp((e^t - 1)\mu).$$

Thus, $Pr[e^{tX} \geq e^{t(1+\delta)\mu}] \leq \mathbb{E}[e^{tX}]/e^{t(1+\delta)\mu} \leq \exp((e^t - 1)\mu - t(1+\delta)\mu)$. To obtain a tight upper bound, we minimize the RHS of the inequality. Let $f(t) = (e^t - 1)\mu - t(1+\delta)\mu$, we minimize it by solving $f'(t) = \mu e^t - (1+\delta)\mu = 0$, and get $t = \ln(1+\delta) > 0$, where $f(t)$ reaches its minimum since $f''(t) = \mu e^t > 0$. Plugging it to the above inequality gives

$$Pr[X \geq (1+\delta)\mu] \leq \exp(\delta\mu - \ln(1+\delta)(1+\delta)\mu) = \left[\frac{e^\delta}{(1+\delta)^{1+\delta}}\right]^\mu.$$

We show $[\frac{e^\delta}{(1+\delta)^{1+\delta}}]^\mu \leq e^{-3\mu\delta^2/3}$. We define $g(\delta) = \delta - (1+\delta)\ln(1+\delta) + \delta^2/3$, then $g'(\delta) = -\ln(1+\delta) + 2/3\delta$, $g''(\delta) = 2/3 - 1/(1+\delta)$. With $g''(1/2) = g'(0) = g(0) = 0$, $g'(1) = 2/3 - \ln 2 < 0$, we have

- $\delta \in (0, 1/2)$: $g''(\delta) < 0$, $g'(\delta) < g'(0) = 0$,
- $\delta \in (1/2, 1)$: $g''(\delta) > 0$, $g'(\delta) < g'(1) < 0$.

Above all, $g'(\delta) \leq 0, \forall \delta \in [0,1]$, thus $g(\delta) \leq 0$, and $Pr[X \geq (1+\delta)\mu] \leq e^{-\mu\delta^2/3}$.

To prove the last bound, let $R = (1+\delta)\mu \geq 6\mu$, then $\delta \leq 5$, and applying the first bound

$$Pr(X \geq R) \leq \left[\frac{e^\delta}{(1+\delta)^{1+\delta}}\right]^\mu \leq \left[\frac{e}{1+\delta}\right]^{\mu(1+\delta)} \leq \left[\frac{e}{6}\right]^{\mu(1+\delta)} \leq 2^{-R}.$$

□

Theorem 11.4 (Lower Chernoff Bounds on Poisson Trials)**.** *Given n independent Poisson trials $\{X_i\}_{i=1}^n$ with $Pr[X_i = 1] = p_i \in (0,1)$, $1 \le i \le n$, and the sum $X = \sum_{i=1}^n X_i$ has its expectation $\mu = \mathbb{E}(X) = \sum_{i=1}^n p_i$. The bounds on the tail probabilities*

$$Pr[X \le (1-\delta)\mu] \le \Big[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\Big]^{\mu} \tag{7}$$

$$Pr[X \le (1-\delta)\mu] \le e^{-\mu\delta^2/2} \tag{8}$$

hold if $0 < \delta < 1$.

Proof. Apply Markov's inequality with $t < 0$, we get

$$\begin{aligned} Pr[X \le (1-\delta)\mu] &= Pr[e^{tX} \ge e^{t(1-\delta)\mu}] \le \mathbb{E}[e^{tX}]e^{-t(1-\delta)\mu} \\ &= \textstyle\prod_{i=1}^n \mathbb{E}[e^{tX_i}]e^{-t(1-\delta)\mu} \le \prod_{i=1}^n e^{p_i(e^t-1)}e^{-t(1-\delta)\mu} \\ &= e^{(e^t-1)\mu - t(1-\delta)\mu}. \end{aligned}$$

Let $t = \ln(1-\delta) < 0$, we show that

$$Pr[X \le (1-\delta)\mu] \le \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu}.$$

To show $\Big[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\Big]^{\mu} \le e^{-\mu\delta^2/2}$ equivalently to show $\mu[-\delta - (1-\delta)\ln(1-\delta)] \le -\mu\delta^2/2$. Let $f(\delta) = -\delta - (1-\delta)\ln(1-\delta) + \delta^2/2$, we have $f(0) = 0$, $f'(\delta) = \ln(1-\delta) + \delta$, and $f''(\delta) = 1 - 1/(1-\delta) < 0$. Since $f'(0) = 0$, so $f'(\delta) \le 0$ in $[0,1)$ and $f(\delta) \le 0$. □

Theorem 11.5 (Chernoff Bounds on Rademacher Distribution)**.** *Let $\{X_i\}_{i=1}^n$ be the independent r.v.s with $Pr(X_i = 1) = Pr(X_i = -1) = 1/2$, $\{X_i\}$ are a.k.a Rademacher random variables. Let $X = \sum_{i=1}^n X_i$. For any $c > 0$*

$$Pr(X \ge c) \le e^{-c^2/2n} \tag{9}$$

$$Pr(X \le -c) \le e^{-c^2/2n} \tag{10}$$

$$Pr(|X| \ge c) \le 2e^{-c^2/2n} \tag{11}$$

Proof. Applying Markov's inequality, we have

$$Pr(X \ge c) = Pr(e^{tX} \ge e^{tc}) \le \mathbb{E}[e^{tX}]/e^{tc} = \prod_{i=1}^n \mathbb{E}[e^{tX_i}]/e^{tc}, \forall t > 0.$$

For any $t \in \mathbb{R}$, $\mathbb{E}[e^{tX_i}] = e^t/2 + e^{-t}/2$, we merge them using the Taylor's series

$$\begin{aligned} e^t &= 1 + t + \tfrac{t^2}{2!} + \tfrac{t^3}{3!} + \cdots, \\ e^{-t} &= 1 - t + \tfrac{t^2}{2!} - \tfrac{t^3}{3!} + \cdots, \end{aligned}$$

then $\mathbb{E}[e^{tX_i}] = 1 + \frac{t^2}{2!} + \frac{t^4}{4!} + \cdots = \sum_{i=0}^{\infty} \frac{t^{2i}}{(2i)!}$. The product of the even terms of $(2i)!$ is $\prod_{n=1}^{i}(2n) = 2^i i! \leq (2i)!$ and $\mathbb{E}[e^{tX_i}] \leq \sum_{i=0}^{\infty} \frac{(t^2/2)^i}{i!} = e^{t^2/2}$. Thus, $Pr(X \geq c) \leq e^{nt^2/2 - tc}$. Let $\underline{t = c/n}$, then $nt^2/2 - tc = -c^2/2n$, $Pr(X \geq c) \leq e^{-c^2/2n}$.

The symmetry of X leads to $Pr(X \leq -c) \leq e^{-c^2/2n}$ and $Pr(|X| \geq c) \leq 2e^{-c^2/2n}$. □

Theorem 11.6 (Chernoff Bounds on Bernounlli Distribution)**.** *Let $\{Y_i\}_{i=1}^n$ be the independent random variables with $Pr(Y_i = 1) = Pr(Y_i = 0) = 1/2$, $\{Y_i\}$ are a.k.a Bernounlli random variables with $p = 1/2$. Let $Y = \sum_{i=1}^n Y_i$, then $\mu = n/2$.*

$$Pr(Y \leq \mu - c) \leq e^{-2c^2/n}, \forall 0 < c < \mu \tag{12}$$

$$Pr(Y \leq (1-\delta)\mu) \leq e^{-\delta^2 \mu}, \forall 0 < \delta < 1 \tag{13}$$

Proof. Let $X_i = 2Y_i - 1$, it has the Rademacher distribution, $X = \sum_{i=1}^n X_i$, then

$$Pr(X \leq -a) \leq e^{-a^2/2n}, \forall a > 0.$$

Because $Pr(X \leq -a) = Pr(2Y - n \leq -a) = Pr(Y \leq \mu - a/2)$, let $c = a/2$, we have

$$Pr(Y \leq \mu - c) \leq e^{-4c^2/2n} = e^{-2c^2/n}.$$

Let $c = \delta\mu$, we have $Pr(Y \leq (1-\delta)\mu) \leq e^{-2\delta^2\mu^2/n} = e^{-\delta^2\mu}$. □

Corollary 11.1. *Let $X_1, X_2, \ldots, X_n$ be independent Poisson trials such that $Pr(X_i) = p_i$. Let $X = \sum_i X_i$ and $\mu = \mathbb{E}[X]$. For $0 \leq \delta < 1$,*

$$Pr(|X - \mu| \geq \delta\mu) \leq 2e^{-\mu\delta^2/3}. \tag{14}$$

Corollary 11.2. *Let $X_1, X_2, \ldots, X_n$ be independent Bernoulli r.v.s such that $Pr(X_i = 1) = Pr(X_i = 0) = 1/2$. Let $X = \sum_i X_i$ and $\mu = \mathbb{E}[X] = n/2$. For any $0 < c < \mu$,*

$$Pr(|X - \mu| \geq c) \leq 2e^{-2c^2/n}. \tag{15}$$

Problem 11.3. *Assigning n individuals to two groups: control group and treatment group. Each individual is assigned to the control group with probability $P(X_i = 1) = 1/2$. Argue that the size of the control group is $n/2 \pm O(\sqrt{n \ln n})$ with probability $\geq 1 - 2/n$.*

Proof. Let X_i denote the ith individual is assigned to the control group. Then $Pr(X_i = 1) = Pr(X_i = 0) = 1/2$, $X = \sum_{i=1}^{n} X_i$ is the size of the control group, and $\mu = \mathbb{E}[X] = n/2$. According to the Chernoff bounds $Pr(|X - \mu| \geq a) \leq 2e^{-2a^2/n}, \forall 0 < a < \mu$. The probability $Pr(|X - \mu| \leq c\sqrt{n \ln n}) = 1 - Pr(|X - \mu| \geq c\sqrt{n \ln n})$. Let $a = c\sqrt{n \ln n}$, plugging into the above inequality gives

$$Pr(|X - \mu| \leq c\sqrt{n \ln n}) \geq 1 - 2e^{-2c^2 \ln n} = 1 - 2n^{-2c^2}.$$

Assign $c = 1/\sqrt{2}$, then $Pr(|X - \mu| \leq c\sqrt{n \ln n}) \geq 1 - 2/n$. □

11.2 Applications

11.2.1 Set Balancing

The problem *set-balancing* is a.k.a *two-coloring a family of vectors*. Given $A \in \{0,1\}^{n \times m}$, find a column vector $b \in \{-1,1\}^m$ to minimize $\|Ab\|_\infty$. It arises in statistical experiment designs. Each column of A represents an subject in the experiment, and each row represents a feature. The vector b partitions subjects to two disjoint groups: the treatment group and the control group, such that the number of subjects with each feature is roughly the same.

A randomized algorithm to search the vector b is independently, randomly choosing each entry from $\{-1,1\}$, then $\|Ab\|_\infty$ can reach $O(\sqrt{m \ln n})$.

Theorem 11.7. *For a random vector b with entries independently and with equal probability chosen from $\{-1,1\}$, then*

$$Pr(\|Ab\|_\infty \geq \sqrt{4m \ln n}) \leq \frac{2}{n}. \tag{16}$$

Proof. For the ith row of A, $1 \leq i \leq n$, the dot product of A's ith row and the random

vector b

$$Z_i = \sum_{j=1}^{m} a_{ij} b_j.$$

Suppose there are k non-zero elements in the row vector of A, Z_i becomes the sum of k independent random variables with $Pr(b_j = 1) = Pr(b_j = -1) = 1/2$. Applying the Chernoff bounds on Rademacher distribution of two tails, we have

$$Pr(|Z_i| \geq \sqrt{4m \ln n}) \leq 2e^{-4m \ln n / 2k} \leq 2e^{-2 \ln n} = \frac{2}{n^2},$$

since $m \geq k$. Considering all rows of A,

$$\begin{aligned} Pr(\|Ab\|_\infty \geq \sqrt{4m \ln n}) &= Pr\Big(\cup_{i=1}^{n} \{|Z_i| \geq \sqrt{4m \ln n}\}\Big) \\ &\leq \sum_{i=1}^{n} Pr(|Z_i| \geq \sqrt{4m \ln n}) \leq \tfrac{2}{n}. \end{aligned}$$

□

11.2.2 Permutation Routing on the Hypercube

A n-dimensional hypercube (or n-cube) is a directed graph $G = (V, E)$ with $N = 2^n$ nodes s.t node i is immediately connected to j iff $h(i, j) = 1$, where h is the Hamming distance of i and j in terms of binary representation of length n. The total number of directed edges in G is $|E| = 2nN$ because the out-degree of each node is exact n.

Permutation routing problem is a.k.a *oblivious routing problem*, where each node in G initially has one packet to deliever and is also the destination of exact one packet. Let $d(i)$ be the destination of the packet of node i. Table 1 demonstrates a permutation routing problem on 3-cube. The two rows present the source and destination node, both are written in the form of binary representation. One row is just a permutation of another row, and it's where the name of the problem comes from.

A *routing algorithm* assigns each pair of nodes a directed path – a sequence of directed edges – connecting the pair. During the routing of packets, one edge may be on the path of more than one packet and one edge can process only one packet per step, it will cause

Table 1: Permutation Routing Problem

i	$0\underline{0}0$	$\underline{0}01$	$01\underline{0}$	$\underline{0}11$	$\underline{1}00$	$1\underline{0}1$	$\underline{1}10$	$11\underline{1}$
$d(i)$	$01\underline{0}$	$\underline{1}01$	$01\underline{1}$	$\underline{1}11$	$\underline{0}00$	$1\underline{1}0$	$\underline{0}01$	$11\underline{0}$

congestion and *bottlenecks*. To resolve the problem, the routing algorithm should also specify a *queueing policy* to order packets in the queue and allows at most one packet to pass through the directed edge in each step. If the permutation routing algorithm routes packets only based on their destination, it's called *oblivious routing algorithm*.

The performance of a routing algorithm can be measured with the *maximum time*, or the *number of parallel steps* required to routing an arbitrary permutation routing problem.

Algorithm 4 n-Cube Bit-Fixing Routing Algorithm

Input: source $i = (a_1, \ldots, a_n)$ and destination $d(i) = (b_1, \ldots, b_n)$ of the packet

```
1: function ROUTE(i, d(i))
2:     for k ← 1 to n do
3:         if a_k ≠ b_k then
4:             traverse the edge (b_1,...,b_{k-1},a_k,...,a_n) → (b_1,...,b_{k-1},b_k,a_{k+1},...,a_n)
5:         end if
6:     end for
7: end function
```

Theorem 11.8. *For arbitrary deterministic oblivious permutation routing algorithm on an n-cube with $N = 2^n$ nodes each of out-degree n, there is an instance of permutation routing requiring $\Omega(\sqrt{N/n})$ steps to finish.*

Lemma 11.1. *Let the route of packet v_i be $\rho_i = (e_1, e_2, \ldots, e_K)$. Let S_i^I be the set of packets (other than v_i) whose routes intersect at least one of $\{e_1, e_2, \ldots, e_K\}$ in ρ_i in Phase I. Then, the delay of v_i is at most $|S_i^I|$.*

Algorithm 5 Two Phase Routing Algorithm

Input: source $i = (a_1, \ldots, a_n)$ and destination $d(i) = (b_1, \ldots, b_n)$ of the packet

1: Phase I: randomly select $\sigma(i)$, invoke ROUTE$(i, \sigma(i))$

2: Phase II: invoke ROUTE$(\sigma(i), d(i))$

Proof. Suppose n is even, and for packet v_i with its source and destination of forms

$$\begin{aligned} i = \quad ab = \quad & a_1, \ldots, a_{n/2}, b_1 \ldots, b_{n/2}, \\ d(i) = \quad ba = \quad & b_1, \ldots, b_{n/2}, a_1, \ldots, a_{n/2}. \end{aligned}$$

There is a node with address bb, and the routing algorithm will alway use it to route a pair of $ab \to ba$. However, among N nodes, $2^{n/2} = \sqrt{N}$ have an address of the form bb. The routing algorithm routes at most n packets for each node at each step, because each node has n out-going edges. Therefore, routing all packets requires at least $N/(n\sqrt{N})$ steps, it's $\sqrt{N}/n$. □

The time to deliver packet v_i is at most n plus the delayed steps for queueing at intermediate node in ρ_i. We need to compute the expected delay.

We analysis the two phase algorithm, and fix the packet v_i with a route $\rho_i = (e_1, e_2, \ldots, e_K)$ of length K. Let $H_j = 1$ if a different ρ_j intersects ρ_i; otherwise $H_j = 0$. Since the intermedia destination is randomly chosen, $\{H_j\}$ are independent Poisson trials. According to the definition, we have

$$|S_i^I| = \sum_{j=1}^{N} H_j.$$

For an edge e in G, let $T(e)$ be the number of routes cross edge e. Then, the number of routes intersects ρ_i should not be larger than the number of routes passing through at least one edge in ρ_i, i.e.

$$|S_i^I| \le \sum_{t=1}^{K} T(e_t).$$

Therefore, we have $\mathbb{E}[|S_i^I|] \le \sum_{t=1}^{K} \mathbb{E}[T(e_t)]$, where K is also a random variable. Suppose $i = (a_1, a_2, \ldots, a_n)$, and $\sigma(i) = (b_1, b_2, \ldots, b_n)$, the times taken to deliver packet v_i from i to

$\sigma(i)$ is the Hamming distance between i and its intermediate destination $\sigma(i)$. Let $Z_t = 1$ if $a_t \neq b_t$; otherwise $Z_t = 0$, $1 \leq t \leq n$. Further, $K = \sum_{t=1}^{n} Z_t$ and $\mathbb{E}[K] = n/2$ for $\{Z_t\}$ are independent Bernoulli random variables with parameter $1/2$.

Besides, $\forall e \in E$, $T(e)$s are the same, and thus independent from K. We get

$$\mathbb{E}[|S_i^I|] = \mathbb{E}[\mathbb{E}[|S_i^I|\Big|K]] \leq \mathbb{E}[\mathbb{E}[\sum_{t=1}^{K} T(e_t)\Big|K]] = \mathbb{E}[KT(e)] = \mathbb{E}[K]\mathbb{E}[T(e)] = \frac{n}{2}\mathbb{E}[T(e)].$$

According to the definition

$$T(e) = \sum_{k=1}^{N} I(\rho_k \text{ crosses } e) = \sum_{k=1}^{N} Pr(\rho_k \text{ crosses } e).$$

Let $e = (b_1,\ldots,b_{j-1},a_j,a_{j+1},\ldots,a_n) \rightarrow (b_1,\ldots,b_{j-1},b_j,a_{j+1},\ldots,a_n)$ corresponding the bit-fixing on the jth entry. To cross the edge e, the origin of the route must be $(*,*,\ldots,*,a_j,\ldots,a_n)$ and the destination of the route be $(b_1,\ldots,b_{j-1},b_j,*,*,\ldots,*)$, where $*$-bit's value does not matter.

The possible number of routes with the specific form of origins are 2^{j-1}, and for any fixed origin (route), the probability that the destination of the route has the specific form is $Pr(b_1,\ldots,b_{j-1},b_j,*,*,\ldots,*) = 2^{n-j}/2^n = 2^{-j}$, then

$$\mathbb{E}[T(e)] = 2^{j-1}2^{-j} = 1/2, \tag{17}$$

$$\mathbb{E}[|S_i^I|] \leq n/4. \tag{18}$$

Let $R = 3n/2 \geq 6\mathbb{E}[|S_1|]$, $Pr(|S_i^I| \geq R) = Pr(|S_1| \geq 3n/2) \leq 2^{-R} = 2^{-3n/2}$. Using union bound for N packets, we get

$$Pr\Big(\cup_{i=1}^{N}\{|S_i^I| \geq 3n/2\}\Big) \leq N2^{-3n/2} = 2^{-n/2}.$$

It's for the Phase I, and Phase II runs Phase I backward. Therefore, we can conclude that: with probability at least $1 - 2^{-n/2}$, all packets are delivered in at most $n + 3n/2 * 2 = 5n$ steps. The randomized 2-Phase routing algorithm can route all packets to their destination in $O(n)$ time with probability close to 1.

12 The Probability Method

The probability method is a way of proving the existence of an object. To prove the existence, there are two important principles:

- Simple Counting: constructing an appropriate probability space $\mathcal{S}$ of objects, and then show that the probability that an object in $\mathcal{S}$ with the required properties is selected is positive. Since the probability is positive, it must exist.
- Averaging Argument: a random variable – in a discrete probability space – must with a positive probability assume at least one value that is not greater than its expectation, and at least one value that is not smaller than its expectation.

It's applied to solve some complicated problems and requires many advanced techniques for constructing proof.

Theorem 12.1. *Given an undirected graph $G = (V, E)$ with n vertices and m edges. There is a partition of V into S and T, s.t. at least $m/2$ edges cross S and T. That is, there is a cut with value at least $m/2$.*

Proof. We construct the cut by randomly assigning each vertices into S and T. Let X_i denote whether an edge $e_i \in E$ crosses S and T, i.e. $\forall i = 1, 2, \ldots, m$

$$X_i = \begin{cases} 1, & e_i \text{ crosses } S \text{ and } T, \\ 0, & otherwise. \end{cases}$$

The probability that each edge crosses S and T is $1/2$, because each vertex is assigned to S or T is $1/2$, and their membership is mutually independent.

The value of the cut $V = S \cup T$ should be $C(S,T) = \sum_{i=1}^{m} X_i$ and its expectation

$$\mathbb{E}[C(S,T)] = \sum_{i=1}^{m} \mathbb{E}[X_i] = m/2.$$

According to the expectation argument, there exists a cut of V in G, s.t $C(S,T) \geq m/2$. □

To derandomized the randomized algorithm to construct a cut from G, we bound the probability to find a cut with value at least $m/2$. Consider $C(S,T)$ as a random variable, it's clear $C(S,T) \leq m$, we derive from the expectation

$$\begin{aligned} m/2 = \mathbb{E}[C(S,T)] &= \textstyle\sum_{i=1}^{m} iPr(C(S,T)=i) \\ &= \textstyle\sum_{i=1}^{m/2-1} iPr(C(S,T)=i) + \sum_{i=m/2}^{m} iPr(C(S,T)=i) \\ &\leq (m/2-1)\textstyle\sum_{i=1}^{m/2-1} Pr(C(S,T)=i) + m\sum_{i=m/2}^{m} Pr(C(S,T)=i) \\ &= (m/2-1)Pr(C(S,T)<m/2) + mPr(C(S,T)\geq m/2) \\ &= (m/2-1)(1-Pr(C(S,T)\geq m/2)) + mPr(C(S,T)\geq m/2). \end{aligned}$$

Let $p = Pr(C(S,T) \geq m/2)$, we have $m/2 \leq (m/2-1)(1-p) + mp$, then

$$p \geq \frac{1}{m/2+1}.$$

It implies that we sample at most $m/2+1$ partitions, we can get a cut of V with value at least $m/2$.

Determining a SAT formula has a solution is NP-hard, we find a weak solution which satisfies as many clauses as possible.

Theorem 12.2. *Given a set of m clauses, and assume the ith clause has n_i literals, $\forall i \leq m$. Let $n = \max_i n_i$, there is a truth assignment that satisfies at least $m(1-2^{-n})$ clauses.*

Proof. Let X_i be the random variable denoting whether the ith clause is satisfied given a random truth assignment. At random, we independently and uniformly assign values $\{T,F\}$ to all variables in the SAT formula. We can get the probability that the ith clause is satisfied, i.e. $Pr(X_i = 1) = 1-2^{-n_i}$, therefore $\mathbb{E}[X_i] = Pr(X_i = 1)$. The number of clauses that are satisfied by the assignment is $X = \sum_{i=1}^{m} X_i$. Its expectation

$$\mathbb{E}[X] = \sum_{i=1}^{m} E[X_i] = \sum_{i=1}^{m} (1-2^{-n_i}) \geq m(1-2^{-n}).$$

The expectation argument shows that there exists a truth assignment s.t at least $m(1-2^{-n})$ clauses are satisfied. □

12.1 Second Moment Method

Theorem 12.3. *If X is a non-negative integer-valued random variable, then*

$$Pr(X = 0) \leq \frac{Var(X)}{(\mathbb{E}[X])^2}. \tag{19}$$

Proof. Apply Chebyshev's inequality,

$$Pr(X = 0) \leq Pr(|X - \mathbb{E}[X]| \geq \mathbb{E}[X]) \leq \frac{\mathbb{E}[X - \mathbb{E}[X]]}{(\mathbb{E}[X])^2} = \frac{Var(X)}{(\mathbb{E}[X])^2}.$$

□

12.2 Conditional Expectation Inequality

Theorem 12.4. *Let $X = \sum_{i=1}^{n} X_i$ where each X_i is a Bernoulli random variable. Then*

$$Pr(X > 0) \geq \sum_{i=1}^{n} \frac{Pr(X_i = 1)}{\mathbb{E}[X|X_i = 1]}. \tag{20}$$

Proof. Let $Y = 1/X$ if $X > 0$, and $Y = 0$ otherwise. It's obvious, $Pr(X > 0) = \mathbb{E}[XY]$. Then

$$\begin{aligned}
\mathbb{E}[XY] &= \sum_{i=1}^{n} \mathbb{E}[X_i Y] = \sum_{i=1}^{n} \Big[\mathbb{E}[X_i Y|X_i = 0]Pr(X_i = 0) + \mathbb{E}[X_i Y|X_i = 1]Pr(X_i = 1)\Big] \\
&= \sum_{i=1}^{n} \mathbb{E}[X_i Y|X_i = 1]Pr(X_i = 1) = \sum_{i=1}^{n} \mathbb{E}[Y|X_i = 1]Pr(X_i = 1) \\
&= \sum_{i=1}^{n} \mathbb{E}[1/X|X_i = 1]Pr(X_i = 1) = \sum_{i=1}^{n} [\sum_x \tfrac{Pr[X=x|X_i=1]}{x}]Pr(X_i = 1) \\
&\geq \sum_{i=1}^{n} \tfrac{Pr(X_i=1)}{\sum_x xPr[X=x|X_i=1]} = \sum_{i=1}^{n} \tfrac{Pr(X_i=1)}{\mathbb{E}[X|X_i=1]},
\end{aligned}$$

where the Jensen's inequality is used since $f(x) = 1/x$ is convex. □

12.3 Lovász Local Lemma

Suppose a large number of events in the probability space happens with probability less than 1, and there are independent from each other, there must exist a positive probability that none of these events occur. The Lovász local lemma relax the mutually independency with a weak partially independency, and arrives the same conclusion.

Definition 12.1 (Dependency Graph). *A dependency graph for a set of events $E_1, E_2, \ldots, E_n$ is a graph $G(V,E)$ such that $V = \{1,2,\ldots,n\}$, and for $i = 1,\ldots,n$, event E_i is mutually independent of the events $\{E_j | (i,j) \neq E\}$.*

Lemma 12.1 (Lovász Local Lemma). *Let $E_1, E_2, \ldots, E_n$ be a set of events, and assume that the following hold*

1. $Pr(E_i) \leq p < 1, \forall i = 1,2,\ldots,n$
2. *the degree of G is bounded by d*
3. $4pd \leq 1$

Then

$$Pr\Big(\bigcap_{i=1}^{n} \bar{E}_i\Big) > 0.$$

Proof. We note that $Pr\Big(\bigcap_{i=1}^{n} \bar{E}_i\Big) = \prod_{i=1}^{n} Pr(\bar{E}_i | \bigcap_{j=1}^{i-1} \bar{E}_j)$. Let $S \subset \{1,\ldots,n\}$, we prove by induction on $s = 0,1,\ldots,n-1$ that

$$Pr\Big(E_k | \bigcap_{j \in S} \bar{E}_j\Big) \leq 2p < 1, \forall k \notin S \tag{21}$$

if $|S| \leq s$. Based on it, we can proof $Pr\Big(\bigcap_{j \in S} \bar{E}_j\Big) > 0$.

If $s = 0$, or $S = \varnothing$, $Pr\Big(E_k | \bigcap_{j \in S} \bar{E}_j\Big) = Pr(E_k) \leq p < 2p$. Let's show that it's true for any non-empty S. Based on the statement in (21) for $s \geq 1$, we show $Pr\Big(\bigcap_{j \in S} \bar{E}_j\Big) > 0$.

If $s = 1$, we can immediate get $Pr(\bar{E}_i) = 1 - Pr(E_i) \geq 1 - p > 0$, which is not dependent on the statement in (21). Assuming (21) holds for $s > 1$, w.l.o.g let $S = \{1,2,\ldots,s\}$, we show

$$\begin{aligned} Pr\Big(\bigcap_{i=1}^{s} \bar{E}_i\Big) &= \prod_{i=1}^{s} Pr(\bar{E}_i | \bigcap_{j=1}^{i-1} \bar{E}_j) \\ &= \prod_{i=1}^{s} [1 - Pr(E_i | \bigcap_{j=1}^{i-1} \bar{E}_j)] \\ &\geq \prod_{i=1}^{s} (1 - 2p) > 0. \end{aligned} \tag{22}$$

We show that (21) holds for $s > 1$. Let $S_1 = \{j \in S | (k,j) \in E\}$, $S_2 = \{j \in S | (k,j) \notin E\}$, $F_S = \bigcap_{i \in S} \bar{E}_i$. If $S_2 = S$, E_k is mutually independent of the events $\bar{E}_j, \forall j \in S$, and thus

$$Pr(E_k | F_S) = Pr(E_k) \leq p \leq 2p.$$

For $|S_2| < s$, apply Bayes' theorem, we have

$$Pr(E_k|F_S) = \frac{Pr(E_k \cap F_S)}{Pr(F_S)} = \frac{Pr(E_k \cap F_{S_1} \cap F_{S_2}}{Pr(F_{S_1} \cap F_{S_2})} = \frac{Pr(E_k \cap F_{S_1}|F_{S_2})Pr(F_{S_2})}{Pr(F_{S_1}|F_{S_2})Pr(F_{S_2})} = \frac{Pr(E_k \cap F_{S_1}|F_{S_2})}{Pr(F_{S_1}|F_{S_2})}.$$

1. Denominator:

$$\begin{aligned} Pr(F_{S_1}|F_{S_2}) &= Pr(\textstyle\bigcap_{i \in S_1} \bar{E}_i | F_{S_2}) = 1 - Pr(\bigcup_{i \in S_1} E_i | F_{S_2}) \\ &\geq 1 - \textstyle\sum_{i \in S_1} Pr(E_i | F_{S_2}) = 1 - |S_1| 2p \geq 1 - 2dp \geq 1/2. \end{aligned}$$

   Using the fact that $|S_1| \leq d$ and $4dp \leq 1$, and the assumption $Pr(E_i|F_S) \leq 2p, |S| < s$.

2. Numerator:

$$Pr(E_k \cap F_{S_1}|F_{S_2}) \leq Pr(E_k|F_{S_2}) = Pr(E_k) \leq p.$$

Therefore, $Pr(E_k|F_S) \leq 2p$, $\forall S$ with $|S| \leq s$. Plug $s = n$ in (22), we get $Pr\Big(\bigcap_{i=1}^{n} \bar{E}_i\Big) > 0$. □

Lemma 12.2 (General Lovász Local Lemma)**.** *Let $G(V,E)$ be a dependency graph for events $E_1, E_2, \ldots, E_n$ in a probability space. Suppose that there exist $x_i \in [0,1], \forall 1 \leq i \leq n$ such that*

$$Pr(E_i) \leq x_i \prod_{(i,j) \in E} (1 - x_i).$$

Then

$$Pr\Big(\bigcap_{i=1}^{n} \bar{E}_i\Big) \geq \prod_{i=1}^{n} (1 - x_i).$$

Proof. Let $S \subset \{1, 2, \ldots, n\}$. We first prove by induction on $s = |S| = 0, 1, \ldots, n-1$ that

$$Pr(E_k | \textstyle\bigcap_{j \in S} \bar{E}_j) \leq x_i, \forall k \notin S.$$

The base case $s = 0$ or $S = \varnothing$ follows from the assumption on the probabilities $Pr(E_i)$. For $s \geq 1$, let $S_1 = \{j \in S | (k,j) \in E\}$, $S_2 = \{j \in S | (k,j) \notin E\}$, $F_S = \bigcap_{i \in S} \bar{E}_i$. Apply the definition of conditional probability,

$$Pr(E_k|F_S) = \frac{Pr(E_k \cap F_{S_1}|F_{S_2})}{Pr(F_{S_1}|F_{S_2})}.$$

1. Denominator: let $S_1 = \{j_1, j_2, \ldots, j_r\}$,

$$\begin{aligned} Pr(F_{S_1}|F_{S_2}) &= Pr(\cap_{j\in S_1}\bar{E}_j|F_{S_2}) \\ &= \prod_{t=1}^{r} Pr(\bar{E}_{j_t}|F_{S_2}\cap_{u=1}^{t-1}\bar{E}_{j_u}) \\ &= \prod_{t=1}^{r}\Big[1 - Pr(E_{j_t}|F_{S_2}\cap_{u=1}^{t-1}\bar{E}_{j_u})\Big] \\ &\le \prod_{t=1}^{r}\Big[1-x_{j_t}\Big] = \prod_{j\in S_1}\Big[1-x_j\Big]. \end{aligned}$$

2. Numerator:

$$Pr(E_k \cap F_{S_1}|F_{S_2}) \le Pr(E_k|F_{S_2}) = Pr(E_k) \le x_k \prod_{j\in S_1}\Big[1-x_j\Big].$$

It follows that

$$\begin{aligned} Pr(E_k|F_S) &\le \frac{x_k\prod_{j\in S_1}(1-x_j)}{\prod_{j\in S_1}(1-x_j)} = x_k, \\ Pr\Big(\bigcap_{i=1}^{n}\bar{E}_i\Big) &= \prod_{i=1}^{n} Pr(\bar{E}_i|\cap_{j=1}^{i-1}\bar{E}_j) \ge \prod_{i=1}^{n}(1-x_i). \end{aligned}$$

□

Problem 12.1. *You are given an instance of k-SAT with n clauses, where each clause has k literals. Show that if $k > \log n$, then this is a satisfiable formula.*

Proof. Let ϵ_i denote the event that clause i is not satisfied. There are k literals in each clause, if clause i is not satisfied, i.e. none of the k literals is given the correct assignment, and $Pr(\epsilon_i) = 2^{-k}$. To be a satisfiable formula, all clauses should be satisfied. Its probability $Pr(\bigcap_{i=1}^{n}\bar{\epsilon}_i) = 1 - Pr(\bigcup_{i=1}^{n}\epsilon_i) \ge 1 - \sum_{i=1}^{n} Pr(\epsilon_i) = 1 - n/2^k > 0, \forall k > \log n$. □

13 Derandomization

14 Data Structures

14.1 Treap

A treap is a randomized binary search tree (BST) in which each key is associated with a random priority. Each treap satisfies both *BST property* and *heap property*. The tree is

maintained such that it's a max-heap, i.e. each internal node's priority is greater than any of its children's priorities, and each node has larger key than that of its left branch, and smaller than that of its right branch.

Although it is not guaranteed to have height as $O(\log n)$, the idea of treap is to use randomization and binary heap property to maintain a balance with high probability. Its expected time complexity of search, insert and delete is $O(\log n)$. To make the related analysis convenient, we assume the keys and priorities are all distinct.

Theorem 14.1. *Given n pairs of keys and priorities, the shape of the treap is uniquely determined no matter what the ordering of insertation.*

Theorem 14.2. *The expected depth of any node x_i in the random treap T is $O(\log n)$.*

Proof. Let x_i denote the node with the ith largest search key, $d(x_i)$ denote the depth of node x_i, i.e. the number of proper ancestors (exclusive itself) down-up to the root node. Given a random treap T of n elements, x_1 is the node with the largest key, and x_n is the node with the smallest search key. To analysis the expected height, we define an indicator variable A_i^j presenting that x_j is a proper ancestor of x_i. Then, the depth of x_i is $d(x_i) = \sum_{j=1}^{n} A_i^j$, and $\mathbb{E}[d(x_i)] = \sum_{j=1}^{n} \mathbb{E}[A_i^j] = \sum_{j=1}^{n} Pr(A_i^j = 1)$.

To compute the expected depth requires to analysis the probability of one node being the proper ancestor of another node.

Lemma 14.1. *Given any two nodes x_i and x_j in T, $\forall i \neq j \leq n$, x_j is an ancestor of x_i iff x_j has the highest priority among all x_k where k is an index between i and j (inclusive).*

Proof. Suppose either x_i or x_j is the root of T, it's trivial to prove the statement because the root of T has the global highest priority.

Suppose neither x_i nor x_j is the root of T, but x_k is, $k \notin \{i, j\}$. There are two status for the relation of x_i and x_j. Either they fall into the same left or right subtree of x_k or belong to two different subtrees of x_k.

Assume they are in the same left or right subtree of x_k. Since the subtree of x_k is a smaller treap, the same lemma can be applied. And the empty treap is a trivial base case.

Assume they belong to different subtrees of x_k, w.l.o.g x_j is in the left subtree of x_k, then $i < k < j$. And x_j is not an ancestor of x_i and indeed it does't has the highest priority among $\{x_i, x_{i+1}, \ldots, x_j\}$, but x_k is an ancestor of x_i and has the highest priority among $\{x_i, x_{i+1}, \ldots, x_k\}$. □

Each node in $\{x_j, x_{j+1}, \ldots, x_i\}$ or $\{x_i, x_{i+1}, \ldots, x_j\}$ has the same chance to be assigned the highest priority, and since there are $|i-j|+1$ nodes in the set, we get

$$Pr(A_i^j = 1) = \frac{1}{|i-j|+1}.$$

The expected depth of x_i can be computed with the above Lemma

$$\begin{aligned} \textstyle\sum_{j=1}^{n} Pr(A_i^j = 1) &= \textstyle\sum_{j=1}^{i-1} Pr(A_i^j = 1) + \sum_{j=i+1}^{n} Pr(A_i^j = 1) \\ &= \textstyle\sum_{j=1}^{i-1} \frac{1}{i-j+1} + \sum_{j=i+1}^{n} \frac{1}{j-i+1} = \sum_{k=2}^{i} \frac{1}{k} + \sum_{k=2}^{n-i+1} \frac{1}{k} \\ &= H(i) - 1 + H(n-i+1) - 1 < \log i + \log(n-i+1) < 2\log n. \end{aligned} \tag{23}$$

It indicates that $\mathbb{E}[d(x_i)] \in O(\log n)$. □

Lemma 14.2. *Given a random treap T with $n > 3$ elements, the expected number of leaves in T is $1 + (n-2)/3$; the expected number of nodes in T with two children is $(n-2)/3$ and the expected number of nodes in T with exactly one child is $(n+1)/3$.*

Proof. Let L be the number of leaves in T, O be the number of nodes in T with one child, and F be the number of nodes with two children. Then $L + O + F = n$. Considering the edges in T, each node with exactly one child has one outgoing edge, and each node with two children has two outgoing edge and those leaves do not have any outgoing edge. The total number of edges is $n-1$. Therefore, $2F + O = n - 1$. Hence, $F = L - 1$ and $O = n - 2L + 1$.

If $n = 1$, the unique element must be a leaf node, and $\mathbb{E}[L] = 1$. If $n = 2$, the two element has the same chance to be a leaf node, therefore $\mathbb{E}[L] = 2 * (1/2) = 1$. Considering $n \geq 3$.

Let L_i be a random indicator to represent whether node i is a leaf. The number of leaves can be written as $L = \sum_i L_i$. To compute the expectation, it's required to compute the probability $Pr(L_i = 1), \forall 1 \le i \le n$. There are two special cases: when $i = 1/n$, the node is leaf iff it has lower priority than node $2/n - 1$, e.g $Pr(L_1 = 1) = Pr(L_n = 1) = 1/2$. When $1 < i < n$, the node i is leaf iff it has the lowest priority among x_{i-1}, x_i, x_{i+1}, and thus $Pr(L_i = 1) = 1/3$. Hence, $\mathbb{E}[L] = 2 * (1/2) + (n-2)/3 = 1 + (n-2)/3$. Taking advantage the relations between F, O and L, we get $\mathbb{E}[F] = \mathbb{E}[L] - 1 = (n-2)/3$ and $\mathbb{E}[O] = n - 2\mathbb{E}[L] + 1 = n - 2 - 2(n-2)/3 + 1 = (n+1)/3$. □

Problem 14.1. *Given a random treap T with n elements, compute the expected length of the left spine of T.*

Proof. Let x_n be the leaf of the left spine of T, then x_n must be the smallest element in T. The expected length of the left spine of T is the expected depth of x_n, we have

$$\mathbb{E}(\text{length of the left spine of } T) = \mathbb{E}[d(x_n)].$$

To compute the expected depth of x_n in T, we apply the result in Eq.(23)

$$\mathbb{E}[d(x_n)] = \sum_{i=1}^{n-1} Pr(A_n^i = 1) = \sum_{i=1}^{n-1} \frac{1}{n-i+1} = H(n) - 1.$$

□

14.2 Skip List

Definition 14.1 (Leveling)**.** *A leveling with r levels of an ordered set $S = \{x_1 < x_2 < \cdots < x_n\}$ is a sequence of nested subsets (called levels) $L_r \subseteq L_{r-1} \subseteq \cdots \subseteq L_2 \subseteq L_1$, s.t. $L_r =$ and $L_1 = S$.*

Definition 14.2 (Level)**.** *Given an ordered set S and a leveling for it, the level of any element $x \in S$ is defined as*

$$\ell(x) = \max_{1 \le i \le r} \{i | x \in L_i\}.$$

Adding two special elements $\{-\infty, \infty\}$ to a given leveling of S defines a skip list of S. Each element $x \in S$ has $\ell(x)$ nodes in pile, and each level is presented as a sorted linked list.

Definition 14.3 (Random Skip List)**.** *For each $x \in S$, a fair coin is flipped repeatedly until a tail appears. For every head, a copy of x is added to a higher level. Sort the elements in each level and form a linked list for each level. In addition to its key and forward pointer for that level, each element keeps a down pointer to its duplicate on its lower level.*

According to the construction of the random skip list, the level $\ell(x)$ of an element $x \in S$ is a geometric with parameter $p = 1/2$, and

$$Pr(\ell(x) = k) = 2^{-k}, Pr(\ell(x) \geq k) = \sum_{i=k}^{\infty} 2^{-k} = 2^{1-k}.$$

The definition of the number of the leveling r implies that

$$Pr(r \geq k) = Pr(\bigcup_{x \in S} \{\ell(x) \geq k\}) \leq \sum_{x \in S} Pr(\ell(x) \geq k) = n2^{1-k}.$$

Lemma 14.3. *The discrete random variable X is non-negative, then $\mathbb{E}[X] = \sum_{i=1}^{\infty} Pr(X \geq i)$.*

Proof. According to the definition of expectation

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{i=0}^{\infty} iPr(X = i) = \sum_{i=1}^{\infty} i[Pr(X \geq i) - Pr(X \geq i+1)] \\
&= \sum_{i=1}^{\infty} iPr(X \geq i) - \sum_{i=1}^{\infty} iPr(X \geq i+1) \\
&= \sum_{i=1}^{\infty} iPr(X \geq i) - \sum_{i=2}^{\infty} (i-1)Pr(X \geq i) \\
&= \sum_{i=1}^{\infty} Pr(X \geq i).
\end{aligned}$$

□

Problem 14.2. *If each trial of an experiment succeed w/p at least p. Compute the expected number of trials to success.*

Solution 14.1. *Let X be the number of trials to success. Then $\mathbb{E}[X] = \sum_{i=0}^{\infty} iPr(X = i) = \sum_{i=1}^{\infty} Pr(X \geq i)$. All trials are independent, therefore $Pr(X \geq i) = Pr(\text{all trials until } i-1 \text{ failed}) \leq (1-p)^{i-1}$. Therefore, $\mathbb{E}[X] \leq 1/(1-(1-p)) = 1/p$.*

Theorem 14.3. *The number of levels (or height) r in a random leveling of a set S of size n has its expectation*

$$\mathbb{E}[r] = O(\log n).$$

Moreover, $r = O(\log n)$ with high probability.

Proof. The number of levels r is non-negative, discrete random variable, then

$$\begin{aligned}
\mathbb{E}[r] &= \sum_{i=0}^{\infty} iPr(r=i) = \sum_{i=1}^{\infty} Pr(r \geq i) \\
&= \sum_{i=1}^{c\log n} Pr(r \geq i) + \sum_{i=c\log n+1}^{\infty} Pr(r \geq i) \\
&\leq c\log n + \sum_{i=c\log n+1}^{\infty} Pr(r \geq i) \\
&= c\log n + \sum_{i=c\log n+1}^{\infty} n2^{1-i} \\
&= c\log n + 2n/2^{c\log n} \leq c\log n + 2.
\end{aligned}$$

The last inequality holds when $c \geq 1$. The probability of having a skip list with levels exceeding $c\log n$

$$Pr(r \geq c\log n) \leq \frac{n}{2^{c\log n-1}} = \frac{2}{n^{c-1}} \in o(1),$$

when $c > 1$. That says, $r = O(\log n)$ with high probability. □

Algorithm 6 Search Skip List

Input: element x, and a leveling of sorted set S with number of levels r

```
1:  v = −∞
2:  while v ≠ ∞ and key(v) < x do
3:      if key(right(v)) < x then
4:          v = right(v)
5:      else if key(right(v)) > x then
6:          v = down(v)
7:      else
8:          return v
9:      end if
10: end while
```

Lemma 14.4. *The expected memory a random skip list for a set of n elements is $O(n)$.*

Proof. The memory of a random skip list for a set of n distinct elements is the sum of the memory each element uses. Let $\ell(x)$ be the levels of x appears in the skip list, it's a geometric r.v. with parameter $p = 1/2$, and $\mathbb{E}[\ell(x)] = 1/p = 2$. The memory used by a skip list for S is $X = \sum_{x \in S} \ell(x)$, and $\mathbb{E}[X] = \sum_{x \in S} \mathbb{E}[\ell(x)] = 2n \in O(n)$. □

Lemma 14.5. *The expected searching time for any element in a random skip list for n distinct elements is $O(\log n)$.*

Proof. To analysis the searching time, we conduct a backward search over the generated skip list from the bottommost to the leftmost element at the topmost level. The search algorithm is *FlipWalk* (see Table 7).

Let X_r be the number of steps required to walk up r levels. We have the relation

$$\mathbb{E}[X_r] = 1 + \mathbb{E}[X_r|\text{Flip}_r].$$

At each level, it requires at least one step before walking up or left. The expected steps of moving up at level i depend on the flipping outcomes: moving up if the flip is Head, moving left otherwise. Thus, we have

$$\begin{aligned}\mathbb{E}[X_r|\text{Flip}_r] &= \mathbb{E}[X_r|\text{Flip}_r = \text{Head}]Pr(\text{Flip}_r = \text{Head}) + \mathbb{E}[X_r|\text{Flip}_r = \text{Tail}]Pr(\text{Flip}_r = \text{Tail}) \\ &= (\mathbb{E}[X_r|\text{Flip}_r = \text{Head}] + \mathbb{E}[X_r|\text{Flip}_r = \text{Tail}])/2 = (\mathbb{E}[X_{r-1}] + \mathbb{E}[X_r])/2,\end{aligned}$$

and $\mathbb{E}[X_r] = 2 + \mathbb{E}[X_{r-1}]$. It implies that the expected number of steps on each level is 2. The empty leveling is a trivial special case: the expected number of steps to walk up $r = 0$ level is also zero, i.e. $\mathbb{E}[X_0] = 0$. Therefore $\mathbb{E}[X_r] = 2r$ and $\mathbb{E}[\mathbb{E}[X_r]] = O(\log n)$. □

Algorithm 7 FlipWalk Backward Searching

Input: searching key element v, the leftmost element at the topmost level u

```
1: while v ≠ u do
2:     if Flip = Head then
3:         v ← walk up(v)                          ▷ element exists when walk up
4:     else
5:         v ← walk left(v)
6:     end if
7: end while
```

14.3 Hash Table

A hash table is a data structure with a table T consisting of n cells indexed by $N = \{0, 1, \ldots, n-1\}$, and hash function $h : M \to N$, where $M = \{0, 1, \ldots, m-1\}$ is a totally ordered universe. Generally, $m \gg n$. Hash table allows $O(1)$ searching operation, works efficiently.

A *perfect hash function* maps distinct keys in $S \in 2^M$ to distinct locations in T. A *collision* occurs when two distinct keys x and y has the same hash value, i.e. $h(x) = h(y)$. These two elements collide at the corresponding location in T.

There is no perfect hash function s.t it maps keys in all possible $S \in 2^M$ with no collision. It's much common to allow a small number of collisions in real applications. To allow collision, the keys colliding at any given location are usually organized into a *secondary data structure*, e.g. linked list accessible from the location. It's so called *chaining*, another approach is *open addressing*, which does not maintain a secondary data structure but looks for another locations in the hash table.

There is a randomized hashing scheme to conduct search and update operations in expected time $O(1)$ without any probabilistic assumption over the operation sequence, but w.r.t the random choices of the hash function.

Definition 14.4 (Strongly k-universal Hash Family). *Let $M = \{0,1,\ldots,m-1\}$ and $N = \{0,1,\ldots,n-1\}$, with $m \geq n$. A family H of functions from M into N is called strongly k-universal or k-uniform if, for any set $S \in 2^M$ with $|S| = k$ and any set $T \in 2^N$ with $|T| = k$, where $k \leq \min\{m,n\}$, the probability that a random hash function h uniformly drawn from H maps the ith element in S to the ith element of T is n^{-k}, i.e.*

$$Pr(\cap_{i=1}^{k}\{h(x_i) = y_i\}) = 1/n^k,$$

where $\{x_1,x_2,\ldots,x_k\}$ are k distinct elements drawn from M, but $\{y_1,y_2,\ldots,y_k\}$ drawn from N may contain some same elements.

Definition 14.5 (Universal Hashing Family). *Let $M = \{0,1,\ldots,m-1\}$ and $N = \{0,1,\ldots,n-1\}$, with $m \geq n$. A family H of functions from M into N is called 2-universal if, for any two distinct elements $x,y \in M$, and for h chosen uniformly at random from H,*

$$Pr(h(x) = h(y)) \leq 1/n.$$

Lemma 14.6. *The strongly 2-universal hashing family belongs to the 2-universal hashing family.*

Proof. Suppose h is a hash function chosen uniformly at random from the strongly 2-universal hashing family H. Given any two distinct elements $x,y \in M$, we have

$$Pr(h(x) = h(y)) = \sum_{i=0}^{n-1} Pr(h(x) = i, h(y) = i) = \sum_{i=0}^{n-1} n^{-2} = n^{-1}.$$

Therefore, h is also a hash function from the 2-universal hashing family. □

Lemma 14.7. *Let h be a hash function chosen uniformly at random from the 2-universal hashing family H, the expected number of search operations is $O(1 + m/n)$.*

Proof. Given $\forall z \in M$, we conduct the lookup operation on the hash table T, let X_i be an indicator r.v of whether $h(x_i) = h(z)$, $i = 0,1,\ldots,m-1$ and $x_i \in M$. If $h(x_i) = h(z)$, it's said that x_i and z are hashed to the same bucket. Therefore, the size X of bucket of $h(z)$ is the

number of search operations required to lookup z, it can be represented using the sum of the r.v.s X_i, i.e. $X = \sum_{x_i \neq z \in M} X_i$ and its expectation

$$\mathbb{E}[X] = \sum_{x_i \neq z \in M} \mathbb{E}[X_i] = \sum_{x_i \neq z \in M} Pr(h(x_i) = h(z)) \leq \sum_{x_i \neq z \in M} 1/n \leq m/n.$$

Adding the hashing operation, which is $O(1)$, therefore it takes $O(1 + m/n)$ times to lookup an element in T. □

Definition 14.6 (Load Factor)**.** *The ration $\alpha = m/n$ is called load factor of the hash table T.*

The choice of the load factor is a tradeoff between the space and time. A higher load factor implies a higher collision probability and longer time in searching, but a lower memory space. A lower load factor requires to increase the size of the hash table and rehashing of the existing elements in old hash table to the new table, but it reduce the collision chance and thus make sure the constant searching complexity. For example, the initial load factor of the HashMap in Java is $\alpha = 0.75 < 1$.

Definition 14.7 (Nearly Universal Hashing Family)**.** *Let $M = \{0, 1, \ldots, m-1\}$ and $N = \{0, 1, \ldots, n-1\}$, with $m \geq n$. A family H of functions from M into N is called nearly universal if, for any two distinct elements $x, y \in M$, and for h chosen uniformly at random from H,*

$$Pr(h(x) = h(y)) \leq 2/n.$$

The simplest technique for nearly universal hashing is *multiplicative hashing*[1], including two variants: *Prime Multiplicative Hashing*(PMH) and *Binary Multiplicative hashing*(BMH). PMH uses modular arithmetic with prime numbers, and BMH uses modular arithmetic with the powers of 2. Both variants requires a integer r.v named *salt a* to define a hash function. The salt is sampled uniformly at random when create the hash table and kept fixed during the entire lifetime of the table. All consequential analysis on the probability of the two hashing families are defined w.r.t the salt a.

Given any non-negative integer n, we denote $[n] = \{0, 1, \ldots, n-1\}$, and $[n]^+ = \{1, 2, \ldots, n-1\}$.

[1] http://web.engr.illinois.edu/ jeffe/teaching/algorithms/notes/12-hashing.pdf

Definition 14.8 (Prime Multiplicative Hashing)**.** *Given a prime number $p > m = |M|$. For any integer $a \in [p]^+$, a hash function $multp_a : M \to N$ is defined in terms of p*

$$multp_a(x) = (ax \bmod p) \bmod n, \forall x \in M.$$

The set of all such hash functions

$$PMH = \{multp_a | a \in [p]^+\}$$

is called the PMH family.

Property 14.1. *$\forall x, y \in M$, and $a \in [p]^+$, $multp_a(x) - multp_a(y) = multp_a(x - y)$.*

Lemma 14.8. *$\forall x, z \in [p]^+$, there exists a unique integer $a \in [p]^+$, s.t $ax \bmod p = z$.*

Proof. Let $x \in [p]^+$. Suppose there are two distinct integers $a, b \in [p]^+$, s.t $ax \bmod p = bx \bmod p$ or $(a-b)x \bmod p = 0$. Because $x \in [p]^+$, it implies that $a - b$ is divisible by p. Since p is prime and $|a - b| < p$, which means $a = b$. Similarly, $\forall z \in [p]^+$, there exists a unique $a \in [p]^+$, s.t. $ax \bmod p = z$.

Suppose it holds for $z = 0$, or there exists a unique integer $a \in [p]^+$, s.t. $ax \bmod p = 0$. It means ax is divisible by p. Because p is prime, either a or x is divisible by p, which is impossible. □

The function $f_a : [p]^+ \to [p]^+$ defined by $f_a(x) = ax \bmod p$ is injective and bijective.

Lemma 14.9. *PMH is universal. It's sufficient to show $\forall x \in [p]^+$, $Pr(multp_a(x) = 0) \leq 1/n$.*

Proof. To prove PMH is universal, we have to show $\forall x \neq y \in [p]^+$

$$Pr(\text{multp}_a(x) = \text{multp}_a(y)) \leq 1/n.$$

The equality $\text{multp}_a(x) = \text{multp}_a(y)$ indicates $\text{multp}_a(x - y) = 0$. It suffices to show $\forall x \in [p]^+$, $Pr(\text{multp}_a(x) = 0) \leq 1/n$. The salts $a \in [p]^+$ lead to $\text{multp}_a(x) = 0$, also present $ax \bmod p$ is divisible by n. There are $\lfloor (p-1)/n \rfloor$ integers divisible by n. According to the

above lemma, $\forall z \in [p]^+$, there exists a unique salt $a \in [p]^+$ s.t $ax \bmod p = z$. Therefore, there are $\lfloor (p-1)/n \rfloor$ salts with $z = kn$, where $1 \le k \le \lfloor (p-1)/n \rfloor$. Considering there are $p-1$ integers in $[p]^+$,

$$Pr(\text{multp}_a(x) = 0) = \lfloor (p-1)/n \rfloor / (p-1) \le 1/n.$$

Thus, PMH is universal. □

Definition 14.9 (Binary Multiplicative Hashing)**.** *Let $M = [2^w], N = [2^\ell], W = \{x \in M \text{ is odd}\}$. Given $a \in W$, a hash function $multb_a : M \to N$ is defined in terms of the powers of 2*

$$multb_a(x) = \lfloor \frac{ax \bmod 2^w}{2^{w-\ell}} \rfloor, \forall x \in M.$$

The set of such hash functions

$$BMH = \{multb_a | a \in W\}$$

is called the BMH family.

BMH is to hash w-bit words to ℓ-bit fingerprint. Let's have a close look at the formula of the hash function. The product ax is $2w$ bits long, since both $a, x \in M = [2^w]$. The modular arithmetic operation $ax \bmod 2^w$ throws out the higher w bits of ax, and the division by $2^{w-\ell}$ is a bitwise right shift operation, i.e. keeps the top $w-(w-\ell) = \ell$ bits of $ax \bmod 2^w$.

Lemma 14.10. *$\forall x, z \in W$, there exists a unique integer $a \in W$ s.t $ax \bmod 2^w = z$.*

Proof. Given $x \in W$. Assume there are two distinct integers $a, b \in W$, s.t

$$ax \bmod 2^w = bx \bmod 2^w.$$

It implies that $(a-b)x \bmod 2^w = 0$, and $(a-b)x$ is divisible by 2^w. Because x is odd, so $a-b$ is divisible by 2^w. Also $-2^w < a-b < 2^w$, it's clear $a = b$. Similarly, $\forall z \in W$, there exists a unique salt $a \in W$, s.t $ax \bmod 2^w = z$. □

The function $f_a : W \to W$ defined by $f_a(x) = ax \bmod 2^w$ is injective and bijective.

Lemma 14.11. *BMH is nearly universal.*

Proof. To prove BMH is nearly universal, we show that $\forall x, y \in W$, $x \neq y$

$$Pr(\text{multb}_a(x) = \text{multb}_a(y)) \leq 2/n.$$

Suppose $x > y$. $\text{multb}_a(x) = \text{multb}_a(y)$ indicates that the top ℓ bits of $ax \bmod 2^w$ and $ay \bmod 2^w$ are identity. Let t denote these equal bits, and r_x, r_y denote the remaining bits of $ax \bmod 2^w$ and $ay \bmod 2^w$. Because $f_a(x) = ax \bmod 2^w$ is injective, then $r_x \neq r_y$. If $r_x > r_y$, then $a(x-y) \bmod 2^w = r_x - r_y > 0$. Because $r_x - r_y \leq 2^{w-\ell}$, so the top ℓ bits of $a(x-y) \bmod 2^w$ are all zeros. Similarly, if $r_x < r_y$, the top ℓ bits of $a(y-x) \bmod 2^w$ are all zeros, or the top ℓ bits of $a(x-y) \bmod 2^w$ are all ones. That is, $\text{multb}_a(x-y) = 0$ or $\text{multb}_a(x-y) = 2^\ell - 1 = n-1$.

According to the above analysis, we have

$$Pr(\text{multb}_a(x) = \text{multb}_a(y)) \leq Pr(\text{multb}_a(x-y) = 0) + Pr(\text{multb}_a(x-y) = n-1).$$

Since $x \neq y$, $a(x-y)$ can be decompose into the product of an odd and the power of 2, i.e.

$$a(x-y) = z2^k,$$

where z is odd, and $0 \leq k \leq w-1$. According the previous Lemma, $az \bmod 2^w$ consists of $w-1$ random bits, followed by a 1. Because both a and z are odd, az is odd. However, 2^i are all even, except when $i = 0$. And $a(x-y) \bmod 2^w = z2^k \bmod 2^w$ consists of $w-k-1$ random bits, followed by a one, followed by k zeros for left shifting operation.

We note $\text{multb}_a(x-y)$ is the top ℓ bits of $a(x-y) \bmod 2^w$. To figure out the distribution of $\text{multb}_a(x-y)$, let's consider three cases as follows:

1. If $k < w-\ell$, then $w-k-l > \ell-1 \geq \ell$, $\text{multb}_a(x-y)$ consists of ℓ random bits

$$Pr(\text{multb}_a(x-y) = 0) = Pr(\text{multb}_a(x-y) = n-1) = 1/2^\ell = 1/n.$$

2. If $k = w-\ell$, then $w-k-1 = \ell-1$, $\text{multb}_a(x-y)$ consists of $\ell-1$ random bits, followed by a one

$$Pr(\text{multb}_a(x-y)=0)=0, Pr(\text{multb}_a(x-y)=n-1)=1/2^{\ell-1}=2/n.$$

3. If $k > w-\ell$, then $w-k-1 < \ell-1$, $\text{multb}_a(x-y)$ consists of one or more random bits, followed by a one, followed by one or more zeros

$$Pr(\text{multb}_a(x-y)=0)=Pr(\text{multb}_a(x-y)=n-1)=0.$$

Above all, $\forall x \neq y \in [2^w], a \in W$

$$Pr(\text{multb}_a(x)=\text{multb}_a(y)) \leq 2/n.$$

That is, BMH is near-universal. □

Problem 14.3. *Let $h : M \to N$ be a uniformly sampled hash function at random from all possible hashing functions, where $|M| = m$, and $|N| = n$. Assuming $m \leq n$, and show the probability that the randomly picked hashing function is a **perfect hashing function**. Put it in another words, considering there are m balls and n bins, the perfect hashing function assigns each bin at most one balls.*

Solution 14.2. *The size of the hashing functions equals to the total number of the assignments. Each ball has n possible assignments, and the balls are assigned independently, the corresponding number is n^m. The number of perfect hashing functions can be computed from the number of permutations of m balls in n bins, it's $A_n^m = n(n-1)(n-2)\cdots(n-m+1)$. Above all, we uniformly sample a hashing function at random from the collection with probability*

$$p = \frac{A_n^m}{n^m} = \frac{n(n-1)(n-2)\cdots(n-m+1)}{n^m} = \prod_{i=1}^{m-1}\left(1-\frac{i}{n}\right).$$

Problem 14.4. *Let $\mathcal{H}$ be a 2-universal family of hash functions from X to Y, where $|Y| = cM^2$, and suppose we use h, a hash function randomly drawn from $\mathcal{H}$ to hash M unique items. Show that the probability of a collision occurring is less than $1/(2c)$.*

Proof. The hash function $h \in \mathcal{H}$ is 2-universal. Given any two distinct items $x, y \in X$

$$Pr(h(x) = h(y)) \leq 1/|Y| = 1/cM^2.$$

Therefore, the probability of a collision occurring

$$Pr(\bigcup_{x \neq y} \{h(x) = h(y)\}) \leq \sum_{x \neq y} Pr(h(x) = h(y)) = \frac{M(M-1)}{2} \frac{1}{cM^2} \leq \frac{1}{2c}.$$

□

Searching in a chained hash table requires $O(1)$ expected time, but the worst search time may be very high. Assuming that we are using *ideal random hashing functions*. To compute the worst search time, we show the lemma on *maximum load* as follows.

Lemma 14.12. *If n balls are independently and uniformly at random assigned to one of n bins, the fullest bin contains $O(\log n / \log\log n)$ balls with high probability.*

Proof. Let X_i be the number of balls assigned to the ith bin, $\forall i = 1, 2, \ldots, n$, then $X_i \sim B(n, 1/n)$. Therefore, $\mathbb{E}[X_i] = 1$, and the probability that the fullest bin contains at least $k_\alpha = c \log n / \log\log n$ balls with probability

$$Pr(\max_i X_i \geq k_\alpha) = Pr(\bigcup_i \{X_i \geq k\}) \leq \sum_i Pr(X_i \geq k_\alpha) = nPr(X_i \geq k_\alpha).$$

Let $p = 1/n$, $q = 1 - 1/n$, then $Pr(X_i = k) = \binom{n}{k} p^k q^{n-k}$, and

$$\frac{\binom{n}{k+1} p^{k+1} q^{n-k-1}}{\binom{n}{k} p^k q^{n-k}} = \frac{(n-k)p}{(k+1)q} = \frac{n-k}{(n-1)(k+1)} \leq \frac{n-k_\alpha}{(n-1)(k_\alpha+1)}.$$

Denote $\lambda = \frac{n-k_\alpha}{(n-1)(k_\alpha+1)}$, then $\lambda < 1$. Rearranging the tail of the binomial coefficients gives

$$Pr(X_i \geq k_\alpha) = \sum_{k \geq k_\alpha} \binom{n}{k} p^k q^{n-k} = \binom{n}{k_\alpha} p^{k_\alpha} q^{n-k_\alpha} \sum_{i \geq 0} \lambda^i < \binom{n}{k_\alpha} p^{k_\alpha} q^{n-k_\alpha} \frac{1}{1-\lambda}.$$

Assume $k_\alpha \geq 1$, then $\lambda \leq 1/(k_\alpha + 1)$ and $1/(1-\lambda) \leq 1 + 1/k_\alpha \leq 2$.

Applying the inequalities

$$\binom{n}{k_\alpha} \leq \left(\frac{en}{k_\alpha}\right)^{k_\alpha}, (1-\frac{1}{n})^{n-k_\alpha} = (1-\frac{1}{n})^{n}(1-\frac{1}{n})^{-k_\alpha} < 1/e.$$

plugging them into the RHS of the inequality of the probability gives

$$n\binom{n}{k_\alpha} p^{k_\alpha} q^{n-k_\alpha} \frac{1}{1-\lambda} < n\frac{2}{e}\left(\frac{e}{k_\alpha}\right)^{k_\alpha} < n\left(\frac{e}{k_\alpha}\right)^{k_\alpha}.$$

Let $c = 3$, we rearrange the RHS

$$\begin{aligned} n\left(\tfrac{e}{k_\alpha}\right)^{k_\alpha} &\leq n\left(\frac{\log\log n}{\log n}\right)^{3\log n/\log\log n} \\ &= e^{\log n + 3\log n(\log\log\log n - \log\log n)/\log\log n} \\ &= e^{-2\log n} e^{(3\log n)\log\log\log n/\log\log n} < 1/n. \end{aligned}$$

Therefore $Pr(\max_i X_i \leq 3\log n/\log\log n) > 1 - 1/n \to 1$ as $n \to \infty$. □

The idea of open addressing is to trade table size for pointers. All elements are directly stored in the hash table. To perform an insertion, the hash table is probed for an empty slot in some systematic way. Instead of using a fixed order, the sequence of positions probed depends on the key to be inserted.

The hash function is redefined as

$$h : U \times \{0,1,\ldots,m-1\} \to \{0,1,\ldots,m-1\}$$

For $\forall x \in U$, the probe sequence $\langle h(x,0), h(x,1), \ldots, h(x,m-1)\rangle$ should be a permutation over $\langle 0,1,\ldots,m-1\rangle$. If no empty position is found in the sequence the hash table overflows.

The main problem with open addressing is the deletion of elements, which may affect the probe sequence for other elements in the table.

Algorithm 8 Open Hashing Insert

Input: Hash table $T \leftarrow \emptyset$, hash function h, $i \leftarrow 0$

```
1:  function HashInsert(T, x)
2:      while i < m do
3:          j ← h(x, i)
4:          if T[j] = NIL then
5:              T[j] ← x
6:              Return j                          ▷ insert entry x in hash table T
7:          else
8:              i ← i + 1
9:          end if
10:     end while
11: end function
```

Theorem 14.4. *Given an open-address hash table with load factor $\alpha = n/m < 1$.*

- *The expected number of probes in an **unsuccessful search** is at most $\frac{1}{1-\alpha}$, assuming uniform hashing*[2]*.*
- *The expected number of probes in a **successful search** is at most $\frac{1}{\alpha}\ln\frac{1}{1-\alpha}$, assuming uniform hashing and each key in the table is equally likely to be searched for.*

Proof. Let random variable X be the number of probes in an unsuccessful search. Let A_i denote that there is an ith probe to an occupied slot. An **unsuccessful search** means that every probe except the last one is to an occupied slot. Therefore, we have

$$\begin{aligned} Pr(X \geq i) &= Pr(A_1 \cap A_2 \cap \cdots \cap A_{i-1}) \\ &= Pr(A_1)Pr(A_2|A_1)Pr(A_3|A_1 \cap A_2)\cdots Pr(A_{i-1}|A_1 \cap A_2 \cap \cdots \cap A_{i-2}) \\ &= \tfrac{n}{m}\tfrac{n-1}{m-1}\tfrac{n-2}{m-2}\cdots\tfrac{n-i+2}{m-i+2} \leq (\tfrac{n}{m})^{i-1} = \alpha^{i-1}. \end{aligned}$$

[2]**Uniform hashing** assumes that each key is equally likely to have any of the $m!$ permutations over $\langle 0, 1, \ldots, m-1\rangle$ as its probe sequence. It's different from **simple uniform hashing** assumption that each key is equally likely to be hashed into any of the m slots.

The expected number of probes in an unsuccessful search

$$\mathbb{E}[X] = \sum_{i=0}^{m} iPr(X = i) = \sum_{i=1}^{m} Pr(X \geq i) \leq \sum_{i=1}^{m} \alpha^{i-1} \leq \frac{1}{1-\alpha}.$$

If α is constant, searching an open-address hash table requires $O(1)$ time. Inserting an entry into an open-address hash table takes at most $1/(1-\alpha)$ on average.

A successful search follows the same probe sequence as when a key element is inserted in the open-address hash table. Let Z be the number of probes in a successful search, X_i be the number of probes to perform the ith insertion. When inserting the ith element, the load factor is $\alpha_i = (i-1)/m$ and the expected number of probes to insert the ith key

$$\mathbb{E}[X_i] \leq \frac{1}{1-\alpha_i} = \frac{m}{m-i+1}, \forall 1 \leq i \leq n.$$

Therefore, the number of probes in a successful search

$$\begin{aligned}
\mathbb{E}[Z] &= \textstyle\sum_{i=1}^{n} \mathbb{E}[X_i] Pr(i) \leq \alpha^{-1} \sum_{i=1}^{n} \frac{1}{m-i+1} \\
&= \alpha^{-1} \textstyle\sum_{k=m-n+1}^{m} \frac{1}{k} \leq \alpha^{-1} \int_{m-n}^{m} \frac{1}{x} dx \\
&= \alpha^{-1} \ln \frac{m}{m-n} = \frac{1}{\alpha} \ln \frac{1}{1-\alpha}.
\end{aligned}$$

□

14.4 Bloom Filter

A Bloom filter consists of an array of m bits, $A[0]$ to $A[m-1]$, initially all set to 0. A Bloom filter uses k independent random hash functions $h_1, \ldots, h_k$ with range $\{0, 1, \ldots, m-1\}$. Assuming each hash function maps an element in the universe to a random number uniformly over the range $\{0, 1, \ldots, m-1\}$.

Suppose we use a Bloom filter to present a set $S = \{s_1, s_2, \ldots, s_n\}$ of n elements from a large universe U. For any element $s \in S$, the bits $A[h_i(s)]$ are set to 1, $\forall 1 \leq i \leq k$. A bit location can be set to 1 multiple times, but only the first change has an effect. To check whether an element x is in S, we check if all bit locations $A[h_i(x)]$ for $1 \leq i \leq k$ are set to 1. If not, x

must not be in S. If they are all set to 1, we may assert that $x \in S$. However, the assertion is not completely certain, it's possible that the existing hashed values successfully cover these bit locations for x. It's called *false positive error*. Our goal is to choose the best k to minimize the false positive probability.

We analyze the chance that a specific bit location being set to 1 after all elements of S have been hashed into the Bloom filter. Denote the event as ϵ_i, $\forall 0 \leq i \leq m-1$. Because each hash function maps an element uniformly to the range, and there are kn random hash values during the insertion. The probability that none of them set $A[i]$ to 1 is

$$Pr(\bar{\epsilon}_i) = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m}.$$

Given an element x, the events $\epsilon_1, \ldots, \epsilon_k$ related to x are mutually independent. The false positive probability p is the probability that all bit locations $A[h_i(x)]$ are set to 1:

$$p = Pr(\epsilon_1 \cap \ldots \cap \epsilon_k) = \prod_{i=1}^{k} (1 - Pr(\bar{\epsilon}_i) = (1 - Pr(\bar{\epsilon}_i))^k \approx (1 - e^{-kn/m})^k.$$

When $k = (\ln 2)(m/n)$, the derivative of p w.r.t k is zero, and that this point is a global minimum $p = 2^{-k} \approx (0.6185)^{m/n}$. The probability falls exponentially in m/n, the number of bits used per item.

15 Fingerprinting

15.1 Polynomial Identity Testing

Lemma 15.1. *Given two r.v.s X_1 and X_2, show that $Pr(X_1) < Pr(X_1|\bar{X}_2) + Pr(X_2)$.*

Proof. According to the law of total probability, we get

$$Pr(X_1) = Pr(X_1|X_2)Pr(X_2) + Pr(X_1|\bar{X}_2)Pr(\bar{X}_2) < Pr(X_2) + Pr(X_1|\bar{X}_2).$$

□

15.2 Determinant of Matrices

15.3 Perfect Matching

16 Markov Chains

A discrete time Markov chain $\mathcal{M}$ is a sequence of r.v.s $X_1, X_2, \ldots$ with the Markov property (aka memorylessness property), namely future behavior of $\mathcal{M}$ depends only on its present state, and not on its previous states

$$Pr(X_t|X_{t-1},\ldots,X_2,X_1) = Pr(X_t|X_{t-1})$$

if the conditional probability is well-defined. The possible values of X_t form a countable set S is the state space of the MC. If it satisfies $Pr(X_t|X_{t-1}) = Pr(X_{t-1}|X_{t-2}), \forall t \geq 2$, $\mathcal{M}$ is a *time-homogeneous* MC. If the size of S is also finite, $\mathcal{M}$ is a finite time homogeneous MC, which can be represented with a directed graph $G = (V,E)$ and a transition matrix P. The entry $P_{ij} = Pr(X_1 = j|X_0 = i), \forall i,j \in S$. Each vertex of the graph is a state in S, there is an edge directed from i to j if $P_{ij} > 0$.

MCs have many applications:

- HMMs, show up in NLP, ML and signal processing.
- Bayesian ML and statistics to approximately sample from intractable distribution.
- Approximately count (e.g. perfect matchings in a graph) and approximate the volume of complicated (convex) sets.
- Analyze the performance of randomized algorithms
- Differential privacy[3] $\arg\min_{\theta\in\Omega} f(\theta;D)$

[3]Differential privacy is a research topic in the areas of statistics and data analytics that uses hashing, subsampling and noise injection to enable crowdsourced learning while keeping the data of individual users completely private.

16.1 Convergence of MCs

The Markov chains are designed to sample some probability distributions which are hard to sample directly. To investigate the convergence of a MC, we define the stationary distribution, which the MCs are expected to converge.

Definition 16.1 (Stationary Distribution)**.** *The transition matrix P is row stochastic $P\mathbb{1} = \mathbb{1}$. There must be a left eigenvector π of P corresponding to the eigenvalue 1, s.t. $\pi P = \pi$, where $\pi_i \geq 0$ and $\mathbb{1}^T\pi = 1$. If π is unique, it's the stationary distribution of $\mathcal{M}$. Therefore, $\mathcal{M}$ is stationary and it converges to π, i.e $\lim_{t\to\infty} \pi_0 P^t = \pi$ given any initial distribution π_0.*

It implies that if we start the MC with state sampled from π, the chain thereafter consists of states sampled from π. If $\mathcal{M}$ contains multiple stationary distributions, it's reducible. If $\mathcal{M}$ is *irreducible* (a.k.a *ergodic*) iff for every i, j there exists a t, s.t $P^t_{ij} > 0$, the the stationary distribution is unique.

Even when the stationary distribution is unique, you may not have

$$\pi = \lim_{t\to\infty} \pi_0 P^t$$

for every initial distribution π_0.

If $\mathcal{M}$ is *aperiodic*

$$gcd(\{n : P^n_{ii} > 0\}) = 1$$

for all $i \in S$, and irreducible, then the stationary distribution π is unique, and $\mathcal{M}$ converges to it from any initial distribution π_0.

Problem 16.1. *Recall an absorbing state i is a state s.t. $Pr(X_{t+1} = i|X_t = i) = 1$. (1) What does row i of P look like if i is absorbing? (2) Can MC be ergodic if it has an absorbing state?*

Solution 16.1. *(1) If state i is absorbing, then $P_{ii} = 1$ and $P_{ij} = 0, \forall j \neq i$. (2) Suppose there is an absorbing state, say state i. Also, assuming that MC starts at state i, it's trapped and unable to reach other states anymore. Therefore, the MC is not ergodic.*

When the size of the state space is large or the transition matrix is dense, it's hard to compute the stationary probability distribution π. There is an alternative approach to verify that $\mathcal{M}$ is stationary: a reversible MC is stationary.

Definition 16.2 (Reversibility)**.** *A MC $\mathcal{M}$ with transition matrix P is reversible if there's a π s.t $\pi_i P_{ij} = \pi_j P_{ji}, \forall i, j \in S$. The equations are known as detailed balance equations.*

Lemma 16.1. *If $\mathcal{M}$ is a reversible Markov chain with transition matrix P, and $\pi_i P_{ij} = \pi_j P_{ji}, \forall i, j \in S$, then π is the steady state or stationary distribution of $\mathcal{M}$.*

Proof. Summing up the two sides w.r.t i, i.e $\sum_i \pi_i P_{ij} = \sum_i \pi_j P_{ji} = \pi_j$, since P is row stochastic. Therefore, $\pi' P = \pi$, which implies that π's the stationary distribution. □

A typical reversible MC is the random walk on a graph $G = (V, E)$, where a step starting from a vertex u to one of its d_u neighbors, and each of which is chosen w/p $1/d_u$. It has the stationary distribution

$$\pi_u = \frac{d_u}{\sum_u d_u} = \frac{d_u}{2|E|}$$

satisfying

$$\pi_u P_{uv} = \frac{d_u}{2|E|}\frac{1}{d_u} = \pi_v P_{vu} = \frac{d_v}{2|E|}\frac{1}{d_v} = \frac{1}{2|E|}.$$

Problem 16.2. *Given a MC and two states x and y. The hitting time of x starting from y is defined as $\tau_{y\to x} = \min\{t | X_t = x, X_0 = y\}$. Consider a MC on $\mathbb{Z}_n = \{0, 1, \ldots, m-1\}$ which swings back and forth with probability $1/2$, and $f_k = \mathbb{E}[\tau_{k\to 0}]$, show the relationship between f_{k-1}, f_k and f_{k+1}.*

Solution 16.2. *Applying the law of total probability,*

$$\mathbb{E}[\tau_{k\to 0}] = \mathbb{E}[\tau_{k\to 0}|X_1 = k-1]Pr(X_1 = k-1|X_0 = k) + \mathbb{E}[\tau_{k\to 0}|X_1 = k+1]Pr(X_1 = k+1|X_0 = k).$$

Let's have a close look at $\mathbb{E}[\tau_{k\to 0}|X_1 = k-1]$ and $\mathbb{E}[\tau_{k\to 0}|X_1 = k+1]$. The former equals $1 + \mathbb{E}[\tau_{k-1\to 0}]$ and the later one is equal to $1 + \mathbb{E}[\tau_{k+1\to 0}]$. Also, $Pr(X_1 = k-1|X_0 = k) = Pr(X_1 = k+1|X_0 = k) = 1/2$. Therefore, $f_k = 1 + (f_{k-1} + f_{k+1})/2$.

Definition 16.3 (Total Variation Distance)**.** *Let X and Y be the r.v.s defined on the same probability space. The total variation distance between X and Y is defined as*

$$d_{TV}(X,Y) = \sup_A [Pr(X \in A) - Pr(Y \in A)],$$

where the supremum is taken over all subsets A, where $Pr(X \in A)$ and $Pr(Y \in A)$ are defined.

Problem 16.3. *Prove that $d_{TV}(X,Y) = \sup_A |Pr(X \in A) - Pr(Y \in A)| \geq 0$.*

Proof. Assuming that S achieves $\sup_A |Pr(X \in A) - Pr(Y \in A)|$. If $Pr(X \in S) - Pr(Y \in S) \geq 0$, it achieves $d_{TV}(X,Y)$, thus $d_{TV}(X,Y) = \sup_A |Pr(X \in A) - Pr(Y \in A)|$. If $Pr(X \in A) - Pr(Y \in A) < 0$, then

$$\begin{aligned} |Pr(X \in S) - Pr(Y \in S)| &= Pr(Y \in S) - Pr(X \in S) = Pr(X \in \bar{S}) - Pr(Y \in \bar{S}) \\ &= \sup_A |Pr(X \in A) - Pr(Y \in A)| \geq \sup_A [Pr(X \in A) - Pr(Y \in A)]. \end{aligned}$$

The second equation implies that $\bar{S}$ achieves $sup_A[Pr(X \in A) - Pr(Y \in A)]$, therefore

$$Pr(X \in \bar{S}) - Pr(Y \in \bar{S}) = \sup_A [Pr(X \in A) - Pr(Y \in A)] = \sup_A |Pr(X \in A) - Pr(Y \in A)|.$$

□

Total variation distance is a property of distribution, it does not depend on the r.v.s X and Y. Given two probability distributions P and Q, the total variation distance between P and Q is defined in terms of the ℓ_1 distance between two vectors:

$$d_{TV}(P,Q) = \max_A \sum_{i \in A} (p_i - q_i) = \max_A |\sum_{i \in A} (p_i - q_i)|.$$

Lemma 16.2. *Let P and Q be two probability distributions, their total variation distance $d_{TV}(P,Q) = \|P - Q\|_1 / 2$.*

Proof. It's clear that $d_{TV}(P,Q) = d_{TV}(Q,P)$. Let $S = \{i | p_i > q_i\}$. Then

$$\begin{aligned} d_{TV}(P,Q) &= \max_A \textstyle\sum_{i \in A} (p_i - q_i) = \sum_{i \in S} (p_i - q_i), \\ d_{TV}(Q,P) &= \max_A \textstyle\sum_{i \in A} (q_i - p_i) = \sum_{i \in \bar{S}} (q_i - p_i). \end{aligned}$$

Therefore

$$\|P-Q\|_1 = \sum_{\forall i} |p_i - q_i| = \sum_{i\in S}(p_i - q_i) + \sum_{i\in \bar{S}}(q_i - p_i) = 2d_{TV}(P,Q)$$

and $d_{TV}(P,Q) = \|P-Q\|_1 / 2$. □

16.2 Coupling

Lemma 16.3 (Coupling Lemma)**.** *Let P and Q be two distributions on some specific r.v Z. Let $\mathbb{E}_P(Z)$ and $\mathbb{E}_Q(Z)$ be the expectations of Z w.r.t P and Q. Suppose that $|Z| \le M$. Then*

$$|\mathbb{E}_P(Z) - \mathbb{E}_Q(Z)| \le 2Md_{TV}(P,Q).$$

Proof. Computing according to the definition of expectations

$$\begin{aligned} |\mathbb{E}_P(Z) - \mathbb{E}_Q(Z)| &= |\textstyle\sum_z z(P(Z=z) - Q(Z=z))| \le \sum_z |z||P(Z=z) - Q(Z=z)| \\ &\le M\textstyle\sum_z |P(Z=z) - Q(Z=z)| = M\|P-Q\|_1 = 2Md_{TV}(P,Q). \end{aligned}$$

□

To measure the well behavior of $\mathcal{M}$, i.e. the convergence rate of $\mathcal{M}$ to the stationary distribution π, we introduce the the mixing time in terms of the total variation distance and the coupling method.

Definition 16.4 (Mixing Time)**.** *$\forall \epsilon > 0$, $\exists t_{min}(\epsilon)$ s.t $d_{TV}(\pi_0 P^t, \pi) \le \epsilon$, $\forall \pi_0$ and $t \ge t_{min}(\epsilon)$. The time $t_{min}(\epsilon)$ is the **mixing time**, defined as*

$$t_{min}(\epsilon) = \min_t \{d_{TV}(\pi_0 P^s, \pi) \le \epsilon, \forall t \ge s\}.$$

Definition 16.5 (Coupling)**.** *Given two distributions P and Q on the same state space Ω, a **coupling** Z of P and Q is a joint distribution on $\Omega \times \Omega$ s.t the first marginal distribution of Z is P and the second marginal distribution of Z is Q.*

Lemma 16.4 (Coupling Lemma)**.** *For any discrete r.v.s X and Y, $d_{TV}(X,Y) \le Pr(X \ne Y)$.*

Proof. Let A be any set that both X and Y has their definitions on A.

$$Pr(X \in A) = Pr(X \in A \cap Y \in A) + Pr(X \in A \cap Y \in \bar{A}),$$
$$Pr(Y \in A) = Pr(Y \in A \cap X \in A) + Pr(Y \in A \cap X \in \bar{A}).$$

It implies that $Pr(X \in A) - Pr(Y \in A) = Pr(X \in A \cap Y \in \bar{A}) - Pr(Y \in A \cap X \in \bar{A})$. Therefore,

$$|Pr(X \in A) - Pr(Y \in A)| \leq Pr(X \in A \cap Y \in \bar{A}) + Pr(Y \in A \cap X \in \bar{A}) = Pr(X \neq Y), \forall A.$$

Therefore, $d_{TV}(X,Y) = sup_A |Pr(X \in A) - Pr(Y \in A)| \leq Pr(X \neq Y)$. □

To prove the convergence for an ergodic MC with transition matrix P and stationary distribution π starting in an initial distribution π_0, we will construct a series of couplings Z_t between the distribution $\pi_0 P^t$ and π, s.t if $(X_t, Y_t) \sim Z_t$ then

1. $\{X_t\}$ is a MC with transition matrix P and an initial distribution π_0
2. $\{Y_t\}$ is a MC with transition matrix P and an initial distribution π
3. If $X_\tau = Y_\tau, \forall \tau > 0$, the two chains $\{X_t\}$ and $\{Y_t\}$ coalesce, i.e $X_t = Y_t, \forall t \geq \tau$.

According to the coupling lemma, $d_{TV}(\pi_0 P^t, \pi P^t) = d_{TV}(\pi_0 P^t, \pi) \leq Z_t(X_t \neq Y_t)$. Let's investigate the probability $Z_t(X_t \neq Y_t), \forall t \geq \tau$.

Lemma 16.5. *The probability $Z_t(X_t \neq Y_t)$ is non-increasing.*

There are two famous application of the coupling lemma: *Shuffling Cards* and *Random Walks on the Hypercube*. Also, both problems are closely related to the *Coupon Collector Problem*.

Problem 16.4 (Shuffling Cards)**.** *Card-shuffling problem refers to compute the number of times to reshuffle a deck of cards such that it becomes sufficiently randomized. The problem can be described as a Markov Chain with finite states. Given two copies X_t and Y_t of the card-shuffling MC with different states. To construct a coupling, we choose a position i uniformly*

at random from $[n]$, and then obtain X_{t+1} from X_t by moving the i-th card C of X_t to the top. To obtain Y_{t+1} from Y_t, a card of Y_t with the same value as that of C is moved to the top. The coupling is a valid coupling, because in both chains the probability a specific card is moved to the top at each step is $1/n$. The stationary distributions the chains are all uniform.

We note that once a card is touched, it's always in the same position in both chains in the consequential steps. Hence, the two chains are coupled when every card is touched at least once. We can analyze the minimum number of steps until the two chains are coupled. Let A_i be the event that card i has not been touched once after r steps. The probability is easy to computed

$$Pr(A_i) = \left(1 - \frac{1}{n}\right)^r \leq e^{-r/n}.$$

The probability that A: any one of the n cards has not been touched can be bounded above using the union bound inequality and

$$Pr(A) = Pr(A_1 \cup A_2 \cup \cdots \cup A_n) \leq ne^{-r/n}.$$

Let $Pr(A) \leq \epsilon$, we get $r \geq n\ln(n/\epsilon)$, i.e. after $n\ln(n/\epsilon)$ steps, the probability that the chains are not bounded is at most ϵ. According to the coupling lemma, the variation distance between the uniform distribution and the distribution of the chain after $n\ln(n/\epsilon)$ steps is bounded above by ϵ. Hence, the mixing time $t_{\min}(\epsilon) \leq n\ln(n/\epsilon)$.

16.3 Monte Carlo Markov Chain

Monte Carlo Markov Chain (MCMC) provides a general approach to sampling from a desired probability distribution. The basic idea is to construct an *ergodic MC* whose set of states is the sample space and whose stationary distribution is the required sampling distribution. Let $X_0, X_1, \ldots, X_n$ be a run of the chain. The MC converges to the stationary distribution from any starting state X_0 so, after sufficient steps r, the distribution of the state X_r will be close to the stationary distribution, so it can be used as a sample. Similarly, repeating this argument with X_r as the starting state, we use X_{2r} as a sample, and so on.

Therefore, we use the sequence of states $X_r, X_{2r}, X_{3r}, \ldots$ as independent samples from the stationary distribution of the MC.

The efficiency of the approach depends on (a) how large r must be to ensure a good sample; (b) how much computation is required for each step of the MC. **Coupling** is used to determine the relation between the value of r and the quality of the sampling.

The required sampling distribution may be uniform distribution, we can modify a random walk over the state space by giving each vertex an appropriate self-loop probability, then the stationary distribution will be uniform. Even the sampling distribution is not uniform, we can generalize the idea to form a non-uniform stationary distribution.

Lemma 16.6. *For a finite state space Ω, and for each $x \in \Omega$, there is a neighborhood structure $N(x) = \{y \neq x : y \text{ is reachable from } x\}$. Let $N = \max_{x\in\Omega} |N(x)|$, and let M be any number s.t. $M \geq N$. Consider a MC $\mathcal{M}$ where*

$$p_{x,y} = \begin{cases} \frac{1}{M}, & x \neq y, y \in N(x), \\ 0, & x \neq y, y \notin N(x), \\ 1 - \frac{|N(x)|}{M}, & x = y. \end{cases}$$

If the chain is irreducible and aperiodic, then the stationary distribution is uniform.

Proof. For any $x \neq y$, if $\pi_x = \pi_y$, then $\pi_x p_{x,y} = \pi_y p_{y,x}$ since $p_{x,y} = p_{y,x}$. Therefore, the uniform distribution $\pi_x = 1/|\Omega|$ is the stationary distribution. □

The **Metropolis Algorithm** is an approach to generalize the construction that transforms any irreducible MC on Ω *with a known stationary distribution* π to a reversible MC on the same state space with a different stationary distribution μ, where $\mu_x = f(x)/\sum_{y\in\Omega} f(y)$ is proportional to some function $f \geq 0$ on Ω that we can easily compute.

Lemma 16.7. *For a finite state space Ω, and for each $x \in \Omega$, there is a neighborhood structure $N(x) = \{y \neq x : y \text{ is reachable from } x\}$. Let $N = \max_{x\in\Omega} |N(x)|$, and let M be any number s.t. $M \geq N$. For all $x \in \Omega$, let $\pi_x > 0$ be the desired probability of state x in the stationary*

distribution. Consider a MC $\mathcal{M}$ where

$$p_{x,y} = \begin{cases} \frac{1}{M}\min\{1, \frac{\pi_y}{\pi_x}\}, & x \neq y, y \in N(x), \\ 0, & x \neq y, y \notin N(x), \\ 1 - \sum_{y \in N(x)} p_{x,y}, & x = y. \end{cases}$$

If the chain is irreducible and aperiodic, then the stationary distribution is given by π.

Proof. For any $x \neq y$, if $\pi_x \geq \pi_y$, then $p_{x,y} = 1/M$ and $p_{y,x} = (\pi_y/\pi_x)/M$, and $\pi_x p_{x,y} = \pi_y p_{y,x}$. According to the symmetry, we have $\pi_x p_{x,y} = \pi_y p_{y,x}$ as $\pi_x < \pi_y$. Therefore, the stationary distribution is given by π. □

Theorem 16.1. *Let $\mathcal{M}_p$ be a reversible MC on Ω with stationary distribution π. Let $f \geq 0$ be a function defined on Ω. A MC $\mathcal{M}_q$ with transition probability*

$$q_{x,y} = \begin{cases} p_{x,y}\min\{1, \frac{\pi_x f(y)}{\pi_y f(x)}\}, & x \neq y, \\ 1 - p_{x,y}\min\{1, \frac{\pi_x f(y)}{\pi_y f(x)}\}, & x = y, \end{cases}$$

is irreducible and its stationary distribution is μ, in which $\mu_x = f(x)/\sum_{y\in\Omega} f(y), \forall x \in \Omega$.

Proof. The MC $\mathcal{M}_p$ with transition probability P and stationary distribution π is reversible. For any $x \neq y \in \Omega$, we have $\pi_x p_{x,y} = \pi_y p_{y,x}$ and also $\mu_y/\mu_x = f(y)/f(x)$. Now we show the MC $\mathcal{M}_q$ is reversible with stationary distribution μ. It's required to verify that $\mu_x q_{x,y} = \mu_y q_{x,y}$, $\forall x \neq y \in \Omega$.

Assuming $\pi_x f(y)/(\pi_y f(x)) = \pi_x \mu_y/(\pi_y \mu_x) \geq 1$, we get

$$q_{x,y} = p_{x,y}, \quad q_{y,x} = p_{y,x}\frac{\pi_y \mu_x}{\pi_x \mu_y}.$$

Applying the known fact about the reversibility of $\mathcal{M}_p$, we have

$$\mu_x q_{x,y} = \mu_x p_{x,y} = \mu_x \pi_y p_{y,x}/\pi_x = \pi_x \mu_y q_{y,x}/\pi_x = \mu_y q_{y,x}.$$

Therefore, $\mathcal{M}_q$ is irreducible and has its stationary distribution μ. □

Here is a good example to explain how the algorithm works. Let $S = \sum_{i=1}^{\infty} i^{-2} = \pi^2/6$. We can design a irreducible MC on the positive integers with the Metropolis approach, s.t in the stationary distribution $\pi_i = 1/(Si^2)$. The neighborhood structures should be $N(i) = \{i-1, i+1\}, \forall i > 1$ and $N(1) = \{2\}$.

16.3.1 Gibbs Sampling

Important algorithmic application of reversibility is Gibbs sampling.

16.4 Differential Privacy

Consider case that we want to compute

$$\arg\min_{\theta \in \Omega} f(\theta; D).$$

One way to achieve DP is to sample from a probability distribution proportional to the value of f, peaked around

$$P^{\gamma}(\theta \in A) \propto \int_A e^{-\gamma f(\theta; D)} d\theta$$

where $A \subset S$. It's clear that γ trades off between DP and accuracy.

Eigen Decomposition & Gibbs Sampling: given a symmetric matrix $A \in \mathbb{R}^{n \times n}$, DP compute a k-basis for the column span of A

$$A \approx U \Sigma V^T,$$

where $U^T U = I$ and $U = \arg\min_{V^T V = I} -tr(V^T A V)$ to use the exponential mechanism we need to sample from the PDF $Pr(V) = F_1(0.5k, 0.5n, A)^{-1} e^{\gamma tr(V^T A V)}$.

16.5 Streaming Algorithms

Streaming algorithms processing items in terms of a massively long consequence of input stream $x = (x_1, x_2, \ldots, x_m)$, where $x_i \in [n]$, and approximately estimates an associated

function $\phi(x)$ over the stream x. The streaming algorithms expect to finish the estimation using small amount of space s, e.g. $s = o(min(m,n))$. The most common estimations include the frequency of the items in x, and the top k most frequent items.

16.5.1 Finding Frequent Items

Each stream of length m implicitly contains a frequency vector $f(x) = (f_1, \ldots, f_n)$, where $f_i \geq 0, \forall i \in [n], \sum_i f_i = m$. A stream algorithm $\mathcal{A}$ gives the output $\mathcal{A}(x)$ of the input stream x. When it comes to the *frequency problem* with parameter k

$$\mathcal{A}(x) = \{j : f_j > k/m\}.$$

To resolve the frequency problem, it requires to estimate the frequencies of items. The classical approach to estimate the frequency in one-pass is the *Misra-Gries Algorithm*. It maintains an associated array A, whose keys are the items seen in the stream, and whose values are the counters associated with the keys.

Algorithm 9 Misra-Gries Algorithm

Input: $A = \varnothing$

for $i = 1, 2, \ldots, m$ **do**
  Process item $j = x_i$
  if $j \in keys(A)$ **then**
    $A[j] = A[j] + 1$
  else if $|keys(A)| < k - 1$ **then**
    $A[j] = 1$
  else
    for each $\ell \in keys(A)$ **do**
      $A[\ell] = A[\ell] - 1$ ▷ All counters in A are reduced by 1
      if $A[\ell] = 0$ **then**
        Remove ℓ from A
      end if
    end for
  end if
end for

Each key requires at most $\lceil \log n \rceil$ and each value requires at most $\lceil \log m \rceil$ bits.The algorithm maintains at most $k-1$ keys in A at any time, and A is stored with a balanced binary search tree. Therefore, with at most $k-1$ pairs of keys and values, the total space required is at most $O(k(\log m + \log n)) = O(k \log(mn))$.

Theorem 16.2. *The Misra-Gries algorithm with parameter k uses one pass and $O(k \log(mn))$ bits of space, and provides, for any token j, an estimate $\hat{f}_j$ satisfying $f_j - m/k \leq \hat{f}_j \leq f_j$.*

16.5.2 Estimating the Number of Distinct Items

Let $d = |\{j : f_j > 0\}|$ be the number of distinct items in the input stream x. The *distinct elements problem* is to output an (ϵ, δ)-approximation to d. The AMS algorithm was designed

for this problem.

For an integer $x > 0$, let $zeros(x)$ denote the number of zeros that the binary expression of x ends with. Formally, $zeros(x) = \max\{i : 2^i \text{ divides } x\}$.

Algorithm 10 AMS Algorithm

Input: a random hash function $h : [n] \to [n]$ from a 2-universal family, $z \leftarrow 0$

```
for each x_j ∈ x do
    Process(x_j)
end for
function Process(j)
    if zeros(h(j)) > z then
        z ← zeros(h(j))
    end if
end function
```

Output: $\hat{d} \leftarrow 2^{z+1/2}$

The algorithm expects that one of the d distinct items in x to hit $zeros(h(j)) \geq \log d$, s.t $\hat{d} = 2^{z+1/2} > d$, and the maximum value of $zeros(h(j))$ over x – maintained in z of the algorithm – should be a good approximation to $\log d$. Let's analyze the quality of the estimation using the AMS algorithm.

For each $j \in [n]$ and each integer $r \geq 0$, let $X_{r,j}$ be an indicator r.v for $zeros(h(j)) \geq r$. Let $Y_r = \sum_{j:f_j>0} X_{r,j}$, and t denote the value of z when AMS terminates. It's true that

$$Y_r > 0 \quad \text{iff} \quad t \geq r,$$
$$Y_r = 0 \quad \text{iff} \quad t \leq r-1.$$

Because $h(j) \in [n]$ is uniformly distributed over a random bit string of length $\log n$. We have $Pr(h(j) \geq r) = 2^{\log n - r} / 2^{\log n} = 2^{-r}, \forall j \in [n]$. Since h is 2-universal, $X_{r,j}$ are pairwise

independent. Let's analyze the expectation and variance of Y_r:

$$\begin{aligned}\mathbb{E}[Y_r] &= \sum_{j:f_j>0}\mathbb{E}[X_{r,j}] = dPr(h(j)\geq r) = d2^{-r},\\ \mathrm{Var}[Y_r] &= \sum_{j:f_j>0}\mathrm{Var}[X_{r,j}] \leq d\mathbb{E}[X_{r,j}^2] = d\mathbb{E}[X_{r,j}] = d2^{-r}.\end{aligned}$$

Applying Markov's and Chebyshev's inequalities respectively, we get

$$\begin{aligned}Pr(Y_r>0) &= Pr(Y_r\geq 1)\leq \mathbb{E}[Y_r] = d2^{-r},\\ Pr(Y_r=0) &\leq Pr(|Y_r-\mathbb{E}[Y_r]|\geq \mathbb{E}[Y_r]) \leq \tfrac{\mathrm{Var}(Y_r)}{(\mathbb{E}[Y_r])^2} = 2^r/d.\end{aligned}$$

Let $a=\min\{k: 2^{k+1/2}\geq 3d\}$, $b=\max\{k: 2^{k+1/2}\leq d/3\}$, then we have

$$\begin{aligned}Pr(\hat{d}\geq 3d) &= Pr(2^{t+1/2}\geq 2^{a+1/2}) = Pr(t\geq a) = Pr(Y_a>0) = d2^{-a}\leq \sqrt{2}/3,\\ Pr(\hat{d}\leq d/3) &= Pr(2^{t+1/2}\leq 2^{b+1/2}) = Pr(t\leq b) = Pr(Y_{b+1}=0) = 2^{b+1}/d\leq \sqrt{2}/3.\end{aligned}$$

The estimation is not a good approximation, because $Pr(d/3<\hat{d}<3d)\geq 1-2\sqrt{2}/3\approx 6\%$. To improve the estimation, a standard *median trick* is used.

16.6 Overdetermined Linear System

To solve an overdetermined consistent linear system $Ax=b$, where $A\in\mathbb{R}^{m\times n}$ and $rank(A)=n$, both m and n are very large and $m\gg n$, researchers proposed a randomized version of the *Kaczmarz algorithm*. The consistency indicates that it has a *solution* and the solution is *unique*.

The algorithm converges faster than the Conjugate algorithms, and does not require to access to all rows of the matrix A at each iteration.

Before move to the randomized Kaczmarz algorithm, we have a brief view of Kaczmarz algorithm. At each step, a row vector of A is selected, and the solution then moves along the selected direction.

Algorithm 11 Kaczmarz Algorithm

Input: initial guess x^0 for $Ax = b$, termination threshold ϵ, epoch $k = 0$

1: **while** $\|x^{k+1} - x^k\|_2 > \epsilon$ **do**

2: Compute index $i = k \bmod (m+1)$

3: Update $x^{k+1} = x^k + \frac{b_i - a_i^T x_k}{\|a_i\|_2^2} a_i$ ▷ a_i: the ith row of A

4: Increase k by 1

5: **end while**

The solution is updated iteratively, such that $\|x_k - x^*\|_2 < \|x_{k-1} - x^*\|_2$ and $\mathbb{E}\|x_k - x^*\|_2^2 < \mathbb{E}\|x_{k-1} - x^*\|_2^2$. The convergence rate of the problem is determined by the condition number of A, i.e. $\kappa(A) = \sigma_1(A)/\sigma_n(A)$, where $\sigma_1(A), \sigma_n(A)$ is the maximum and the minimum singular value of A, respectively. Since $\sigma_1(A) \geq \sigma_n(A)$, therefore, $\kappa(A) \geq 1$.

Theorem 16.3. *Let x^* be the true solution. After T iterations of the Kaczmarz algorithm*

$$\mathbb{E}\|x_T - x^*\|_2^2 \leq (1 - \frac{\sigma_n^2(A)}{m\|A\|_{2\to\infty}^2})^T \|x_0 - x^*\|_2^2,$$

if uniformly random sample the rows according to the probability

$$p_i = \frac{\|a_i\|_2^2}{\sum_i \|a_i\|_2^2} = \frac{\|a_i\|_2^2}{\|A\|_F^2}.$$

Proof. The idea is to show the iteration decreases the approximation error by a factor of at least $1 - \frac{\sigma_n^2(A)}{m\|A\|_{2\to\infty}^2}$ in expectation, i.e

$$\mathbb{E}(\|x_{k+1} - x^*\|_2^2 | x_k) \leq (1 - \frac{\sigma_n^2(A)}{m\|A\|_{2\to\infty}^2}) \mathbb{E}\|x_k - x^*\|_2^2,$$

□