FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Randomized Algorithms Exam Notes

Emil Petersen Victor Nordam Suadicani

2019/06/21

Contents

1 Introduction

Curriculum: RA 1.1 - 1.3.

1.1 Randomized Quicksort

Random quicksort works by choosing pivots randomly. We want to get an expected running time of this algorithm.

Let S be the array in sorted order. Let S_i be the ith element of S, i.e. the ith smallest element. We count the number of comparisons the algorithm performs in order to give its bound. Let X_{ij} be an indicator variable that is 1 if S_i and S_j are compared in the algorithm. Number of comparisons are then:

$$\sum_{i=1}^{n}\sum_{j>i}^{n} X_{ij}$$

We want the expected value of comparisons:

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{n}\sum_{j>i}^{n} X_{ij}\right] &= \sum_{i=1}^{n}\sum_{j>i}^{n}\mathbb{E}[X_{ij}] \\ &= \sum_{i=1}^{n}\sum_{j>i}^{n}\frac{2}{j-i+1}\end{aligned}$$

Last line follows because they are only compared if they are chosen among $S_i, \ldots, S_j$ sequence.

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j>i}^{n}\frac{2}{j-i+1} &\leq \sum_{i=1}^{n}\sum_{k=1}^{n-i+1}\frac{2}{k} \\ &\leq 2\sum_{i=1}^{n}\sum_{k=1}^{n}\frac{1}{k} \\ &= 2nH(n) \\ &= O(n\ln n)\end{aligned}$$

Thus expected run-time (number of comparisons) is optimal.

1.2 Min-Cut

Randomly choose an edge to contract. Firstly, min-cut is never reduced by a contraction. Secondly, we may not get any optimal min-cut, since if an edge from a min-cut is chosen, it will be eliminated.

Probability of not choosing any edge in a particular min-cut C of size k at step i is $\Pr[\mathcal{E}_i]$. There are at least $kn/2$ edges because otherwise there is some vertex with less than k edges, which is then a min-cut. At step i there are at least $k(n-i+1)/2$ edges left. Chance of not picking one of the k edges of C is then:

$$\mathcal{E}_1 = 1 - \frac{k}{nk/2} = 1 - \frac{2}{n}$$

Generally for the next iterations:

$$\mathcal{E}_i = 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}$$

Chance of *not* picking one of these edges at any step is then:

$$\Pr\left[\bigcap_{i=1}^{n-2} \mathcal{E}_i\right] \geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = \frac{2}{n(n-1)} \geq \frac{2}{n^2}$$

We repeat the algorithm $n^2/2$ times. The chance of not finding any min-cut in any of the runs is then:

$$\left(1 - \frac{2}{n^2}\right)^{n^2/2} < \frac{1}{e}$$

Further runs decrease this chance even more at the expense of more running time.

2 Game-Theoretic Techniques

Curriculum: RA 2 + proof of Yao's minimax principle from Wikipedia.

2.1 Game Tree Evaluation

A game tree is a tree representing a two-player turn-based game between a *minimizer* and a *maximizer*. Every even level of the tree is a choice for the minimizer – every odd level a choice for the maximizer. Given a game tree, you can in theory compute an optimal strategy for each player.

We only concern ourselves with the boolean game tree evaluation problem, $T_{d,k}$, where the tree is an d-ary tree with depth $2k$ and leaves have values in $\{0, 1\}$. Then $\min \equiv \wedge$ and $\max \equiv \vee$.

The worst-case lower bound for any deterministic algorithm is d^{2k}.

Proof. Take any deterministic algorithm A. Simply construct the instance $T_{d,k}$ such that the last option that A considers determines the outcome of the choice, i.e. that the minimizer sees 1's and the last one is a 0 and vice versa for the maximizer. Then A has to look at all the d^{2k} leaves. □

The randomized algorithm simply checks the leaves in a random order. This means that an adversarial instance cannot be constructed. The expected number of leaves read by this randomized algorithm for binary tree instances $(T_{2,k})$ is at most 3^k, which is better than the deterministic bound of $2^{2k} = 4^k$.

Proof. Consider $\vee$ node v and let Y_v be a random variable that counts the number of v's children that are evaluated. Then, consider v's children:

$$\begin{aligned}
\mathbb{E}[Y_v \mid 00] &= 2 \\
\mathbb{E}[Y_v \mid 11] &= 1 \\
\mathbb{E}[Y_v \mid 01] &= \mathbb{E}[Y_v \mid 10] = \frac{3}{2}
\end{aligned}$$

Now consider what v evaluates to:

$$\begin{aligned}
\mathbb{E}[Y_v \mid 0] &\leq 2 \\
\mathbb{E}[Y_v \mid 1] &\leq \frac{3}{2}
\end{aligned}$$

Then consider a $\wedge$ node v and let X_v count the number of v's *grand-children* that are evaluated. Let Y_1 and Y_2 be v's children. Then,

$$\begin{aligned}
\mathbb{E}[X_v \mid 00] &= \mathbb{E}[Y_1 \mid 0] = 2 \\
\mathbb{E}[X_v \mid 11] &= \mathbb{E}[Y_1 \mid 1] + \mathbb{E}[Y_2 \mid 1] \leq \frac{3}{2} + \frac{3}{2} = 3 \\
\mathbb{E}[X_v \mid 01] &= \mathbb{E}[X_v \mid 10] \\
&= \frac{1}{2}\mathbb{E}[Y_1 \mid 0] + \frac{1}{2}(\mathbb{E}[Y_1 \mid 1] + \mathbb{E}[Y_2 \mid 0]) \\
&\leq \frac{1}{2} \cdot 2 + \frac{1}{2}\left(\frac{3}{2} + 2\right) = \frac{11}{4} \leq 3
\end{aligned}$$

Therefore,

$$\mathbb{E}[X_v] \leq 3$$

Now let Z_v be the number of leaf reads done by the algorithm and let $\mathcal{E}_v$ be the event that v is evaluated. Then $\mathbb{E}[Z_v \mid \mathcal{E}_v] \leq 3^k$ with v being the root, by induction on k. First case of $k = 0$ is trivial:

$$\mathbb{E}[Z_v \mid \mathcal{E}_v] = 0 \leq 1 = 3^0 = 3^k$$

Next case is $k > 0$. Let C be the grandchildren of v. Then by hypothesis, $\mathbb{E}[Z_c \mid \mathcal{E}_c] \leq 3^{k-1}$ for $c \in C$.

$$\begin{aligned}
\mathbb{E}[Z_v \mid \mathcal{E}_v] &= \sum_{c \in C} \mathbb{E}[Z_c \mid \mathcal{E}_c] \cdot \Pr[\mathcal{E}_c \mid \mathcal{E}_v] \\
&\leq \sum_{c \in C} 3^{k-1} \cdot \Pr[\mathcal{E}_c \mid \mathcal{E}_v] \\
&= 3^{k-1} \cdot \mathbb{E}[X_v \mid \mathcal{E}_v] \leq 3^{k-1} \cdot 3 = 3^k
\end{aligned}$$

□

2.2 Game Theory: Payoff Matrix

In game-theory, two-player games, strategies and their payoffs can be represented by a matrix. Each row/column represents a strategy, and the entry for that coordinate is what the column player should pay the row player for that combination of strategies. Example: Rock, paper, scissors.

$$\begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}$$

First r/c is rock, second r/c paper, third r/c scissors. This game is a zero-sum game, as the total amount of money is static. In such a game, the row player tries to maximize the outcome, while the column player tries to minimize.

$$V_R = \max_i \min_j M_{ij} \qquad V_C = \min_j \max_i M_{ij}$$

We have $V_R \leq V_C$ as $V_R = \min_j M_{kj} \leq M_{k\ell} \leq \max_i M_{i\ell} = V_C$ for some optimal strategy k for R and optimal strategy ℓ for C. Usually, there is no clear optimal strategy (i.e. the best strategy depends on what the other player does), so we use randomization in what is called *mixed strategies*. This way, each player assigns a distribution on his strategies, and we look at the expected payoff from this distribution. Thus the task becomes to pick a distribution over the strategies such that the expected payoff is optimal. Let $\mathbf{p}$ be the distribution for R and $\mathbf{q}$ the distribution for C. Then

$$\mathbb{E}[\text{ payoff }] = \mathbf{p}^\top \mathbf{M}\mathbf{q} = \sum_{i=1}^{n} \sum_{j=1}^{m} p_i M i j q_j.$$

It turns out that there is an optimal mized strategy that works against any opponent. This is due to Von Neumann's Minimax theorem, which I will not discuss in this presentation.

Rather, I want to show how all this game-theory is relevant to randomized algorithms, by discussing and proving Yao's Minimax principle. We can think of algorithm design as a game, where the designer is the column player, and the row player is an adversary. The entries are the runtime on the input that the row player gives to the designed algorithm. The principle shows that for

any input distribution $\mathbf{p}$ that the adversary chooses, the expected cost of the optimal deterministic algorithm on I_{vep} is a lower bound on the cost of the optimal Las Vegas randomized algorithm $A_{\mathbf{q}}$. Thus we can use this theorem to prove lower bounds on the optimal randomized algorithm for a problem, as we can pick any input distribution and use its expected cost (using the optimal deterministic algorithm) as a lower bound.

2.3 Yao's Minimax Principle

For all distributions $\mathbf{q}$ over $\mathcal{A}$ and $\mathbf{p}$ over $\mathcal{I}$,

$$\min_{A \in \mathcal{A}} \mathbb{E}_{\mathbf{p}}[C(I_{\mathbf{p}}, A)] \leq \max_{I \in \mathcal{I}} \mathbb{E}_{\mathbf{q}}[C(I, A_{\mathbf{q}})]$$

Proof. Let $\min_{A \in \mathcal{A}} \mathbb{E}[C(I_{\mathbf{p}}, A)] = D$ and $\max_{I \in \mathcal{I}} \mathbb{E}[C(I, A_{\mathbf{q}})] = E$.

$$D = \sum_A q_A D \leq \sum_A q_A \mathbb{E}[C(I_{\mathbf{p}}, A)] = \sum_A \sum_I q_A p_I C(I, A) = \sum_I p_I \mathbb{E}[C(I, A_{\mathbf{q}})] \leq \sum_I p_I E = E$$

□

3 Moments and Deviations

Curriculum: RA 3.

3.1 Occupancy Problems

Is this really interesting enough? Hm

Occupancy problems are problems revolving around an analogy of m balls and n bins, where the balls are randomly distributed between the bins. Each bin has a random variable associated with it, X_i which is the number of balls in that bin. Possible problems include:

- What is the expected maximum number of balls in any bin?
- What is the expected number of bins with k balls?

For example, say we want to find k such that with very high probability, no bin contains more than k balls. Then,

$$\forall i\colon 1 - \frac{1}{n} \leq \Pr\left(X_i < k^* = \min\left(n+1, \left\lceil \frac{2e}{e-1}\frac{\ln n}{\ln\ln n}\right\rceil\right)\right)$$

See lecture 3 slides, page 18. In other words, the probability of any bin having less than k^* balls is high, or precisely $1 - 1/n$.

3.2 Markov's Inequality

Let X be a random variable taking only non-negative values. Then:

$$\forall t > 0\colon \Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$$

Proof. Consider $Z = [X \geq t]$. Then $Z \leq \frac{X}{t}$, so:

$$\Pr[X \geq t] = \mathbb{E}[Z] \leq \mathbb{E}\left[\frac{X}{t}\right] = \frac{\mathbb{E}[X]}{t}$$

□

3.3 Chebyshev's Inequality

Let X be a random variable with $\mathbb{E}[X] = \mu$ and $\mathrm{Var}[X] = \sigma^2$. Then:

$$\forall t > 0\colon \Pr[|X - \mu| \geq t\sigma] \leq \frac{1}{t^2}$$

Proof. Let $k = t^2$ and $Y = (X-\mu)^2$. Then $\sigma^2 = \mathbb{E}[Y]$ by definition of the variance. Then:

$$\begin{aligned}\Pr[|X - \mu| \geq t\sigma] &= \Pr\left[(X-\mu)^2 \geq t^2\sigma^2\right] \\ &= \Pr[Y \geq k\mathbb{E}[Y]] \\ &\leq \frac{1}{k} = \frac{1}{t^2}\end{aligned}$$

□

3.4 Two-Point Sampling

Let n be a prime. Choose a, b uniformly at random from $Z_n = \{0, \ldots, n-1\}$. Then let $r_i = (a \cdot i + b) \mod n$. Then for any $i \neq j (\mod n)$, r_i and r_j are independent and uniform in Z_n. Thus, $r_1, \ldots, r_n$ are pairwise independent.

3.5 The Coupon Collectors Problem

The problem is as follows: Suppose that there are n types of coupons, and at each trial a coupon is chosen independently and uniformly at random from all the coupons. How many trials do we expect before we have at least one of each coupon? Also, how likely is it that the number of trials deviate significantly from its expectation?

To answer the first question, let X be the number of trials before we have at least one of each coupon. Let $\{C_1, \ldots, C_X\} \in \{1, \ldots n\}$ be the result of each trial. Trial C_i is thus a *success* if its not any of the previous coupons, i.e. if $C_i \notin \{C_1, \ldots, C_{i-1}\}$.

To analyze further, we split all the trials into epochs, where epoch $i \in \{0, \ldots, n-1\}$ consists of all trials from just after the ith success, until the $(i+1)$th success. Let X_i be the number of trials in epoch i. We then have $X = \sum_{i=0}^{n-1} X_i$. In epoch i, the probability of picking a coupon we have not seen before is $p_i = \frac{n-i}{n}$, as any of the $n-i$ coupons are fine. Each X_i is geometrically distributed, so $\mathbb{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i}$. Thus

$$\mathbb{E}[X] = \sum_{i=0}^{n-1} \mathbb{E}[X_i] = \sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{j=1}^{n} \frac{1}{j} = n \, \mathrm{H}_n = n \ln n + O(n).$$

So, in expectation it it takes $n \ln n + O(n)$ trials to fetch all coupons.

In order to bound the deviance from the expectation, we use a union bound. Let $\mathcal{E}_i^r$ be the event that coupon i is *not* collected in the first r trials. Then

$$\Pr[\mathcal{E}_i^r] = \left(1 - \frac{1}{n}\right)^r \leq e^{-\frac{r}{n}},$$

using the **very useful inequality** $1 + x \leq e^x$. Since the probability of a union of events is at most the sum of probabilities, we have for $r = \beta n \ln n$:

$$\Pr[X > r] = \Pr\left[\bigcup_{i=1}^{n} \mathcal{E}_i^r\right] \leq \sum_{i=1}^{n} \Pr\left[\mathcal{E}_i^r\right] \leq \sum_{i=1}^{n} e^{-\frac{r}{n}} = \sum_{i=1}^{n} n^{-\beta} = n^{-(\beta-1)},$$

which shows that with high probability, the number of trials is close to the expected number.

4 Tail Inequalities

Curriculum: RA 4.

Introduction

This topic is about the *Chernoff bound*, a tailbound that is often better than Markov's or Chebyshev's bounds, but which has some additional requirements on the random variable. It usually allows one to get exponentially small probability bounds.

In this presentation I will go through both topics.

4.1 The Chernoff Bound

The Chernoff bound is a tail bound on a series of Poisson trials. Poisson trails are a series of coin flips, each coin with its own bias. Let $X_1, \ldots X_n$ be independent Poisson trials, such that for $1 \leq i \leq n$, $\Pr[X_i = 1] = p_i$, where $0 < p_i < 1$. Denote $X = X_1 + \ldots + X_n$. We wish to find a tail inequality such that $\Pr[X > (1+\delta)\mathbb{E}[X]] < \epsilon$ for $\delta > 0$ and $\epsilon > 0$. Sometimes we know ϵ and wants to find δ, sometimes the other way round; we might also wish to find $\Pr[X < (1-\delta)\mathbb{E}[X]] < \epsilon$.

Theorem (Chernoff Bound)**.** *Let $X_1, \ldots, X_n$ be independent, Poisson trials such that, for $1 \leq i \leq n$ $\Pr[X_i = 1] = p_i$, where $0 < p_i < 1$. Then for $X = \sum_{i=1}^n X_i$, $\mu = \mathbb{E}[X] = \sum_{i=1}^n p_i$ and any $\delta > 0$*

$$\Pr[X > (1+\delta)\mu] \leq \left(\frac{\mathrm{e}^\delta}{(1+\delta)^{1+\delta}} \right)^\mu.$$

There is an equivalent formulation for $\Pr[X < (1-\delta)\mu]$, which I do not have time for in this presentation. To prove the Chernoff bound, we use the following ideas: instead of analyzing X, we analyse e^{tX} for some $t > 0$. We also use the independence of the trials to turn the expectation of a product into a product of expectations. Finally, we minimize the expression w.r.t. t to get the best possible bound. The proof will take the remaining time of the presentation.

Proof. For any $t > 0$:

$$\begin{aligned}
\Pr[X > (1+\delta)\mu] &= \Pr[e^{tX} > e^{t(1+\delta)\mu}] && \text{by } e^{t(\cdot)} \text{ the inequality of the probability.} \\
&< \frac{\mathbb{E}\left[e^{tX}\right]}{e^{t(1+\delta)\mu}} && \text{By Markov.} \\
&= \frac{\prod_{i=1}^n \mathbb{E}\left[e^{tX_i}\right]}{e^{t(1+\delta)\mu}} && \text{By Independence. } (\star) \\
&= \frac{\prod_{i=1}^n \left(1 + p_i(e^t - 1)\right)}{e^{t(1+\delta)\mu}} && \text{By definition of } \mathbb{E}. \ (\diamond) \\
&\leq \frac{\prod_{i=1}^n e^{p_i(e^t-1)}}{e^{t(1+\delta)\mu}} && \text{By } 1 + x \leq e^x. \\
&= \frac{e^{\sum_{i=1}^n p_i(e^t-1)}}{e^{t(1+\delta)\mu}} \\
&= \frac{e^{(e^t-1)\mu}}{e^{t(1+\delta)\mu}} \\
&= \left(\frac{e^{(e^t-1)}}{e^{t(1+\delta)}} \right)^\mu. && (\clubsuit)
\end{aligned}$$

We have ($\star$) because $\mathbb{E}\left[e^{tX}\right] = \mathbb{E}\left[e^{t\sum_{i=1}^{n} X_i}\right] = \mathbb{E}\left[\prod_{i=1}^{n} e^{tX_i}\right] = \prod_{i=1}^{n} \mathbb{E}\left[e^{tX_i}\right]$. We have ($\diamond$) because $\mathbb{E}\left[e^{tX_i}\right] = p_i e^t + (1-p_i)e^0 = p_i e^t + 1 - p_i = 1 + p_i(e^t - 1)$.

Continuing, we minimize ($\clubsuit$) by finding a suitable t. This is done by taking the derivative w.r.t. t and setting it to 0. The solutions to this is $\ln(1+\delta)$, which we can plug in to ($\clubsuit$) and get

$$\left(\frac{e^{(e^t-1)}}{e^{t(1+\delta)}}\right)^{\mu} = \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu},$$

which is what the theorem states. □

Example

Suppose you play a certain game n times, and win each time independently with probability $\frac{1}{3}$. What is the probability that you win more than half your games?

Let $X = \sum_{i=1}^{n} X_i$, $\mu = \mathbb{E}[X] = \frac{n}{3}$, $\delta = \frac{1}{2}$.

$$\Pr\left[X > \frac{n}{2}\right] = \Pr\left[X > (1+\delta)\mu\right] < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu} = \left(\frac{e^{\frac{1}{2}}}{\left(\frac{3}{2}\right)^{\left(\frac{3}{2}\right)}}\right)^{\frac{n}{3}} < (0.9646)^n$$

This is exponentially small in n.

Contrast with Chebyshev, where $\sigma_X^2 = \sum_{i=1}^{n} p_i(1-p_i) \leq \frac{n}{4}$, so $\sigma_X \leq \frac{\sqrt{n}}{2}$. Thus

$$\Pr\left[X > \frac{n}{2}\right] \leq \Pr\left[|X - \frac{n}{3}| \geq \frac{n}{6}\right] \leq \frac{1}{\left(\frac{\sqrt{n}}{3}\right)^2} = \frac{9}{n}.$$

This is because $\frac{n}{6} = \frac{\sqrt{n}}{3}\frac{\sqrt{n}}{2} = \frac{\sqrt{n}}{3}\sigma_X$.

4.2 Applications of the Chernoff Bound

This lecture concerns algorithms for two problems: *Routing in a parallel computer* and *wiring*. Both algorithms can be analyzed using the Chernoff bound. In this presentation I will go through the routing problem. Proving the Chernoff bound is not a part of this presentation.

4.3 Routing in a Parallel Computer

The problem is as follows: Given a directed graph on N nodes, where each node i initially contains one packet destined for some node $\mathrm{d}(i)$, such that $\mathrm{d}(\cdot)$ is a permutation (that is, each node sends exactly one package, and receives exactly one package). In each step, every edge can carry a single packet. A node may send a package on each outgoing edge (if it has packets). How many steps are necessary and sufficient?

A *route* for a packet is a list of edges it can follow rfom its source to its destination. A solution algorithm implicitly specify routes for all packets. If two packets want to use the same edge, one may have to wait. The *queueing discipline* for an algorithm is how it decides which packet goes first. The algorithm we use here uses FIFO as queueing, and is *oblivious*, meaning that the route it specifies for a packet only depends on the destination of the packet.

Theorem. *For any deterministic oblivious permutation routing algorithm on a network of N nodes each of out-degree d, there is an instance of permutation routing requiring $\Omega\left(\sqrt{\frac{N}{d}}\right)$ steps.*

We want to show that this can be improved by using a random routing algorithm (the Valiant algorithm). The network is the n-dimensional hypercube (with $N = 2^n$ nodes and Nn edges.). The packets are moved across the network by reducing the Hamming distance of the package (bit-fixing strategy, from left to right).

4.4 Valiant Algorithm

The algorithm works in two phases. In the first phase it picks a random $\sigma(i) \in \{1, \ldots, N\}$. Packet v_i travels to $\sigma(i)$ using bit-fixing strategy. In the second phase, packet v_i travels from $\sigma(i)$ to $\mathrm{d}(i)$ using bit-fixing strategy. The queueing discipline is FIFO. For simplicity, we assume that the entire phase one is complete before phase two begins.

Let $\mathrm{delay}(v_i)$ denote the number of steps v_i spends in queues waiting for other packets to move during phase 1. The total number of steps for v_i to wait is at most $n + \mathrm{delay}(v_i)$ (The distance to travel is at most n and the wait is $\mathrm{delay}(v_i)$).
Lemma. *Let $p_i = (e1, \ldots, e_k)$ be the route for v_i and let S_i be the set of other paths intersecting p_i. Then* $\mathrm{delay}(v_i) \leq |S_i|$.

Proof. Define the *lag* (w.r.t. p_i) of a packet $v \in S_i \cup \{v_i\}$ that is ready to move along edge $e_j \in p_i$ at time t to be $t - j$. $\mathrm{delay}(v_i)$ is then the lag of v_i when it finally gets to traverse e_k. We say packet $v \in S_i$ leaves p_i in the last time step where it traverses an edge in p_i.

Suppose $\mathrm{delay}(v_i) > \ell$:

$\Longrightarrow$ At some time t, the lag of v_i increase to $\ell + 1$.

$\Longrightarrow$ At time t, some $v \in S_i$ follows e_j where $\ell = t - j$.
Otherwise, v_i would be following e_j.
At time t this v has lag ℓ w.r.t. p_i.

$\Longrightarrow$ There is a last time t' where some $v \in S_i$ has lag ℓ w.r.t. p_i.
In step t', some $v \in S_i$ is ready to follow $e_{j'} \in P_i$, where $t' - j' = \ell$.

$\Longrightarrow$ In step t', some $\omega \in S_i$ follows $e_{j'} \in p_i$, where $t' - j' = \ell$.
By choice of t', ω leaves p_i with lag ℓ w.r.t. p_i.
(otherwise ω is ready to traverse $e_{j'+1}$ in step $t' + 1$ and have lag ℓ in step $t' + 1$).

(Note that every $\omega \in S_i$ can leave p_i only once). Thus $\mathrm{delay}(v_i) = \ell'$ implies that $|S_i| \geq |\{0, \ldots, \ell' - 1\}| = \ell'$. This again implies that $\mathrm{delay}(v_i) = \ell' \leq |S_i|$. □

Expected Routing Delay

Let H_{ij} indicate that p_i and p_j share at least one edge. Then for any fixed i, $\mathrm{delay}(v_i) \leq |S_i| = \sum_{j=1}^{N} \mathrm{H}_{ij}$. Since $\sigma(\cdot)$ are all independen, the H_{ij} for $j \neq i$ are indpendent Poisson trials. Thus we can use the Chernoff bound for $\mathrm{delay}(v_i)$ if we can estimate $\mathbb{E}\left[\sum_{j=1}^{N} \mathrm{H}_{ij}\right]$.

For each edge e in the hypercube, let $T(e)$ count the number of routes using e. Fix the route $p_i = (e_1, \ldots, e_k)$ with $k \leq n$. Then

$$\sum_{j=1}^{N} \mathrm{H}_{ij} \leq \sum_{j=1}^{N} T(e_\ell) \implies \mathbb{E}\left[\sum_{j=1}^{N} \mathrm{H}_{ij}\right] \leq \mathbb{E}\left[\sum_{j=1}^{N} T(e_\ell)\right] = \sum_{\ell=1}^{k} \mathbb{E}[T(e_\ell)].$$

The expected length of each route is $\frac{n}{2}$ (as we flip each bit), so we have

$$\sum_{\ell=1}^{k} \mathbb{E}[T(e_\ell)] = \sum_{j=1}^{N} \mathbb{E}[|p_j|] = \sum_{j=1}^{N} \frac{n}{2} = \frac{Nn}{2}.$$

Let E be the edges of the network. By symmetry, $\mathbb{E}[T(e)] = \mathbb{E}[T(e')]$ for all $e, e' \in E$. So for any $e \in E$ we have

$$\mathbb{E}[T(e)] = \frac{1}{|E|} \sum_{e \in E} \mathbb{E}[T(e)] = \frac{1}{Nn} \frac{Nn}{2} = \frac{1}{2}.$$

We can now use this result, showing that for $p_i = (e_1, \ldots, e_k)$ we have

$$\sum_{j=1}^{N} \mathrm{H}_{ij} \leq \sum_{\ell=1}^{k} \mathbb{E}[T(e_\ell)] = \frac{k}{2} \leq \frac{N}{2}.$$

Applying the Chernoff bound we get

$$\Pr\left[\sum_{j=1}^{N} \mathrm{H}_{ij} > 6n\right] < 2^{-(1+11)\frac{n}{2}} = 2^{-6n}.$$

Since $\mathrm{delay}(v_i) \leq \sum_{j=1}^{N} \mathrm{H}_{ij}$, this gives that

$$\Pr[\mathrm{delay}(v_i) > 6n] < 2^{-6n}.$$

Now we are almost done, as the we can use the union bound to bound the probability that any node waits for than $6n$ steps:

$$\Pr[\exists v_i : \mathrm{delay}(v_i) \geq 6n] \leq \sum_{i=1}^{N} \Pr \mathrm{delay}(v_i) \geq 6n < N2^{-6n} = 2^n \cdot 2^{-6n} = 2^{-5n}.$$

Thus, since each path has length at most n, $\Pr[\#\text{ of steps in Phase } 1 \ > 7n] < 2^{-5n}$. The analysis for phase 2 is symmetric. Thus with high probability every packet reaches its destination in $14n$ or fewer steps.

5 Data Structures

Curriculum: RA 8 + High Speed Hashing Notes.

5.1 The Data-Structuring Problem

Maintain disjoint sets of items with keys from a totally ordered universe which supports a lot of operations: insert, delete, find, join, concat, split to name some.

5.2 Random Treaps

Combination of binary tree and heap.

5.3 Hashing Fundamentals

Given large universe U of keys and $1 \leq m < U$, a hash function $h : U \to [m]$ is a random variable whose values are functions from $U \to [m]$. Said differently, $h(x)$ is a random variable taking values from $[m]$.

We want hash functions that are fast, small in space, and sometimes we want certain properties. It is also desirable that there are few collisions, that is, the event $h(x) = h(y), x \neq y$ should be very rare.

5.4 Universality

To formalise the idea of rare collision chance, we talk about *universal* hash functions. These are hash functions where for keys x, y chosen independently at random, we have:

$$\Pr\left[h(x) = h(y)\right] \leq \frac{1}{m}$$

Sometimes we only get close to this property, in which case we may have *c-universality*:

$$\Pr\left[h(x) = h(y)\right] \leq \frac{c}{m}$$

5.5 Hashing with Chaining

We have $S \subseteq U$ that we wish to store and retrieve single key of in expected constant time. We have $|S| = n$ and $n \leq m$. Then pick hash function $h \colon U \to [m]$ and create array L containing m lists. Then we can find keys in S by looking in $L[h(x)]$. Now want to show that $L[h(x)]$ has expected size 1.

Assume that $x \notin S$ for worst case. Assume h is universal. Then the expected length of $L[h(x)]$ is:

$$\mathbb{E}[|L[h(x)]|] = \mathbb{E}\left[\sum_{y \in S} [h(y) = h(x)]\right] = \sum_{y \in S} \mathbb{E}[[h(y) = h(x)]] = \sum_{y \in S} \frac{1}{m} = \frac{n}{m} \leq 1$$

5.6 Two-Level Hashing

Imagine storing a static set. Using a special technique, we can achieve *worst case* constant time lookup with linear space.

Step 1: Pick universal hash function $h : U \to [n]$. Let S_i be the keys that hash to i. Let $s_i = |S_i|$ and $B = \sum_{i=0}^{n-1} \binom{s_i}{2}$. If $B \geq n$, try again.

Step 2: For each $i \in [n]$, pick universal hash function $h_i : U \to [s_i(s_i - 1)]$ until h_i has no collisions on S_i.

Step 1 takes expected $O(n)$ time:

$$B = \sum_{x,y \in S, x \neq y} [h(x) = h(y)]$$

$$\mathbb{E}[B] = \sum_{x,y \in S, x \neq y} \Pr[h(x) = h(y)] \leq \binom{n}{2} \frac{1}{n} = \frac{n}{2}$$

$$\Pr[B \geq 2\mathbb{E}[B]] \leq \frac{\mathbb{E}[B]}{2\mathbb{E}[B]} = \frac{1}{2}$$

Expected number of tries in step 1 is therefore 2, each try is linear time, so $O(n)$.

Step 2 takes $O(s_i)$ for every s_i.

$$B_i = \sum_{x,y \in S_i, x \neq y} [h(x) = h(y)]$$

$$\mathbb{E}[B_i] = \sum_{x,y \in S_i, x \neq y} \Pr[h(x) = h(y)] \leq \binom{si}{2} \frac{1}{s_i(s_i - 1)} = \frac{1}{2}$$

$$\Pr[B_i \geq 2\mathbb{E}[B_i]] \leq \frac{\mathbb{E}[B_i]}{2\mathbb{E}[B_i]} = \frac{1}{2}$$

Expected number of tries before $B_i < 1$ is 2 and each try is linear time, so $O(s_i)$.

6 Linear Probing

Curriculum: "Linear Probing with 5-Independent Hashing" Article.

Introduction

Linear probing is a way of hashing values from a large universe U of keys to a smaller range $[t]$. We care about space, time and properties of the hash function. For linear probing, we need a hash function h with the property that it is 5-independent, meaning that any 5 distinct keys hash independently and uniformly in $[t]$. Hashing with linear probing is one of the most efficient ways in practice to implement hash tables because of nice cache behavior.

6.1 Hash Table with Linear Probing

We want to maintain s subset $S \subseteq U$, $|S| = n$. Let $t \geq \frac{3}{2}n$ and make a table T of size t with initial values of **nil**. We then pick a hash function $h : U \to [t]$ and do inserts as follows: If $T[h(x)]$ is **nil**, then assign $T[h(x)] \leftarrow x$; otherwise move forward in the table (with wrap-around if you reach the end) until we find an empty spot. To check for membership of x, we do the same approach and check if the current element is x. Deletion is more tricky – requires finding a value further that can replace the value of the deleted value.

Let the distance from x to the next **nil** be the *cost* of x.

If the hash-function is fully random, then $\mathbb{E}[\text{cost}(x)] \in O(1)$. But, we can make the same claim with a 5-independent hash function. This is what we will show now.

k-independence

As mentioned a function h is k-independent if any k distinct keys hash independently and uniformly in $[t]$. Let p be a prime and let $a_0, \ldots, a_{k-1} \in [p]$ be chosen uniformly and independently at random. Then

$$h(x) = \left(\left(\sum_{i=0}^{k-1} a_i x^i \right) \mod p \right) \mod t$$

is a k-independent hash function (almost uniform, which is OK if $t \ll p$).

6.2 Linear Probing with 5-Independence

We want to bound the expected size $\mathbb{E}[|\,\mathrm{R}(q)|]$ of the longest filled run $\mathrm{R}(q)$ containing $h(q)$ for some key q, as this is an upper bound on the cost of querying q.

We use the trick of *dyadic ℓ-intervals* to do this. These are intervals of length 2^ℓ starting at positions i with $i \mod 2^\ell = 0$. The idea is to show that if $\mathrm{R}(q)$ is large, a dyadic interval of similar size close to $h(q)$ has many hits.

The following lemmas and proofs should probably be skipped in the presentation:
Lemma. *Let R be any maximal, filled run. If $|R| \geq 2^{\ell+2} = 4 \cdot 2^\ell$, one of the first 4 ℓ-intervals intersecting R has at least $\frac{3}{4}2^\ell$ keys in $S \setminus \{q\}$ hashing into it.*

Proof. Let I_0, I_1, I_2, I_3 be the first 4 ℓ-intervals intersecting R. Let $L = (\cup_{i=0}^{3} I_i) \cap R$ be the a prefix of R consisting of the intervals. We have $|L| \geq 3 \cdot 2^\ell + 1$, since the first interval might only intersect with 1 element, while the 3 others fully intersect. At least $|L|$ keys hash to L, so some I_i is hit by $\frac{3}{4}2^\ell$ keys from $S \setminus \{q\}$. □

Lemma. *If $2^{\ell+2} \leq |\,\mathrm{R}(q)| < 2^{\ell+3}$, one of the 12 following ℓ-intervals has at least $\frac{3}{4}w^\ell$ keys in $S \setminus \{q\}$ hashing into it:*

- *The ℓ-interval I_q containing $h(q)$; or one of*
- *the 8 ℓ-intervals to the left of I_q; or one of*
- *the 4 ℓ-intervals to the right of I_q.*

Proof. Since $|R(q)| < 8 \cdot 2^\ell$, the first ℓ-interval intersecting $R(q)$ is at most 8 intervals to the left of I_q. The first 4 ℓ-intervals intersecting $R(q)$ are therefore among the 12 intervals mentioned, and we use previous lemma. (One extreme is that the 4 intervals starts at the 8th interval to the left of I_q; the other is that the interval containing I_q is the first intersecting interval, meaning that the remaining 3 are to the right of I_q). □

Corollary 1. *Let P_ℓ be the probability that any given ℓ-interval has at least $\frac{3}{4}2^\ell$ keys in $S \setminus \{q\}$ hashing into it. Then*

$$\Pr\left[2^{\ell+2} \leq |R(q)| < 2^{\ell+3}\right] \leq 12P_\ell. \qquad \textit{by union bound.}$$

Thus we have

$$\mathbb{E}[|R(q)] \leq 3 + \sum_{\ell=0}^{\log_2 t} 2^{\ell+3} \cdot 12P_\ell \in O\left(1 + \sum_{\ell=0}^{\log_2 t} 2^\ell \cdot P_\ell\right)$$

because the corollary only tells us about runs of length at least 4, which is where the added constant 3 comes from ($2^{0+3} = 8$, so we have to consider length 4 as its own case; it has probability at most 1 of having size 3, so we just add 3, which is sufficient for the bound).

We need to bound P_ℓ to get $O(1)$ expected cost. Assume h is 5-independent.
Claim. *The expected cost is $O(1)$.*

Proof. Given an ℓ-interval I, for $x \in S \setminus \{q\}$, let $X_x = [h(x) \in I]$. Then $X = \sum_{x \in S \setminus \{q\}} X_x$ is the number of keys in $S \setminus \{q\}$ that hash into I, and $\mu = \mathbb{E}[X] \leq n\frac{2^\ell}{t} \leq \frac{2}{3}2^\ell$. Since h is 5-independent, the variables X_x are 4-wise independent. To make the proof, we use a 4th moment bound.

Since $\frac{2}{3}2^\ell \geq \mu$, we have $\sqrt{\frac{2}{3}}2^\ell \geq \sqrt{2^\ell \mu}$.

$$\begin{aligned} X \geq \frac{3}{4}2^\ell &\implies X - \mu \geq \frac{3}{4}2^\ell - \mu \geq \left(\frac{3}{4} - \frac{2}{3}\right)2^\ell \\ &\implies X - \mu \geq \frac{1}{12}2^\ell > \frac{\sqrt{2^\ell}}{10}\sqrt{\mu} \\ &\implies |X - \mu| \geq \frac{\sqrt{2^\ell}}{10}\sqrt{\mu}. \end{aligned}$$

From which we get that

$$P_\ell = \Pr[X \geq \frac{3}{4}2^\ell] \leq \Pr\left[|X - \mu| \geq \frac{\sqrt{2^\ell}}{10}\sqrt{\mu}\right]$$

Now we use a theorem that allows us to say that $\Pr\left[|X - \mu| \geq d\sqrt{\mu}\right] \leq \frac{4}{d^4}$, for $d > 0$ to say that

$$P_\ell \leq \Pr\left[|X - \mu| \geq \frac{\sqrt{2^\ell}}{10}\sqrt{\mu}\right] \leq \frac{40000}{2^{2\ell}}.$$

Finally, we can now say that

$$\mathbb{E}[\text{cost(q)}] \leq \mathbb{E}[|R(q)|] \in O\left(1 + \sum_{\ell=0}^{\log_2 t} 2^\ell P_\ell\right) \subseteq O\left(1 + \sum_{\ell=0}^{\log_2 t} 2^\ell \frac{40000}{2^{2\ell}}\right) = O(1).$$

This is what we wanted to prove. What remains is to show the theorem used above to get the bound on P_ℓ. We do not have time for that now. □

6.3 Proposed Proof Sketch for Presentation

The overall approach for the proof is as follows: We bound the expected cost by the size of *runs*, and show that a run of large size implies that there is an ℓ-dyadic set that is near full. Thus, since runs are only large when some ℓ-set is near full, we can bound the probability of a run being large by the probability that some ℓ-set is near full. We can show this probability is small using (among other things) the 5-independence property of the hash-function. One can then go back and use this knowledge to prove that the expected cost is $O(1)$.

There are two pieces of information that we are going to take for granted for the sake of time:

(1) The first is a bound on the expected run size:

$$\mathbb{E}[|R(q)] \leq O\left(1 + \sum_{\ell=0}^{\log_2 t} 2^\ell \cdot P_\ell\right).$$

Here P_ℓ is the probability that any given ℓ-interval is near full, and t is the size of the table. We wish to bound P_ℓ, for which we need a second piece of information.

(2) P_ℓ is bounded as follows:

$$P_\ell = \Pr\left[X \geq \frac{3}{4}2^\ell\right] \leq \Pr\left[|X - \mu| \geq \frac{\sqrt{2^\ell}}{10}\sqrt{\mu}\right]$$

Taking outset in (2), I will now use the 5-independence property of the hash function to give a 4th moment bound on P_ℓ. We need 5-independence, as we desire keys from $S \setminus \{q\}$ to be hashed 4-independently, no matter the hash-value $h(q)$.

Theorem. *If $X_0, \ldots, X_{n-1} \in \{0, 1\}$ are 4-wise independent, $X = \sum_{i\in[n]} X_i$ and $\mu = \mathbb{E}[X] \geq 1$, then for $d > 0$,*

$$\Pr\left[|X - \mu| \geq d\sqrt{\mu}\right] \leq \frac{4}{d^4}.$$

Thus by the theorem and (2) we have $P_\ell \leq \Pr\left[|X - \mu| \geq \frac{\sqrt{2^\ell}}{10}\sqrt{\mu}\right] \leq \frac{40000}{2^{2\ell}}$. Combining with (1) we get $\mathbb{E}[|R(q)|] = O(1)$. What remains is to actually prove the theorem.

Proof. Let $X_0, \ldots, X_{n-1} \in \{0, 1\}$ be 4-wise independent. Let $p_i = \Pr[X_i = 1] = \mathbb{E}[X_i]$, $X = \sum_{i\in[n]} X_i$ and $\mu = \mathbb{E}[X] = \sum_{i\in[n]} p_i$ and $d > 0$. The variance σ^2 of X is then the sum of variances by 4-independence, and each variance σ_i^2 of X_i is

$$\sigma_i^2 = \mathbb{E}\Big[(X_i - p_i)^2\Big] = p_i(1 - p_i)^2 + (1 - p_i)p_i^2 = p_i(1 - p_i) \leq p_i.$$

It follows that $\sigma^2 = \sum \sigma_i^2 \leq \mu$. We then have

$$\Pr\left[|X - \mu| \geq d\sqrt{\mu}\right] = \Pr\left[(X - \mu)^4 \geq d^4\mu^2\right] \leq \frac{\mathbb{E}[(X - \mu)^4]}{d^4\mu^2} \qquad \text{by Markov's.}$$

We now want to bound $\mathbb{E}\big[(X-\mu)^4\big] \leq 4\mu^2$ such that $\frac{\mathbb{E}\left[(X-\mu)^4\right]}{d^4\mu^2} \leq \frac{4\mu^2}{d^4\mu^2} = \frac{4}{d^4}$.

$$\begin{aligned}
\mathbb{E}\Big[(X-\mu)^4\Big] &= \mathbb{E}\left[\left(\sum_{i\in[n]}(X_i - p_i)\right)^4\right] \\
&= \sum_{i,j,k,l\in[n]} \mathbb{E}[(X_i-p_i)(X_j-p_j)(X_k-p_k)(X_l-p_l)] \qquad \text{l.o.e.}
\end{aligned}$$

If $i \notin \{j,k,l\}$ then by 4-independence $\mathbb{E}[(X_i-p_i)(X_j-p_j)(X_k-p_k)(X_l-p_l)]$ is zero, by observing that $\mathbb{E}[(X_i-p_i)] = p_i - p_i = 0$ can be factored out. So if the term is nonzero then either all 4 terms are equal, or two pairs of two terms (chosen in any one of $\binom{4}{2}$ ways) are equal. This allows us to rewrite $\mathbb{E}\big[(X-\mu)^4\big]$. Here we go!

$$\begin{aligned}
\mathbb{E}\Big[(X-\mu)^4\Big] &= \sum_{a\in[n]} \mathbb{E}\Big[(X_a-p_a)^4\Big] + \binom{4}{2} \sum_{\substack{a,b\in[n]\\a<b}} \mathbb{E}\Big[(X_a-p_a)^2\Big]\mathbb{E}\Big[(X_b-p_b)^2\Big] \\
&\leq \sum_{a\in[n]} \mathbb{E}\Big[(X_a-p_a)^2\Big] + \binom{4}{2} \sum_{\substack{a,b\in[n]\\a<b}} \mathbb{E}\Big[(X_a-p_a)^2\Big]\mathbb{E}\Big[(X_b-p_b)^2\Big] \\
&= \sum_{a\in[n]} \sigma_a^2 + 6\sum \sum_{\substack{a,b\in[n]\\a<b}} \sigma_a^2\sigma_b^2 \\
&\leq \sum_{a\in[n]} \sigma_a^2 + 6\sum \sum_{\substack{a,b\in[n]\\a<b}} p_a p_b \qquad \text{as } \sigma_i \leq p_i \\
&\leq \sum_{a\in[n]} \sigma_a^2 + 3 \sum_{a,b\in[n]} p_a p_b \\
&= \sum_{a\in[n]} \sigma_a^2 + 3\left(\sum_{i\in[n]} p_i\right)^2 \\
&\leq \mu + 3\mu^2 \\
&\leq 4\mu^2.
\end{aligned}$$

□

7 The Probabilistic Method

Curriculum: RA 5.

Introduction

This topic deals with a method of proving the existence of some object satisfying some desirable properties. The method is called *the probabilistic method* and this presentation covers some of the content of the corresponding Chapter 5 in the RA book.

1. Define the probabilistic method and it's core ideas. (∼4 min?)
2. Show example of usage: The MAX-SAT problem. (remainder of time)

7.1 The Probabilistic Method

The core idea of the probabilistic method is very simple:

(1) Any random variable $X \in \mathbb{R}$ takes some value $x \leq \mathbb{E}[X]$ and some value $x' \geq \mathbb{E}[X]$. We can use this fact to determine the existence of objects that has at least/most the expected value.

(2) If a random object taken from a universe U has nonzero probability of satisfying a property P, then there must be an object in U satisfying P. This guarantees the existence of such an object, as we have a positive probability of finding it using a random procedure.

One simple example is the Max-Cut-problem, a problem where we wish to maximize the size of a cut dividing a graph $G(V, E)$ into two subsets A, and B, for which we can use the probabilistic method to show that there exists a division with a cut C of size $\geq \frac{|E|}{2}$.

Proof. We have a graph $G(V, E)$. Assign each vertex v uniformly at random to either A or B. Then for any edge e we have $\Pr[e \in C] = \frac{1}{2}$. This means that

$$\mathbb{E}[|C|] = \mathbb{E}\left[\sum_{e \in E}[e \in C]\right] = \sum_{e \in E} \Pr[e \in C] = \frac{|E|}{2},$$

by linearity of expectation and the probability of e being in C. By (1) of the probailistic method, then there is such a division of G. □

Sometimes proofs are constructive, in which case we can use the procedure in the proof to actually find such an instance where the property holds. This is however very difficult for many problems.

7.2 Max-Sat Problem

Next, consider the Max-Sat problem. First we use the probabilistic method to show that there is a truth assignment that satisfies at least $\frac{m}{2}$ clauses, m being the number of clauses. Then, we use this to construct two approximations algorithms Max-Sat-Simple and Max-Sat-RR for the problem. Finally, we combine the two algorithms into a better approximation algorithm.

Formally, the problem is as follows: Given a set of m clauses in conjunctive normal form over n variables, find a truth assignment that maximizes the number of satisfied clauses (trivial; probably skip this at presentation). This is NP-hard, so we approximate.
Claim. *There is a truth assignment of m clauses in conjunctive normal form of size $\frac{m}{2}$.*

Proof. Set each literal to either TRUE or FALSE uniformly at random. Let $Z_i, i \in [m]$ be an indicator variable for whether the ith clause is satisfied. Any clause containing k distinct literals is not satisfied with probability 2^{-k}. Thus each clause is satisfied with probability $1 - 2^{-k} \geq \frac{1}{2}$, implying that $\mathbb{E}[Z_i] \geq \frac{1}{2}$ for all clauses. Since there are m clauses, by linearity of expectation we have that the expected number of satisfied clauses $\mathbb{E}[\sum_{i=1}^{m} Z_i] \geq \frac{m}{2}$. □

We now use this claim to construct an approximation algorithm for MAX-SAT.

Algorithm: MAX-SAT-SIMPLE

Randomly assigning the literals values is a $\frac{1}{2}$-approximation algorithm because in expectation it satisfies $\frac{m}{2}$ clauses and we can at most satisfy all m clauses. Call this algorithm MAX-SAT-SIMPLE. Specifically, it is an $(1 - 2^{-k})$-approximation if there are at least k distinct literals in each clause. If there are at least 2 distinct literals in each clause, it is a $\frac{3}{4}$-approximation. We will combine this result with the following algorithm.

Algorithm: MAX-SAT-RR

The idea is to use linear programming and randomized rounding to get an approximation algorithm. Let C_j^+ (C_j^-) be the set of indices of variables that are unnegated (negated) in clause j. Then the following linear program solves MAX-SAT:

$$\begin{aligned} &\text{maximize } \sum_{j=1}^{m} z_j \\ &\text{subject to } y_i, z_j \in \{0,1\} && \forall i, j \\ &\qquad \sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \geq z_j && \forall j \end{aligned}$$

We relax this linear program:

$$\begin{aligned} &\text{maximize } \sum_{j=1}^{m} \hat{z}_j \\ &\text{subject to } \hat{y}_i, \hat{z}_j \in [0,1] && \forall i, j \\ &\qquad \sum_{i \in C_j^+} \hat{y}_i + \sum_{i \in C_j^-} (1 - \hat{y}_i) \geq \hat{z}_j && \forall j \end{aligned}$$

Now use randomized rounding to set each x_i to TRUE with probability $\hat{y}_i$. The algorithm that first solves the relaxed linear program and then assigns the literals in this manner is a $\left(1 - \frac{1}{e}\right)$-approximation.

Claim. *MAX-SAT-RR is a $\left(1 - \frac{1}{e}\right)$-approximation algorithm.*

The proof of this claim we delay a bit. First we show the following lemma:

Lemma. *Let $\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k$. The probability that a given clause C_j with k_j distinct literals is satisfied by MAX-SAT-RR is at least $\beta_{k_j} \hat{z}_j$.*

Proof. Assume without loss of generality that C_j has the form $x_1 \vee \ldots \vee x_{k_j}$ as we can renumber any literal as we like, and interchange any literal with its opposite in an equivalent instance.

Each x_i is set to FALSE with probability $1 - \hat{y}_i$, meaning that C_j is satisfied with probability

$$1 - \prod_{i=1}^{k_j} (1 - \hat{y}_i) \geq 1 - \left(1 - \frac{\hat{z}_j}{k_j}\right)^{k_j} \tag{$\star$}$$

The inequality holds, as the product is minimized when all $\hat{y}_i$ are equal. Since $\hat{z}_j = \sum_{i=1}^{k_j} \hat{y}_i$, we have $\hat{y}_i = \frac{\hat{z}_j}{k_j}$. Denote the right-side of ($\star$) as $\mathrm{f_k}(z) = 1 - \left(1 - \frac{z}{k}\right)^k$.

This function $\mathrm{f_k}$ is concave for $z \in [0, 1]$, meaning that all linear interpolations between two function values in that range is less than or equal to the function value at that interpolation. Pick the two function values as $\mathrm{f_k}(0) = 0$ and $\mathrm{f_k}(1) = \left(1 - \frac{1}{k}\right)^k = \beta_k$, as this is the range of a probability. Thus for $z \in [0, 1]$ we have $\mathrm{f_k}(z) \geq z\beta_k$ by the concavity of $\mathrm{f_k}$. Specifically, we can use this to say that

$$\Pr\left[C_j \text{ is satisfied}\right] \geq \mathrm{f_{k_j}}(\hat{z}_j) \geq \beta_j \hat{z}_j,$$

which is what the lemma states. □

Observe that $\beta_k \geq 1 - e^{-1}$, by the inequality $1 + x \leq e^x$:

$$1 - x \leq e^{-x} \implies 1 - \frac{1}{k} \leq e^{-\frac{1}{k}} \implies \left(1 - \frac{1}{k}\right)^k \leq e^{-1}.$$

We can now prove the claim that MAX-SAT-RR is an $(1 - e^{-1})$-approximation.

Proof. Let $Z_j = [C_j \text{ is satisfied}]$. By the lemma, $\mathbb{E}[Z_j] \geq \beta_{k_j} \hat{z}_j \geq \left(1 - \frac{1}{e}\right)\hat{z}_j$, because $\beta_k \geq 1 - \frac{1}{e}$. Then the expected number of satisfied clauses is:

$$\begin{aligned}\sum_{j=1}^{m} \mathbb{E}[Z_j] \geq \sum_{j=1}^{m} \left(1 - \frac{1}{e}\right)\hat{z}_j &= \left(1 - \frac{1}{e}\right) \sum_{j=1}^{m} \hat{z}_j \\ &\geq \left(1 - \frac{1}{e}\right) \sum_{j=1}^{m} z_j\end{aligned}$$

The last inequality is due to the fact that the relaxed solution is always at least as good as the solution to the strict problem Thus it is a $\left(1 - \frac{1}{e}\right)$-approximation algorithm. □

Combining MAX-SAT-SIMPLE and MAX-SAT-RR

Finally, we show that the algorithm which runs both MAX-SAT-SIMPLE and MAX-SAT-RR and then takes the max is and $\frac{3}{4}$-approximation algorithm. Denote n_1 as the expected number of satisfied clauses for any instance I of MAX-SAT-SIMPLE and ditto for n_2 for MAX-SAT-RR.

Claim. $\max\{n_1, n_2\} \geq \frac{3}{4} \sum_{j=1}^{m} \hat{z}_j$.

Proof. It is sufficient to show that $\frac{n_1 + n_2}{2} \geq \frac{3}{4} \sum_{j=1}^{m} \hat{z}_j$. Let S^k be the set of clauses with k distinct literals. Then we have

$$n_1 \geq \sum_k \sum_{C_j \in S^k} \left(1 - 2^{-k}\right) \geq \sum_k \sum_{C_j \in S^k} \left(1 - 2^{-k}\right)\hat{z}_j,$$

because $0 \leq \hat{z}_j \leq 1$. We also have

$$N_2 \geq \sum_k \sum_{C_j \in S^k} \beta_k \hat{z}_k.$$

Putting together, we get

$$\begin{aligned}
\frac{n_1+n_2}{2} &\geq \frac{1}{2}\sum_k \sum_{C_j \in S^k} \Big(\Big(1-2^{-k}\Big)+\beta_k\Big)\hat{z}_j \\
&\geq \frac{1}{2}\sum_k \sum_{C_j \in S^k} \frac{3}{2}\hat{z}_j && \text{by case match } k=1,\, k=2,\, k\geq 3. \\
&= \frac{3}{4}\sum_{j=1}^{m} \hat{z}_j && \text{as we sum over all distinct } \hat{z}_j. \quad \square
\end{aligned}$$

8 Algebraic Techniques

Curriculum: RA 7.

Introduction

This topic is about algebraic techniques using randomness to verify equality between elements. In our case, matrices and polynomials. In this presentation I will first go through the concept of fingerprinting. Then, we will show this applied to the verification of matrix multiplication with Freivald's algorithm. Finally, we will go through the Schwartz-Zippel theorem for multivariate polynomial identity verificaion, and its proof.

8.1 Fingerprinting

Alice wants to send $a \in U$ to Bob, who wants to check if $a = b$. Deterministically, Alice needs to send $\log_2 |U|$ bits, as she would otherwise send the same message for two distinct elements in U.

In **fingerprinting** Alice can choose r random bits and use them to select a random *fingerprint* function $F : U \to S$. Alice then sends $F(a)$ instead, and Bob checks whether $F(A) = F(B)$. This can be done with $r + \log_2 |S|$ bits, but might give the wrong answer (false positives).

We want r and $|S|$ small, F fast and $\Pr_F[F(a) = F(b) \mid a \neq b]$ small. We now apply this idea to matrix multiplication verification.

8.2 Freivald's Technique

Freivald's algorithm is used to verify matrix multiplication: Given matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, we want to verify that $\mathbf{AB} = \mathbf{C}$. It does so in a way that allows us to avoid doing the actual multiplication. Formally, $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}^{n \times n}$, where $\mathbb{F}$ is a field (in a field of whole numbers over some prime p, $\mathbb{Z}_p$, a polynomial of degree d has at most d roots).
Theorem. *For $\mathbf{r} \in \{0,1\}^n$ chosen uniformly at random,*

$$\Pr\left[\mathbf{A}(\mathbf{Br}) = \mathbf{Cr} \mid \mathbf{AB} \neq \mathbf{C}\right] \leq \frac{1}{2}$$

The idea is to compute $\mathbf{x} = \mathbf{Br}$, $\mathbf{y} = \mathbf{Ax}$, and $\mathbf{z} = \mathbf{Cr}$ in $O(n^2)$ time and then compare $\mathbf{y} = \mathbf{z}$ in linear time. By repeating t times, we can in $O(tn^2)$ time get an error probability of 2^{-t}. This is faster than the best matrix multiplication scheme we have. The *fingerprint* function in this case is $F(\mathbf{AB}) = \mathbf{A}(\mathbf{Br})$.

Proof. Suppose $\mathbf{AB} \neq \mathbf{C}$. Let $\mathbf{D} = \mathbf{AB} - \mathbf{C}$. Then $\mathbf{D} \neq 0$ and there must be some indices i, j

where $\mathbf{D}_{ij} \neq 0$. Then

$$\begin{aligned}
\Pr_{\mathbf{r}}[\mathbf{A}(\mathbf{Br}) = \mathbf{Cr}] &= \Pr_{\mathbf{r}}[\mathbf{Dr} = 0] \\
&\leq \Pr_{\mathbf{r}}[\mathbf{D}_i\mathbf{r} = 0] && \text{Since at least } \mathbf{D}_i\mathbf{r} \text{ must be 0 if } \mathbf{Dr} = 0 \\
&= \Pr_{\mathbf{r}}\left[\sum_k \mathbf{D}_{ik}\mathbf{r}_k = 0\right] \\
&= \Pr_{\mathbf{r}}\left[\mathbf{r}_j = \frac{-1}{\mathbf{D}_{ij}} \sum_{k \neq j} \mathbf{D}_{ik}\mathbf{r}_k\right] && \text{By subtracting terms and dividing.} \\
&\leq \frac{1}{2}
\end{aligned}$$

□

The last step uses the *principle of deferred decision*: All entries in $\mathbf{r}$ are independent, so we can pretend that $\mathbf{r}_j$ is chosen last. Since it is chosen uniformly from $\{0, 1\}$, the chance of getting the correct value is at most $\frac{1}{2}$, as one outcome will definitely not yield the correct result, while the other might.

Polynomials

Given polynomials $P_1(x), P_2(x) \in \mathbb{F}[x]$ of degree $\leq d$ as black boxes. How do we check if $P_1 = P_2$?

Theorem. *Let $Q \in \mathbb{F}[x]$ be a polynomial with degree d, let $\mathbb{S} \in \mathbb{F}$ be finite and $|\mathbb{S}| \geq d + 1$. For $x \in \mathbb{S}$ picked uniformly at random*

$$\Pr_x[Q(x) = 0 \mid Q \neq 0] \leq \frac{d}{|\mathbb{S}|} \tag{$\star$}$$

Proof. A nonzero polynomial of degree d over any field has at most d distinct roots. The polynomial evaluates to zero at its roots. Thus, $\mathbb{S}$ contains at most d roots, and the probability of picking one of them is at most $\frac{d}{|\mathbb{S}|}$. □

Read ($\star$) as *the probability that $Q(x)$ evaluates to 0 given that Q is a nonzero polynomial (i.e. is always 0), is less than $\frac{d}{|\mathbb{S}|}$*. The idea is to choose $Q = P_1 - P_2$, which yields a polynomial of degree at most $2d$. The ways $\mathbb{S}$ is chosen means that $\frac{d}{|\mathbb{S}|} \leq \frac{1}{2}$. Thus repeating the above t times we get an error probability of 2^{-t}. This procedure is nice because we can compute the fingerprint function $F(P_1 - P_2) = (P_1 - P_2)(x) = P_1(x) - P_2(x)$ without computing $P_1 - P_2$.

8.3 Schwartz-Zippel Theorem

The Schwartz-Zippel theorem is used for polynomial identity verification, as in we just discussed. But this applies to the multivariate case as well. The degree for a polynomial $x_1^{d_1}x_2^{d_2}\cdots x_n^{d_n}$ is $d_1 + d_2 + \ldots + d_n$. The total degree of a polynomial is the maximum degree of any of its terms.

Theorem. *Let $Q(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a multivariate polynomial with total degree d. Fix any finite set $\mathbb{S} \subseteq \mathbb{F}$ and let $r_1, \ldots, r_n$ be chosen independently and uniformly at random from $\mathbb{S}$. Then*

$$\Pr\left[Q(r_1, \ldots, r_n) = 0 \mid Q \neq 0\right] \leq \frac{d}{|\mathbb{S}|}.$$

The remainder of this presentation is the proof of the theorem.

Proof. The proof is by induction over n, the number of variables. For $n = 1$ (univariate polynomials), we have already proven that the bound holds.

For $n \geq 2$, assume that the theorem holds for all smaller n. Let $Q \neq 0$ (there is nothing to prove if $Q = 0$) and $k > 0$ be the max exponent of x_n (if $k = 0$, x_n has no effect on the polynomial). Then, there exists $Q_0, \ldots, Q_k$ such that

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} Q_i(x_1, \ldots, x_{n-1})x_n^i$$

Why? You can construct Q_i by grouping all terms containing x_n^i factoring x_n^i out.

By construction $Q_k \neq 0$ and the degree of Q_k is less than $d - k$ (because we factored out x_n^k). Pick $r_1, \ldots, r_{n-1}$ independently and uniformly at random from $\mathbb{S}$. By the principle of deferred dicision, we can pretend that $r_1, \ldots, r_{n-1}$ are picked before r_n. Let $C_i = Q_i(r_1, \ldots, r_{n-1})$ be the evaluation of Q_i on these randomly chosen elements. C_i is a new random variable depending on $r_1, \ldots, r_{n-1}$. By the induction hypothesis we have that $\Pr[C_k = 0] \leq \frac{d-k}{|\mathbb{S}|}$, since we know that C_i is the evaluation of a nonzero polynomial.

If $C_k \neq 0$, then $q(x) = \sum_{i=1}^{k} C_i x^i = Q(r_1, \ldots, r_{n-1}, x) \neq 0$ has total degree k.

This implies, using the induction hypothesis, that for u.a.r $r_n \in \mathbb{S}$, we have $\Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{k}{|\mathbb{S}|}$.

Finally,

$$\Pr[Q(r_1, \ldots, r_n) = 0] \leq \Pr[C_k = 0] + \Pr[q(r_n) = 0 \mid C_k \neq 0] \leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} = \frac{d}{|\mathbb{S}|}.$$

□

The final calculation uses that $\Pr[\mathcal{E}_1] \leq \Pr[\mathcal{E}_1 \mid \overline{\mathcal{E}_2}] + \Pr[\mathcal{E}_2]$. This is easy to intuitively see by drawing a Venn diagram.

9 Data Stream Algorithms

Curriculum: Data Stream Algorithms Notes.

9.1 Basic Streaming Model

This topic is about answering queries effectively about a large stream of data. Queries could be for example the minimum or maximum element, the number of elements or the average, etc. Some queries are harder to answer. We will look at a deterministic algorithm and a sketching algorithm (*Misra-Gries* and *Count Sketch*) for frequency estimation, i.e. how many times an element x appeared in the stream.

A *stream* is a sequence $\sigma = x_0, x_1, \ldots, x_{n-1} \in [u]$, where n and u are large. The elements of the stream can only be accesses one element at a time and in order. We have little space and must answer questions about the part of the stream we have seen so far.

We focus on the frequency estimation query: Given an element $x \in [u]$, estimate the frequency $f_x = |\{i \in [n] \mid x_i = x\}|$. We do not know the queried element x before processing the stream, and so must be prepared to answer for any x. Denote the estimate $\hat{f}_x$.

9.2 Deterministic Algorithm: Misra-Gries

Claim. *Let $k \in \mathbb{N}$. Using only $O(k)$ words of $O(\log n + \log u)$ bits each, we can maintain values $\hat{f}_x \geq 0$ for all $x \in [u]$ such that $f_x - \frac{n}{k} \leq \hat{f}_x \leq f_x$.*

To show the claim, we introduce the Misra-Gries algorithm. We only explicitly store $\hat{f}_x > 0$, i.e. all estimates are implicitly initialized as 0. When processing $x \in \sigma$, first set $\hat{f}_x \leftarrow \hat{f}_x + 1$, then if the set $A = \{y \in [u] \mid \hat{f}_x > 0\}$ has $|A| \geq k$, set $\hat{f}_x \leftarrow \hat{f}_x - 1$ for $y \in A$ (effectively, this means that we only have at most k estimates that are > 0 stored).

In practical settings, we maintain a dictionary A, adding or incrementing key values when they appear. If $|A| \geq k$ at some point, we delete all keys with value 1 and decrement all other key values by 1.

Theorem. *After processing n elements, we have*

$$f_x - \frac{n}{k} \leq \hat{f}_x \leq f_x \text{ for all } x \in [u].$$

If we can prove this, the claim holds.

Proof. The algorithm starts with $\hat{f}_x = f_x = 0$ and only increases $\hat{f}_x$ when f_x increases, so clearly $\hat{f}_x \leq f_x$.

Each time $\hat{f}_x$ is decreased x is part of a set $A \subseteq [u]$ of size $|A| \geq k$ where every $y \in A$ has $\hat{f}_x > 0$ and all are decreased at the same time. The total number of rounds of decreases is therefore at most $\frac{n}{k}$ (as you have to fill up A with k distinct elements before anything is decreased. Or think of it like this: each decrease subtracts k from the total number n of elements we observe. We can do this at most $\frac{n}{k}$ times before we run out of elements). In particular, we have that $\hat{f}_x$ is decreased by at most $\frac{n}{k}$. □

9.3 Randomized Algorithm: Basic Count Sketch

Suppose that the stream is a sequence of n (key, value)-pairs

$$\sigma = (x_0, \Delta_0), (x_1, \Delta_1), \ldots, (x_{n-1}, \Delta_{n-1}) \in [n] \times \{-u, \ldots, u\}.$$

For some key x, let I_x be the set of all indices i such that $x_i = x$. The frequency of x is defined as $f_x = \sum_{i \in I_x} \Delta_i$.

If $\Delta_i > 0 \forall i$, then it is called the cash register model or insertion model. Otherwise called the turnstile model. Deterministic algorithms cannot estimate the turnstile model.

Given x we still want to compute an estimate $\hat{f}_x$ of f_x. This cannot be done deterministically using only $O(\log n + logu)$ bits, so we do it randomly.

The algorithm is as follows: Construct a zero-array C of size $k = \left\lceil \frac{3}{\epsilon^2} \right\rceil$. The size is chosen to fit with the Chebyshev bound later on. Pick 2-independent hashfunctions $h : [n] \to [k]$ and $s : [n] \to \{-1, 1\}$.

Process each element (x, Δ) in the stream by $C[h(x)] \leftarrow C[h(x)] + s(x)\Delta$.

For a query on key x, return the estimate $\hat{f}_x = s(x)C[h(x)]$.

Lemma. *For $x \in [n]$, let $\hat{f}_x = s(x)C[h(x)]$. Then $\mathbb{E}\left[\hat{f}_x\right] = f_x$. Thus $\hat{f}_x$ is an unbiased estimator for f_x.*

Proof.

$$\begin{aligned}
\hat{f}_x = s(x)C[h(x)] &= s(x) \sum_{y \in [n]} f_y s(y) B_{xy} && \text{where } B_{xy} = [h(y) = h(x)]. \\
&= f_x + \sum_{y \neq x} f_y s(x) s(y) B_{xy} && \text{as } x = y \implies f_y s(x) s(y) B_{xy} = f_x. \\
\mathbb{E}\left[\hat{f}_x\right] &= f_x + \sum_{y \neq x} f_y \mathbb{E}[s(x)s(y)B_{xy}] \\
&= f_x + \sum_{y \neq x} f_y \mathbb{E}[s(x)]\mathbb{E}[s(y)]\mathbb{E}[B_{xy}] && \text{by 2-independence.} \\
&= f_x.
\end{aligned}$$

The first equality is because

$$C[h(x)] = \sum_{y \in [n], h(y) = h(x)} s(y) f_y = \sum_{y \in [n]} f_y s(y) [h(y) = h(x)] = \sum_{y \in [n]} f_y s(y) B_{xy}.$$

□

So we have proved that the estimator is unbiased. What about the variance? For an n-dimensional vector $\mathbf{f}$ and $i \in [n]$, define $\mathbf{f}_{-i}$ to be the $(n-1)$-dimensional vector obtained by dropping index i. Then we have $||\mathbf{f}_{-x}||_2^2 = ||\mathbf{f}||_2^2 - f_x^2$. We are going to use this in the following variance proof.

Lemma.

$$\mathrm{Var}\left[\hat{f}_x\right] = \mathbb{E}\left[\left(\hat{f}_x - f_x\right)^2\right] = \frac{||\mathbf{f}_{-x}||_2^2}{k}$$

Proof.

$$\begin{aligned}
\mathbb{E}\Big[\Big(\hat{f}_x - f_x\Big)^2\Big] &= \mathbb{E}\left[\left(\sum_{y\neq x} f_y s(x)s(y)B_{xy}\right)^2\right] \\
&= \sum_{y\neq x}\sum_{z\neq x} \mathbb{E}[f_y s(x)s(y)B_{xy} f_z s(x)s(z)B_{xz}] \\
&\text{any argument where } y \neq z \text{ is 0.} \\
&= \sum_{y\neq x} f_y^2 \mathbb{E}\Big[B_x y^2\Big] + 0 \\
&= \sum_{y\neq x} f_y^2 \frac{1}{k} && \text{as } \mathbb{E}\Big[B_{xy}^2\Big] = \frac{1}{k}. \\
&= \frac{||\mathbf{f}_{-x}||_2^2}{k}
\end{aligned}$$

□

We can now use this to apply Chebyshev's inequality to get that

$$\Pr\left[|\hat{f}_x - f_x| \geq \varepsilon||\mathbf{f}_{-x}||_2\right] \leq \frac{\mathrm{Var}\Big[\hat{f}_x\Big]}{(\varepsilon||\mathbf{f}_{-x}||_2)^2} = \frac{1}{k\varepsilon^2} \leq \frac{1}{3}.$$

One can also repeat $t \geq 36 \ln \frac{1}{\delta}$ times and use the median trick to get an even better bound of $\leq \delta$ for some $\delta > 0$, which we do not have time for in this presentation.

9.4 The Median Trick

Define a random variable X to be *bad* if $|X - \mathbb{E}[X]| > \Delta$. Consider random variables $X_1, \ldots, X_t \in \mathbb{R}$ with $\mathbb{E}[X_1] = \cdots = \mathbb{E}[X_t] = \mu$. Let Y be the *median* of $X_1, \ldots, X_t$. If Y is bad, then at least $\lceil \frac{t}{2} \rceil$ of the X_i are bad. Suppose the X_i are all independent, and $\Pr[X_i \text{ bad }] \leq \frac{1}{3}$. Let $B_i = [X_i \text{ bad }]$ and $B = \sum_i B_i$. Then $\mathbb{E}[B] \leq \frac{t}{3}$, and

$$\Pr[Y \text{ bad }] \leq \Pr\left[B \geq \frac{t}{2}\right] = \Pr\left[B \geq \frac{3}{2}\frac{t}{3}\right] \leq \left(\frac{e^{\frac{1}{2}}}{\left(\frac{3}{2}\right)^{\frac{3}{2}}}\right)^{\frac{t}{3}} \leq e^{-\frac{t}{36}}.$$

This is because for $0 \leq x \leq 1$ we have

$$\frac{e^x}{(1+x)^{(1+x)}} \leq e^{-\frac{x^2}{3}}.$$

Thus for any $\delta > 0$, to get $\Pr[Y \text{ bad }] \leq \delta$, we want $e^{-\frac{t}{36}} \leq \delta$, or equivalently $t \geq 36 \ln \frac{1}{\delta}$. We could actually have gotten a slightly tighter result by just evaluating directly. In particular, we can easily reduce the constant from 36 to ≈ 27.727. However, it is nice to know simple approximations like the one above, because most of the time you do not need the tight result.