RAD

Marcus Teller

October 31, 2020

Contents

1 Chapter 1 **3**
- 1.1 Quicksort 3
  - 1.1.1 Algorithm 3
  - 1.1.2 Analysis 3
- 1.2 Min-cut 4
- 1.3 Binary planar partitions 5
  - 1.3.1 Problem 5

2 Chapter 3 **5**
- 2.1 Markovs inequality 5
- 2.2 Chebyshevs inequality 5
- 2.3 Randomized Selection 5
  - 2.3.1 Problem 5
  - 2.3.2 Analysis 6

3 Chapter 4 **8**
- 3.1 Set balancing 8
  - 3.1.1 Problem 8
  - 3.1.2 Algorithm 8
  - 3.1.3 Analysis 8
- 3.2 Chernoff bounds 10

4 Hash Tables **10**
- 4.1 Hash functions 10
  - 4.1.1 Universal Hashing 10
- 4.2 Hash table with chaining 10
  - 4.2.1 Query time 11
  - 4.2.2 two-level hashing 12

5 Distinct elements **12**
5.1 Problem 12
5.2 Algorithm 12
5.3 Analysis 13
5.4 Median trick 13
5.5 Alternate algorithm 14

6 Heavy Hitters **14**
6.1 Problem 14
6.2 Algorithm 15
6.3 Analysis 15

7 Second moment **16**
7.1 Problem 16
7.2 Algorithm 17
7.3 Analysis 17
7.3.1 Distance preserving dimensionality reduction 19
7.4 General u-universal hash 20

8 Algebraic Techniques **20**
8.1 Freivalds technique 20
8.1.1 Problem 20
8.1.2 Algorithm 20
8.1.3 Analysis 20
8.2 Verifying polynomial identities 20
8.2.1 Problem 20
8.2.2 Algorithm 21
8.2.3 Analysis 21
8.3 Swartz-Zippel theorem 21
8.3.1 Problem 21
8.3.2 Algorithm 21
8.4 Equality between bit strings 22
8.4.1 Problem 22
8.4.2 Algorithm 22
8.4.3 Analysis 22
8.5 Pattern matching 23
8.5.1 Problem 23
8.5.2 Algorithm 23
8.6 Polynomial Pattern matching 23
8.6.1 Algorithm 23
8.7 Multiset matching(not part of exam) 23
8.7.1 Problem 23
8.7.2 Algorithm 23

1 Chapter 1

1.1 Quicksort

1.1.1 Algorithm

you know

1.1.2 Analysis

$S_{(i)}$ is the element of rank i. Define $X_{ij} = [S_{(i)}, S_{(i)}$ are compared], we now see that the number of comparisons is

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

Lets now take the expectation of this value

$$\mathbb{E}[\sum_{i=1}^{n} \sum_{j>i} X_{ij}] = \sum_{i=1}^{n} \sum_{j>i} \mathbb{E}[X_{ij}]$$

Let p_{ij} be the probability that $S_{(i)}, S_{(i)}$ are compared, now draw a tree and look at the level order permutation π, we notice that two elements are only compared if one is an ancestor of another. We have two observations:

There is only a comparison between $S_{(i)}, S_{(i)}$ if any of them occur earlier in π than any $S_{(l)}$ such that $i < l < j$. Let $S_{(k)}$ be the earliest in π with rank between i and j, if $k \notin \{i, j\}$, then i and j are in the different sub trees, if k is either i or j, then they are compared.

Any of the elements $S_{(i)}, .., S_{(j)}$ are euqally likely to be the first of them to be chosen as a partitioning element, thus the probability that the first element is either $S_{(i)}, S_{(i)}$ is $2/(j - i + 1)$ We get that

$$\begin{aligned}\sum_{i=1}^{n} \sum_{j>i} p_{ij} &= \sum_{i=1}^{n} \sum_{j>i} \frac{2}{j-i+1} \\ &\leq \sum_{i=1}^{n} \sum_{k=1}^{n-i+1} \frac{2}{k} \\ &\leq 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k}\end{aligned}$$

The number of comparisons is bounded above by $2nH_n$ where H_n is the $n'th$ harmonic number. We know that $H_n \sim \ln n + \Theta(1)$, from this we get the expected running time as $O(n \log n)$

1.2 Min-cut

Consider an undirected multigraph $G = (V, E)$ where $|V| = n$ and $|E| = m$. A min-cut is a cut $C \subseteq E$, with minimum cardinality, such that G is split in two. Lets define $k = |C|$, and G_i as G at the beginning of the i'th iteration of the algorithm. $i \in \{1, 2, .., n-2\}$.

We notice that $m \geq \frac{kn}{2}$, otherwise, by the pigeonhole principle, one vertex would have less than k edges, and that would be a cut smaller than the min-cut which is a contradiction. This also means that the number of edges in G_i is at least $\frac{k(n-i+1)}{2}$.

Let ε_i be the event that an edge of C is NOT picked at the i'th step. The probability that an edge randomly chosen in the first step IS in C, is $\leq \frac{k}{kn/2} = \frac{2}{n}$, this means that $\Pr[\varepsilon_1] \geq 1 - \frac{2}{n}$. It is now also clear that for the algorithm to return a correct answer, it has to return any min-cut, since we are looking at a specific cut, we have

$$\Pr[\text{success}] \geq \Pr[\cap_{i=1}^{n-2} \varepsilon_i] = \Pr[\varepsilon_1] \cdot \Pr[\varepsilon_2 \mid \varepsilon_2] \cdot \ldots \cdot \Pr[\varepsilon_{n-2} \mid \cap_{j=1}^{n-3} \varepsilon_j] \tag{1}$$

We know that at iteration i, the mincut size is $\geq k$, since contractions cannot reduce the min-cut size, this means that

$$\Pr[\varepsilon_i \mid \cup_{j=1}^{i-1} \varepsilon_j] \geq 1 - \frac{2}{n-i+1} \tag{2}$$

Now by combining (1) and (2) We get

$$\begin{aligned}
\Pr[\cap_{i=1}^{n-2} \varepsilon_i] &\geq \prod_{i=1}^{n-1} \left(1 - \frac{2}{n-i+1}\right) \\
&= \prod_{i=1}^{n-1} \left(\frac{n-i-1}{n-i+1}\right) \\
&= \frac{1 \cdot 2 \cdots (n-2)}{3 \cdot 4 \cdots n} = \frac{2}{(n-1)n} \geq \frac{2}{n^2}
\end{aligned}$$

This is quite terrible for larger graphs, so lets now run the algorithm t times, and return the smallest cut. Now we have

$$\Pr[Err] \leq \left(1 - \frac{2}{n^2}\right)^t$$

Now by using $1 + x \leq e^x$ where $x = -\frac{2}{n^2}$ we get

$$\Pr[Err] \leq \left(1 - \frac{2}{n^2}\right)^t \leq e^{-\frac{2t}{n^2}}$$

By setting $t = \frac{n^2}{2}$ we get $\Pr[Err] \leq \frac{1}{e}$

1.3 Binary planar partitions

1.3.1 Problem

We are given a bunch of line segments, and we wish to draw them in the correct order, given a viewpoint.

2 Chapter 3

2.1 Markovs inequality

Holds for any positive random variable X and any $a > 0$

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$$

Alternate formulation:

$$\Pr[X \geq a\mathbb{E}[X]] \leq \frac{1}{a}$$

Proof:

$$\mathbb{E}[X] = \sum_{x} x\Pr[X = x] \geq \sum_{x \geq a} x\Pr[X = x] \geq \sum_{x \geq a} a\Pr[X = x] = a\sum_{x \geq a}\Pr[X = x] = a\Pr[X \geq a]$$

Fairly weak bound, can be improved with chebyshevs inequality.

2.2 Chebyshevs inequality

Lets start by defining $\mu_X = \mathbb{E}[X]$ and $\sigma_X^2 = \mathrm{Var}[X] = \mathbb{E}[(X - \mu_X)^2]$.

Given a random variable X where $\sigma_X > 0$ and a $t > 0$ it holds that

$$\Pr[\mid X - \sigma_X \mid \geq t\sigma_X] \leq \frac{1}{t^2}$$

Proof:

$$\Pr[\mid X - \sigma_X \mid \geq t\sigma_X] = \Pr[(X - \mu_X)^2 \geq t^2\sigma_X^2] \overset{\text{markov}}{\leq} \frac{1}{t^2}$$

Since $\mathbb{E}[(X - \mu_X)^2] = \sigma_X^2$

2.3 Randomized Selection

2.3.1 Problem

Given n numbers, return the $i'th$ largest number(could be median in case of $i = n/2$)
Any deterministic algorithm needs a minimum of $2n$ comparisons, randomized selection needs only $\frac{3}{2}n + O(n)$ comparisons

Algorithm:
Input: List S with n elements and $k \in \{1, .., n\}$

1) Pick $n^{3/4}$ elements from S independently and with replacement, let this new multiset be R
2) Sort R
3) let $x = kn^{-1/4}$, $l = \max\{\lfloor x - \sqrt{n} \rfloor, 1\}$, $h = \min\{\lceil x + \sqrt{n} \rceil, n^{3/4}\}$
We can think of l and h to be lower and upper bound of our search space, and x as representing the approximate index of k'th lowest element. We now also set $a = R_{(l)}$ and $b = R_{(h)}$, ie. we pick a as the l'th smallest element in our sampled list, and b as the h'th smallest in our sampled list. We now also introduce the notation $r_L(a)$ which represents the rank of a in L. Now we find $r_S(a)$ and $r_S(b)$, that is the rank of a and b, in the underlying list S both of these operations require $n - 1$ comparisons.
4) There are different cases but this one is interesting.
Assume $k \in [n^{1/4}, n - n^{1/4}]$, and let $P = \{y \in S \mid a \leq y \leq b\}$. We now need to check if $S_{(k)} \in P$ or $|P| > 4n^{3/4} + 2$, which can be done by evaluating $r_S(a) \leq k \leq r_S(b)$, the size of P can be found by evaluating $r_S(b) - r_S(a) + 1$. If the test suceeds go to 5)
5) Sort P and return $P_{(k - r_S(a)+1)} = S_{(k)}$. the intuition here is that if $k = r_S(a)$, we will simply return the first element in P, otherwise we simply offset it correctly.

2.3.2 Analysis

Theorem:
With probability $1 - O(n^{-1/4})$ line 4 succeds in first iteration, and the algorithm performs $2n + O(n)$ comparisons.

Proof:
What can go wrong?

1. $S_{(k)} < a$
2. $S_{(k)} > b$
3. $|P| > 4n^{3/4} + 2$

Lets focus on 1.
For $i = 1..n^{3/4}$. let $X_i = [R_i \leq S_{(k)}]$ and $X = \sum_i X_i$, we see that X is the number of samples less than or equal $S_{(k)}$.

Lets look at the different kinds of errors:

$$\begin{aligned}\Pr[\text{error 1.}] &= \Pr[X < l] \\ &= \Pr[X < \max\{\lfloor x - \sqrt{n} \rfloor, 1\}]\end{aligned}$$

The intuition here is that if there is less than l elements before $S_{(k)}$, then it is not "pushed" far enough along in the array, to be part of P, and we have to run another iteration.

If we now look at the max, we see that the only way we ever get a 1, is if $\lfloor x - \sqrt{n} \rfloor = 0$, and if that is the case, X has to be 0, this means that we can swap the $<$ for a $\leq$, and remove the max. This does however also mean that the condition is easier to fulfill, thus we get

$$\begin{aligned}
\Pr[\text{error 1.}] &= \Pr[X < l] \\
&= \Pr[X < \max\{\lfloor x - \sqrt{n} \rfloor, 1\}] \\
&\leq \Pr[X \leq \lfloor x - \sqrt{n} \rfloor] \\
&\leq \Pr[X \leq x - \sqrt{n}]
\end{aligned}$$

We want to use chebyshev, so, first we will find

$$\mu_X = \mathbb{E}\left[\sum_i X_i\right] = \sum_i \mathbb{E}\left[X_i\right] = \sum_i \frac{k}{n} = \frac{n^{3/4}k}{n} = n^{-1/4}k = x$$

Lemma:
Given <u>independent</u> random variable $Y_1, .., Y_m$, let $Y = \sum_i^m Y_i$, then $\sigma_Y^2 = \sum_i^m \sigma_{Y_i}^2$

Using this, lets now try and find σ_X^2, using it we get

$$\sigma_X^2 = \sum_i^{n^{3/4}} \sigma_{X_i}^2 = \sum_i^{n^{3/4}} \frac{k}{n}(1 - \frac{k}{n})$$

This comes from the fact that they are bernoulli distributed, and the distribution has variance $p(1-p)$, since $p(1-p) \leq \frac{1}{4}$, we get

$$\sigma_X^2 = \sum_i^{n^{3/4}} \sigma_{X_i}^2 = \sum_i^{n^{3/4}} \frac{k}{n}(1 - \frac{k}{n}) \leq \sum_i^{n^{3/4}} \frac{1}{4} = \frac{n^{3/4}}{4}$$

To get the standard deviation we now take the square root:

$$\sigma_X = \sqrt{\frac{n^{3/4}}{4}} = \frac{n^{3/8}}{2}$$

Lets return to an earlier statement

$$\Pr[\text{error 1.}] \leq \Pr[X \leq x - \sqrt{n}]$$

Since we found out that $\mu_X = x$ we can insert

$$\begin{aligned}
\Pr[\text{error 1.}] &\leq \Pr[X \leq x - \sqrt{n}] \\
&= \Pr[X \leq \mu_X - \sqrt{n}] \\
&= \Pr[\mu_X - X \geq \sqrt{n}] \\
&\leq \Pr[\mid \mu_X - X \mid \geq \sqrt{n}] \\
&= \Pr[\mid \mu_X - X \mid \geq 2n^{1/8} \frac{n^{3/8}}{2}] \\
&\leq \frac{1}{4n^{1/4}} \text{ in the case we get equal} \\
&= O(n^{-1/4})
\end{aligned}$$

3 Chapter 4

3.1 Set balancing

3.1.1 Problem

Given a $m \times n$ 0/1 matrix, where m is the number of features, and n is the size of the population, our goal is to find a vector $\bar{b} \in \{-1, 1\}^n$ such that $||A\bar{b}||_\infty$ is as small as possible. This can be thought of as deviding a group of n people with m binary features into two balanced groups.

$$\underbrace{\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}}_{\bar{b}} = \underbrace{\begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}}_{A\bar{b}} \Rightarrow ||A\bar{b}||_\infty = 2$$

3.1.2 Algorithm

Set $\bar{b}_i = 1$ with probability 1/2, choices are independent.

3.1.3 Analysis

Consider a row A_j in A, assume A_j takes on the form $[\underbrace{1, .., 1}_{k}, \underbrace{0, .., 0}_{n-k}]$, this can be done by reordering the columns.

For $i = 1..k$ let $X_i = \left\{ \begin{array}{ll} 1 & b_i = -1 \\ 0 & b_i = 1 \end{array} \right.$. Let $X = \sum_{i=1}^{k} X_i$. $\mu = \mu_X = \frac{k}{2}$, since X_i is 1 with probability 1/2.

Lets now look at

$$\Pr[A_j\bar{b} = 0] = \Pr[X = \frac{k}{2}]$$
$$\Pr[A_j\bar{b} = 2] = \Pr[X = \frac{k}{2} - 1]$$

We see that in general for all $c \in \mathbb{N}_0$

$$\Pr[A_j\bar{b} = 2c] = \Pr[X = \frac{k}{2} - c] \iff \Pr[A_j\bar{b} > 2c] = \Pr[X < \frac{k}{2} - c] \quad (3)$$

+We want to use a chernoff bound on the form

$$\Pr[X < (1-\delta)\mu] < e^{-\frac{\delta^2\mu}{2}} = \frac{1}{m^2} \quad (4)$$

We can solve this for δ^2 and get $\delta^2 = \frac{4\ln m}{\mu} = \frac{8\ln m}{k} \Rightarrow \delta = 2\sqrt{\frac{2\ln m}{k}}$ This is now equivalent of putting equation (4) on the following form

$$\Pr[X < \frac{k}{2} - \delta\mu] < \frac{1}{m^2}$$

We can now use equation (3) and get

$$\begin{aligned}\frac{1}{m^2} &> \Pr[A_j\bar{b} > 2\delta\mu] \\ &= \Pr[A_j\bar{b} > k\delta] \\ &= \Pr[A_j\bar{b} > \sqrt{2k\ln m}] \\ &\geq \Pr[A_j\bar{b} > \sqrt{2n\ln m}]\end{aligned}$$

Now by symmetry if we reverse X, such that it now counts the 1' s instead, we get

$$\Pr[A_j\bar{b} < -2\sqrt{2n\ln m}] < \frac{1}{m^2}$$

Using a union bound we get

$$\Pr[|A_j\bar{b}| > 2\sqrt{2n\ln m}] < \frac{2}{m^2}$$

Using another union bound over all the rows, we get

$$\begin{aligned}\Pr[||A_j\bar{b}||_\infty > 2\sqrt{2n\ln m}] &= \Pr[\bigcup_{j=1}^{n}\left\{|A_j\bar{b}| > 2\sqrt{2n\ln m}\right\}] \\ &\leq m\frac{2}{m^2} = \frac{2}{m}\end{aligned}$$

Now the positive event: With probability $\geq 1 - \frac{2}{m}$ we have

$$||A\bar{b}||_\infty \leq 2\sqrt{2n\ln m}$$

3.2 Chernoff bounds

Tail bound using moment generating functions.

$$\Pr[X > (1+\delta)\mu] < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu} \tag{5}$$

$$\Pr[X < (1+\delta)\mu] < e^{-\delta^2\mu/2} \tag{6}$$

BONUS:

> which follow from the inequality $\frac{2\delta}{2+\delta} \leq \log(1+\delta)$ from the list of logarithmic inequalities. Or looser still:
>
> $$\Pr(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta^2\mu}{3}}, \qquad 0 \leq \delta \leq 1,$$
> $$\Pr(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta\mu}{3}}, \qquad 1 \leq \delta,$$

Figure 1: Practical chernoff bounds

4 Hash Tables

4.1 Hash functions

A hash function is a function $h : U \to [m]$. In the ideal world, all $|U|$ hash values independent and uniformly distributed. The only way of doing this takes up $|U|$ space, so we need to suffice with pseudo-randomness.

Choose a random prime p such that $U \subseteq [p]$, ie. $[p]$ must be atleast as large as U. We also need to pick $a, b \in [p]$, now lets define $h_{a,b}(x) = (ax + b) \mod p$ and $h^m_{a,b}(x) = h_{a,b} \mod m$.

4.1.1 Universal Hashing

Consider the hash function $h : U \to [m]$. For $x, y \in U$, $x \neq y$, $\Pr_{\mathrm{h}}[h(x) = h(x)] \leq 1/m$

Thm. If $a \in [p]_+ = \{1, ..p-1\}$, $b \in [p]$ are uniforme and independent, then $h_{a,b}(x)$ is universal

Thm. $U = [2^w]$, w=8,17,32,64, $m = 2^l$, and $a =\in [2^w]$ is a random odd number. Then we have

$$h_a(x) = (a \cdot x) >> (w - l)$$

4.2 Hash table with chaining

Or task is to store $S \subseteq U$, $|S| \leq n$ in $O(n)$ space, with $O(1)$ expected query time, and $O(1)$ expected insertion time.

Use universal $h : U \to [m]$, $m \geq n$, we now have an array L of lists. We will use it such that $L[i] = \{x \in S \mid h(x) = i\}$. We see that $x \in S \iff x \in L[h(x)]$. With this scheme we only need to remember $m, a, b, *L$[1]. We then use $O(n+m)$ space, m for the array and n for the linked lists. Time complexity $O(1+|S_{h(x)}|)$, we could ask for the expected size $\mathbb{E}[|S_{h(x)}|]$.

Assume $x \notin S$. Lets start by remembering

$$|S_{h(x)}| = \sum_{y \in S} [h(y) = h(x)] \tag{7}$$

Now we can use linearity of expectation to get

$$\begin{aligned}
\mathbb{E}[|S_{h(x)}|] &= \sum_{y \in S} \mathbb{E}[h(y) = h(x)] \\
&= \sum_{y \in S} \Pr[h(y) = h(x)] \\
&\leq \sum_{y \in S} \frac{1}{m} = \frac{n}{m} \overset{m \geq n}{\leq} 1
\end{aligned}$$

4.2.1 Query time

Given a set S, we want constant query time, can be achieved if $|S_i| \leq 1$ for all i if and only if we have 0 collisions, ie. $x \neq y$, $x, y \in S$, $h(x) \neq h(y)$.

$$C_h \triangleq \#\text{ collisions } x \neq y, x, y \in S, h(x) = h(y) \tag{8}$$

lets now look at $\mathbb{E}[C_h]$, we can look at all pairs,

$$\mathbb{E}[C_h] \leq \binom{n}{2} \frac{1}{m} = \frac{n(n-1)}{2m} \tag{9}$$

Lets now look at

$$\Pr[C_h \geq \frac{n(n-1)}{m}] \overset{\text{markov}}{\leq} \frac{\mathbb{E}[C_h]}{n(n-1)/m} = \frac{1}{2} \tag{10}$$

If we are unfortunate enough and $C_h \geq \frac{n(n-1)}{m}$, we simply choose a new hash function. We expect 2 attempts since its a random bernoulli distributed variable.

If we set $m = n^2$, we have $C_h < \frac{n(n-1)}{m} = 1 - \frac{1}{n} < 1 \Rightarrow C_h = 0$

[1] Pointer to L

4.2.2 two-level hashing

Let $m = n$, and we get $C_h < \frac{n(n-1)}{m} = n - 1$. For every i, save S_i with $m_i = n_i^2$ where $n_i = |S_i|$. Ie, for each S_i, we have another hash function. Space is

$$O(1 + n + m + \sum_{i \in [m]} (1 + n_i + m_i)) = O(n + \sum_{i \in [m]} (1 + n_i + n_i^2))$$

We get $1 + n + m$ from saving the standard hash table, and then $\sum_{i \in m}(1 + n_i + m_i)$. Now, the 1 in the sum will simply turn into n and disappear, we will rewrite $n_i + n_i^2 = 2n_i + n_i^2 - n_i = 2n_i + n_i(n_i - 1)$ obtaining

$$O(n + \sum_{i \in [m]} (n_i + n_i(n_i - 1)))$$

Since $\sum n_i = n$ and $\binom{n}{2} = \frac{n(n-1)}{2}$ we can rewrite to

$$O(n + 2\sum_{i \in [m]} \binom{n_i}{2})) = O(n + C_h) = O(n)$$

The $\binom{n_i}{2}$ comes from the fact that if 2 values end up in the same bucket, they have collieded, so summing over this gives us the total number of collisions. And remember that $C_h < n - 1$.

5 Distinct elements

5.1 Problem

Given a stream $j_1, .., j_s \in [n]$, we wish to find $f_j = \#\{i | j_i = j\}$. We now wish to find $S = \{j \mid f_j > 0\}$, and return an estimate of $d = |S|$, in other words we wish to find the number of distinct elements in the stream.

For this problem, we will need a strong universal(two universal) $h : [n] \to [2^l]$, $2^l > d^2$. ie. $i \neq j \Rightarrow (h(i), h(j))$ uniform in $[2^l]^2 \iff h(i)$ and $h(j)$ are independent and uniform in $[2^l]$. We now define a function $zeros(y) = \#\text{trailing zeros} = \max\{q, 2^q \mid y\}$.

5.2 Algorithm

Algorithm 1 Estimate d from a stream of data

$z = 0$
Process(j): $z = \max\{z, zeros(h(j))\}$
return $\hat{d} = 2^{z+1/2}$

5.3 Analysis

Lets look at a $j \in S$, and lets say $X_{r,j} = [zeros(h(j)) \geq r]$, $Y_r = \sum_{j \in S} X_{r,j}$, it is now clear that $z \geq r \iff Y_r > 0$. If Y_r was 0, there would have been no $h(j)$ with at least r trailing zeros.

We can quickly deduce that $\mathbb{E}[X_{r,j}] = \Pr[zeros(h(j)) \geq r] = 2^{-r}$. This means that

$$\mathbb{E}[Y_r] = \sum_{j \in S} \mathbb{E}[X_{r,j}] = d/2^r \tag{11}$$

We can now also look at the variance

$$\operatorname{Var}[X_{r,j}] \leq \mathbb{E}[X_{r,j}^2] \leq \mathbb{E}[X_{r,j}] = 1/2^r \tag{12}$$

$$\operatorname{Var}[Y_r] \overset{\text{pairwise indep.}}{=} \sum_{j \in S} \operatorname{Var}[X_{r,j}] \leq d/2^r \tag{13}$$

When our algorithm returns, we are given the estimate $\hat{d} = 2^{z+1/2}$. Let a be the smallest integer such that $2^{a+1/2} > 6d$. This represents the smalles value of z where we could guess 6 times too high. Lets explore this.

$$\Pr[\hat{d} \geq 6d] = \Pr[z \geq a] = \Pr[Y_a > 0] = \Pr[Y_a \geq 1] \leq \frac{\mathbb{E}[Y_a]}{1} = d/2^a \tag{14}$$

We can now use the fact that

$$2^{a+1/2} > 6d \iff 2^a\sqrt{2}/6 > d$$

to get

$$d/2^a < \sqrt{2}/6 < 1/4 \tag{15}$$

Lets now bound our guess from below, using a very symmetrical argument. Let b be the largest integer such that $2^{b+1/2} \leq d/6$.

$$\begin{aligned}
\Pr[\hat{d} \leq d/6] &= \Pr[z \leq b] \\
&= \Pr[Y_{b+1} = 0] \\
&= \Pr[Y_{b+1} - \mathbb{E}[Y_{b+1}] = -\mathbb{E}[Y_{b+1}]] \\
&\leq \Pr[|Y_{b+1} - \mathbb{E}[Y_{b+1}]| \geq \mathbb{E}[Y_{b+1}]] \\
&\leq \frac{\operatorname{Var}[Y_{b+1}]}{\mathbb{E}[Y_{b+1}]^2} \leq \frac{1}{\mathbb{E}[Y_{b+1}]} = \frac{d}{2^{b+1}} \leq 1/4
\end{aligned}$$

5.4 Median trick

Make k independent estimates $\hat{d}_0, .., \hat{d}_{k-1}$ (using k different hash functions), sort them and return the median. let this be $\hat{d}_{(\lceil k/2 \rceil)}$. Lets look at the probability of getting it right now, let $t = 6d$, we need to look at $\Pr[\hat{d}_{(\lceil k/2 \rceil)} > t]$. We will consider

$$Z_i = [\hat{d}_i \geq t]$$

$$Z = \sum_{i \in [k]} Z_i$$

We now get the following important equation

$$\Pr[\hat{d}_{(\lceil k/2 \rceil)} > t] = \Pr[Z > \lceil k/2 \rceil] \tag{16}$$

This essentially says, if more than half of the estimates are larger than t, then that is the only situation where the median also is.

$$\mu = \mathbb{E}[Z] = \sum_{i \in [k]} \Pr[Z_i] \leq \frac{k}{4} \tag{17}$$

What we are now asking is

$$\Pr[Z \geq 2\mu] \overset{\text{chernoff(wtf)}}{\leq} \exp(-\mu/3) = \exp(-k/12) \tag{18}$$

5.5 Alternate algorithm

We want to use $O(k)$ space, we will work with some hashtable B where $|B| \leq k$.

Algorithm 2 Estimate d from a stream of data with $O(k)$ counters

$z = 0$
$B = \varnothing$
Process(j):
if $zeros(h(j)) \geq z$ and $j \notin B$ **then**
    $B = B \cup \{j\}$
    while $|B| > k$ **do**
        remove all $i \in B$ where $zeros(h(i)) = z$
        z+=1
    end while
end if
return $\hat{d} = |B|2^z$

Here, B is the set of j seen so far with atleast z zeroes.

6 Heavy Hitters

6.1 Problem

This time our stream is slightly more complex, its is now a stream of key value pairs

$$(j_0, \Delta_0), (j_1, \Delta_1), ..., (j_{s-1}, \Delta_{s-1}) \in [n] \times \mathbb{Z}$$

Now we define

$$f_j \triangleq \sum_{\substack{i \in [s] \\ j_i = j}} \Delta_i \tag{19}$$

We want to keep track of all f_j using k counters in an array $C[k]$. We need two 2-universal(strong universal) hash functions $h : [n] \to [k]$, and $s : [n] \to \{-1, 1\}$, note that the two hash functions should be independent.

6.2 Algorithm

C is an array, and we support Query at any time, Process is called at each datapoint in the stream.

Algorithm 3 Heavy hitters

$C = 0^k$
Process(j, Δ): $C[h(j)] \leftarrow C[h(j)] + s(j) \cdot \Delta$
Query(q): **return** $\hat{f}_q = s(q) \cdot C[h(q)]$

6.3 Analysis

For analysis lets look at $X = \hat{f}_q$. Lets start with a basic definition.

$$H_j \triangleq [h(j) = h(q)]$$

Firstly we observe that

$$\begin{aligned} X &= s(q) \sum_{j \in [n]} s(j) f_j H_j \\ &= s(q)s(q) f_q H_q + \sum_{\substack{j \neq q \\ j \in [n]}} s(j)s(q) f_j H_j \\ &= f_q + \overbrace{\sum_{\substack{j \neq q \\ j \in [n]}} s(j)s(q) f_j H_j}^{\text{Error term}} \end{aligned} \tag{20}$$

Lets now look at the expected value of the error term, lets remember that $s(j)$ and $s(q)$ are independent because of strong universality.

$$\sum_{i \neq q} \mathbb{E}[s(j)s(q) f_j H_j] = \sum_{i \neq q} \underbrace{\mathbb{E}[s(j)]}_{0} \mathbb{E}[s(q)] f_j \mathbb{E}[H_j] = 0$$

Because of this we get

$$\mathbb{E}[X] = f_q \tag{21}$$

Lets now look at the variance of X

$$
\begin{aligned}
\mathrm{Var}[X] &= \mathbb{E}[(X - f_q)^2] \\
&= \mathbb{E}[(\sum_{j\neq q} s(j)s(q)f_jH_j)^2] \\
&= \sum_{\substack{i,j\in[n]\\ i\neq q\neq j}} \mathbb{E}[\underbrace{s(i)s(j)(s(q))^2 f_i f_j H_i H_j}_{(s(i)s(q)f_iH_i)(s(j)s(q)f_jH_j)}] \\
&= \sum_{\substack{i,j\in[n]\\ i\neq q\neq j}} \mathbb{E}[s(i)s(j)f_if_jH_iH_j] \\
&= \sum_{\substack{i,j\in[n]\\ i\neq q\neq j}} \begin{cases} f_i^2 \overbrace{\mathbb{E}[H_i]}^{1/k} & i=j \\ 0 & i\neq j \end{cases} \\
&= \sum_{j\neq q} f_j^2/k = ||f_{-q}||_2^2/k
\end{aligned}
$$

Now lets look at the probability of deviating from its mean.

$$
\Pr[|\hat{f}_q - f_q|] > \varepsilon ||f_{-q}||_2] \leq \frac{\mathrm{Var}[\hat{f}_q]}{\varepsilon^2 ||f_{-q}||_2^2} = \frac{1}{k\varepsilon^2} \tag{22}
$$

Now if $k \geq 4/\varepsilon^2$ we get $1/k\varepsilon^2 \leq 1/4$.

Now we will apply the median trick. We pick t independent X_i, and return $X_{(\lceil t/2 \rceil)}$, we say X_i is bad if $|X_i - \mathbb{E}[X]| > Q$, $B_i = [|X_i - \mathbb{E}[X]| > Q]$. Lets now define

$$
B \triangleq \sum_{i\in[t]} B_i
$$

We see that $X_{(\lceil t/2 \rceil)}$ Bad $\iff$ $B > t/2$, $\mathbb{E}[B] \leq t/4 = \mu$, now

$$
\Pr[X \text{ bad}] = \Pr[B \geq 2\mu] \leq \exp(-\overbrace{\delta^2}^{1}\mu/3) = \exp(-t/12)
$$

7 Second moment

7.1 Problem

We are again given a stream of key value pairs

$$
(j_0, \Delta_0), (j_1, \Delta_1), ..., (j_{s-1}, \Delta_{s-1}) \in [n] \times \mathbb{Z}
$$

We define a total value for $j \in [n]$,

$$
f_j \triangleq \sum_{\substack{i\in[s]\\ j_i = j}} \Delta_i \tag{23}
$$

We can imagine f as an n dimensional vector, $f \in \mathbb{Z}^n$. We now define the m'th moment as

$$F_m(f) \triangleq \sum_{i \in [n]} f_i^m = \|f\|_m^m \tag{24}$$

Our goal is now to estimate $F_2(f)$ and therefore $||f||_2 = \sqrt{F_2(f)}$, we will do this by using a sketch of k counters.

For this, we will need a 4-universal hash function. Generally we have u-universal $h : U \to R$. For distinct $x_0, .., x_{u-1}$, the vector $(h(x_0), .., h(x_{u-1}))$ is uniform in R^u. Equivalently $h(x_0), .., h(x_{u-1})$ are all independent and uniform in R.

7.2 Algorithm

We will use a count sketch vector $C \in \mathbb{Z}^k$, use a 4-universal $h : [n] \to [k]$, $s[n] \to \{-1, 1\}$.

Algorithm 4 Second moment estimation

$C = 0^k$
Process(j, Δ): $C[h(j)] \leftarrow C[h(j)] + s(j) \cdot \Delta$
return C

7.3 Analysis

We will start with the following lemma

Lemma 7.1

$$\forall b \in [k] : C[b] = \sum_{i \in [s]} s(j_i)\Delta_i[h(j_i) = b] = \sum_{j \in [h]} s(j) f_j [h(j) = b] \tag{25}$$

We want to estimate $F_2(f)$ by $X = F_2(C) = \sum_{b \in [k]} C[b]^2$.
We will show that $\mathbb{E}[X] = F_2$ and $\mathrm{Var}[X] < 2F_2^2/k$ and by using chebyshev that

$$\Pr[|X - F_2| > \varepsilon F_2] \leq \frac{\mathrm{Var}[X]}{\varepsilon^2 F_2^2} < \frac{2}{k\varepsilon^2} \leq 1/4$$

by setting $k \geq 8/\varepsilon^2$. We will use the notation $s_j = s(j)$ and $h_j = h(j)$ To repeat, we have that

$$
\begin{aligned}
X &= \sum_{b\in[k]} C[b]^2 = \sum_{b\in[k]} (\sum_{j\in[n]} s_j f_j [h_j = b])^2 \\
&= \sum_{b\in[k]} \sum_{i,j\in[n]} s_i s_j f_i f_j [h_j = h_i = b] \\
&= \sum_{i,j\in[n]} s_i s_j f_i f_j [h_j = h_i] \\
&= \sum_{i=j\in[n]} f_i^2 + \overbrace{\sum_{i\neq j\in[n]} s_i s_j f_i f_j [h_j = h_i]}^{Y}
\end{aligned}
$$

We now want to show that $\mathbb{E}[Y] = 0 \iff \mathbb{E}[X] = F_2$. Lets look at any term in the sum where $i \neq j$

$$\mathbb{E}[s_i s_j f_i f_j [h_i = h_j]] = \mathbb{E}[s_i]\mathbb{E}[s_j]\mathbb{E}[...] = 0$$

We now want to bound the variance $\mathrm{Var}[X] = \mathrm{Var}[Y] = \mathbb{E}[Y^2]$.

$$
\begin{aligned}
\mathbb{E}[Y^2] &= \mathbb{E}[(\sum_{i\neq j\in[n]} s_i s_j f_i f_j [h_j = h_i])^2] \\
&= \sum_{i\neq j, i'\neq j'\in[n]} \mathbb{E}[(s_i s_j f_i f_j [h_i = h_j])(s_{i'} s_{j'} f_{i'} f_{j'} [h_{i'} = h_{j'}])]
\end{aligned}
$$

Consider a term where i is unique, ie. $i \notin \{j, i', j'\}$

$$\mathbb{E}[(s_i ...)] = \mathbb{E}[s_i]\mathbb{E}[...] = 0$$

So we need to have $j \neq i$ and $k \neq l$, however none of them can be unique, this means that we need to have $(i,j) = (i',j')$ or $(i,j) = (j',i')$. Lets look at the case where $(i,j) = (i',j')$. We now only need to use 2 indicies, since $i = i'$ and $j = j'$ we can multiply by 2 for symmetry

$$
\begin{aligned}
\mathrm{Var}[X] &= 2\sum_{i\neq j\in[n]} \mathbb{E}[(s_i f_i s_j f_j [h_i = h_j])^2] \\
&= 2\sum_{i\neq j\in[n]} \mathbb{E}[f_i^2 f_j^2 [h_i = h_j]] \\
&= 2\sum_{i\neq j\in[n]} f_i^2 f_j^2 \mathbb{E}[[h_i = h_j]] \\
&= 2\sum_{i\neq j\in[n]} f_i^2 f_j^2 / k \\
&< \frac{2}{k} \sum_{i,j\in[n]} f_i^2 f_j^2 \\
&= \frac{2}{k} F_2^2
\end{aligned}
$$

The last step is because we run over 2 indicies, it is equivalent of raising F_2 to the power of 2.

7.3.1 Distance preserving dimensionality reduction

We can view the count sketch as a dimensionality reducing linear map $C^{h,s}$: $\mathbb{Z}^n \to \mathbb{Z}^k$. We can see that its linear because

$$C^{h,s}(f)_b = \sum_{j \in [n]} s(j)[h(j) = b] f_j$$

Because it is linear, we can write it as a matrix-vector product.

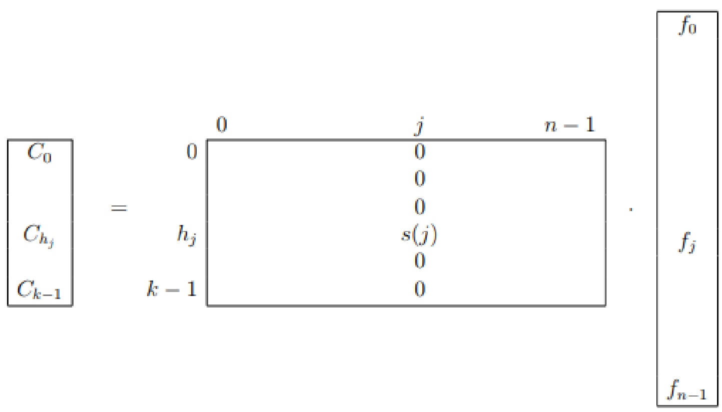

Figure 2: Linear map

In the matrix, we see that each row corresponds to a vector-vector product with the f vector and all the row in the matrix will have $s(j)$ for all j where they hash to the row index. Notice that the matrix is implicit, we never actually calculate it. So what does this mean? If we have two different vectors, f, g(extremely sparse)$\in \mathbb{Z}^n$, $C^{h,s}(f+g) = C^{h,s}(f)+C^{h,s}(g)$, equivalently $C^{h,s}(\alpha f) = \alpha C^{h,s}(f)$. We can imagine our algorithm as continuously adding the sketch of a vector with 0's everywhere except for index $h(j)$, in which place it has Δ. We want to show that $F_2(C^{h,s}(f)) \approx F_2(f)$. We can formalize this as

$$F_2(f-g) \approx F_2(C^{h,s}(f-g)) = F_2(C^{h,s}(f) - C^{h,s}(g)) \Rightarrow$$
$$||f-g||_2 \approx ||C^{h,s}(f-g)||_2 \ = ||C^{h,s}(f) - C^{h,s}(g)||_2$$

This is super amazing, since it essentially says that our count sketch is almost distance preserving.

7.4 General u-universal hash

For distinct elements $x_0, .., x_{u-1}$, the vector is uniform in R^u if

$$h(x) = (\sum_{i \in [u]} a_i x^i) \mod p$$

8 Algebraic Techniques

8.1 Freivalds technique

8.1.1 Problem

Lets look at $n \times n$ matricies A, B and our goal is to calculate $C = AB$, this can be done in $O(n^{2.3\cdots})$ time, and is extremely complicated. Our algorithm verifies this product in $O(n^2)$ time.

8.1.2 Algorithm

Generate a random vector $r \in \{0,1\}^n$, return true if $Cr = A(Br)$. The only way this algorithm errors, is if $AB \neq C$ but $A(Br) = Cr$, so lets look at that situation.

8.1.3 Analysis

Lets let $D = C - AB \neq 0$(since this was a precondition for error) but $Dr = 0$. This must mean that $\exists i, j : D_{ij} \neq 0$ and that $(Dr)_i = \sum_{k \in [n]} D_{ik} r_k = 0$. We now fix all r_k where $k \neq j$, we now have

$$D_{ij} r_j + \overbrace{\sum_{\substack{k \in [n] \\ k \neq j}} D_{ik} r_k}^{\text{constant}} = 0$$

The reason it is constant, is because we fixed all r_k where $k \neq j$. Now, if the above is true where $r_j = 0$, then it has to be false when $r_j = 1$, since we assumed $D_{ij} \neq 0$, this means that

$$\Pr[D_r = 0] \leq \Pr[(Dr)_i = 0] \leq 1/2$$

8.2 Verifying polynomial identities

8.2.1 Problem

We have some degree $\leq d$ polynomials $P(x), Q(x)$ defined over a field $\mathbb{F}[x]$, we now wish to answer the question is $Q = P$?

8.2.2 Algorithm

The idea again is to pick $S \subseteq \mathbb{F}$, and pick $r \in S$ uniformly at random, and output true if $P(r) = Q(r)$, we have a false positive in the case where $P \neq Q$ but $P(r) = Q(r)$. We will prove that if $P \neq Q$, then

$$\Pr[P(r) = Q(r)] = \frac{d}{|S|} \tag{26}$$

8.2.3 Analysis

We could have written this as $R = P - Q$ and $R = 0$, now just like before we examine the false positive situation. $R \neq 0$, but $R(r) = 0$, we will prove that $\Pr[R(r) = 0] \leq d/|S|$.

From the fundamental theorem of algebra, we know that R has $\leq d$ solutions in $\mathbb{S}$, for this reason, we can only pick a maximum of d bad values out from S, giving is $d/|S|$

8.3 Swartz-Zippel theorem

8.3.1 Problem

We are now looking at a multivariate polynomial $Q(x_1, .., x_n) \in \mathbb{F}[x_1, .., x_n]$, with total degree $\leq d$, this means if we have $\alpha x_1^{d_1} x_2^{d_2} .. x_n^{d_n}$, then the degree is $\sum_{i \in [n]} d_i$.

8.3.2 Algorithm

We now pick a random vector $(r_1, .., r_n) \in S^n$ uniformly at random, we then evaluate $Q(r_1, .., r_n) = 0$, we will show that

$$\Pr[Q(r_1, .., r_n) = 0] \leq \frac{d}{|S|}$$

Let k be the largest degree of x_1 in Q, we now define

$$Q_i(x_2, .., x_n) = \sum_{i=0}^{k} x_1^i Q_i(x_2, ..., x_n) \tag{27}$$

Lets start by picking $r_2, .., r_n$, we now ask

$$\Pr[Q_k(r_2, .., r_n) = 0] \leq \frac{d-k}{|S|}$$

Since Q_k represents all the terms multiplied by x_1^k, and we know that the sum of the degrees are a max of d, the the degree of $Q_k \leq d - k$.

Lets now fix $r_2, .., r_n$, and now assume $Q_k(r_2, .., r_n) \neq 0$ we now define

$$q(x_1) \triangleq Q(x_1, r_2, .., r_n)$$

We know that q has degree k, and that $g \neq 0$, since we assumed $Q_k(r_2, .., r_n) \neq 0$, by using the base case step we now get

$$\Pr[q(r_1) = 0 \mid Q_k(r_2, ..., r_n) \neq 0] \leq \frac{k}{|S|}$$

From this we get

$$\begin{aligned}\Pr[Q_k(r_1, .., r_n) = 0] &\leq \Pr[Q_k(r_2, .., r_n) = 0] + \Pr[q(r_1) = 0 \mid Q_k(r_2, ..., r_n) \neq 0] \\ &\leq \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}\end{aligned}$$

8.4 Equality between bit strings

8.4.1 Problem

We have two bitstrings, $\bar{a} = (a_0, .., a_{m-1})$ and $\bar{b} = (b_0, .., b_{n-1})$ $(m \leq n)$, our task is to figure out if $\bar{a} = \bar{b}$, check with $O(\lg n)$ bits. Our goal is to be able to compare massive bitstrings. We can interpret a and b as numbers

$$a = \sum_{i \in [m]} a_i 2^i$$

8.4.2 Algorithm

Pick a randome prime $p < n^2$, we now want to check $a \mod p \equiv b \mod p$, here we will use a maximum of $2 \lg n$ bits, since $\lg(n^2) = 2\lg(n)$

8.4.3 Analysis

False positive is the case where $a \neq b$ but $a \mod p \equiv b \mod p$, this only hapens if p divides $|a - b| < 2^n$. This can be viewed as $a = np + a'$, $b = mp + b'$, since $a \mod p \equiv b \mod p$ we get $a' = b'$ and then $a - b = (n - m)p$. The number of primes that divide $|a - b|$ is at most $\lg|a - b| = \lg 2^n = n$.

We can obtain this result by looking at the factorization of $|a - b|$

$$|a - b| = \prod p_i^{d_i}$$

Since, all primes are ≥ 2, then surely the number of primes is at least $\lg n$, this means that

$$\Pr[p \text{ divides } |a - b|] \leq \frac{n}{\#\text{primes} < n^2} \approx \frac{n}{n^2 / \ln(n^2)} = \frac{2 \ln n}{n}$$

8.5 Pattern matching

8.5.1 Problem

Does there exist a j, such that $(a_0, ..., a_{m-1}) = B_j = (b_j, .., b_{j+m-1})$, and lets assume $n >> m$

8.5.2 Algorithm

The first thing we can do is calculate $a \mod p$ and all $B_j \mod p$, however we only want to use $O(n)$ time. We do this by going in reverse.

$$B_j \mod p = 2(B_{j+1} - b_{j+1}2^{m-1}) + b_j \mod p$$

Since we remove the last bit $b_{j+1}2^{m-1}$, shift 1 to the right $(*2)$ and add the new bit b_j.

8.6 Polynomial Pattern matching

8.6.1 Algorithm

Lets define

$$A(x) \triangleq \sum_{i \in [m]} a_i x^i$$

And the same for b. Now we can simply ask $A(x) = B(x)$, just like earlier we can ask if $A(r) \mod p = B(r) \mod p$ where p is a prime $\geq n^2$ and r is random in $\mathbb{Z}_p$. The polinomials have degree m, so the probability of a false positive is $\leq n/p \leq 1/n$

8.7 Multiset matching(not part of exam)

8.7.1 Problem

We have two multisets $\{a_1, .., a_m\}, \{b_1, .., b_m\}$, we want to know if they are equal.

8.7.2 Algorithm

We now look at the set of polynomials

$$A^s[x] = \prod_{i=1}^{m} (x - a_i)$$

and likewise for B^s. Now like the other times we simply ask $A^s(r) = B^s(r)$