Lecture #2: Advanced hashing and concentration bounds

- Bloom filters
- Cuckoo hashing
- Load balancing
- Tail bounds

Idea: For the sake of efficiency, sometime we allow our data structure to make mistakes

Bloom filter: **Hash table that has only false positives**
(may report that a key is present when it is not, but always reports a key that is present)
Very simple and fast

Example: Google Chrome uses a Bloom filter to maintain its list of potentially malicious web sites.
- Most queried keys are not in the table
- If a key is in the table, can check against a slower (errorless) hash table

Many applications in networking (see survey by Broder and Mitzenmacher)

Data structure: Universe $\mathcal{U}$. Parameters $k, M \geq 1$

Maintain an array A of M bits; initially $A[0] = A[1] = \cdots = A[M-1] = 0$

Choose k hash functions $h_1, h_2, \dots, h_k : \mathcal{U} \to [M]$

(assume completely random functions for sake of analysis)

Data structure: Universe $\mathcal{U}$. Parameters $k, M \geq 1$

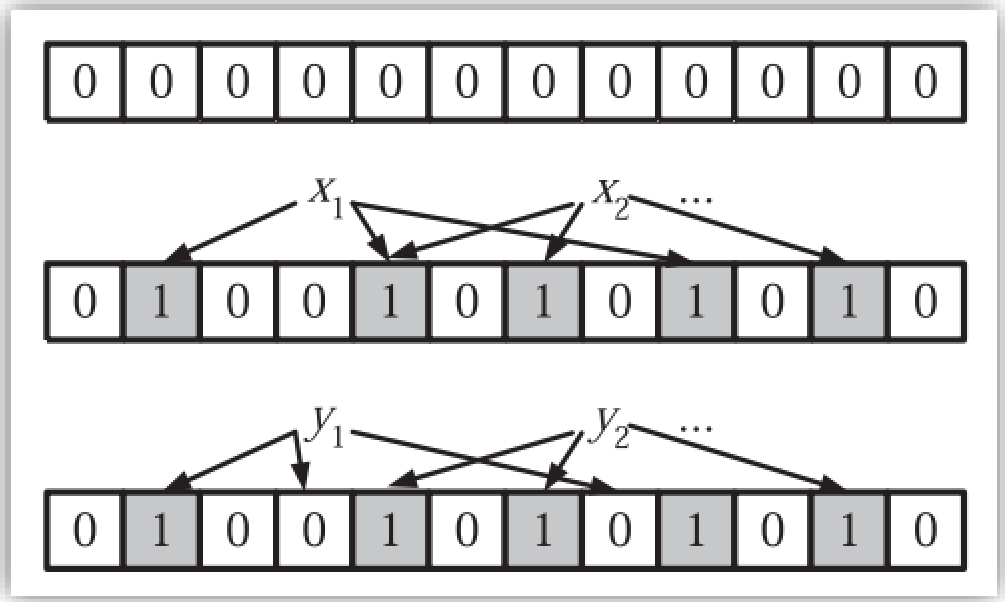
Maintain an array A of M bits; initially $A[0] = A[1] = \cdots = A[M-1] = 0$

Choose k hash functions $h_1, h_2, \ldots, h_k : \mathcal{U} \to [M]$

(assume completely random functions for sake of analysis)

To add a key $x \in \mathcal{U}$ to the dictionary $S \subseteq \mathcal{U}$, set bits

$A[h_1(x)] := 1, A[h_2(x)] := 1, \ldots, A[h_k(x)] := 1$



Data structure: Universe $\mathcal{U}$. Parameters $k, M \geq 1$

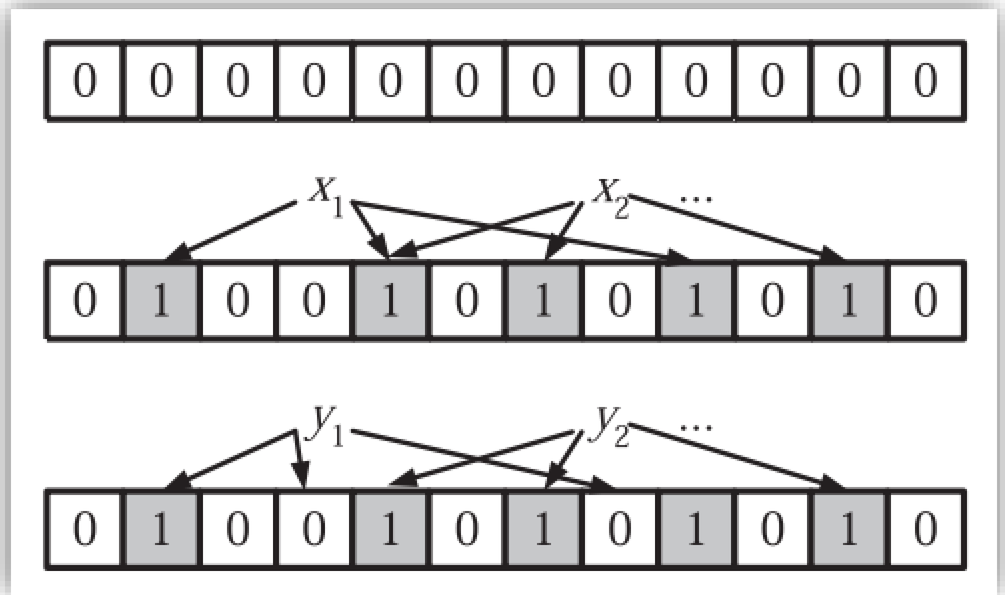
Maintain an array A of M bits; initially $A[0] = A[1] = \cdots = A[M-1] = 0$

Choose k hash functions $h_1, h_2, \ldots, h_k : \mathcal{U} \to [M]$

(assume completely random functions for sake of analysis)

To add a key $x \in \mathcal{U}$ to the dictionary $S \subseteq \mathcal{U}$, set bits

$A[h_1(x)] \coloneqq 1, A[h_2(x)] \coloneqq 1, \ldots, A[h_k(x)] \coloneqq 1$



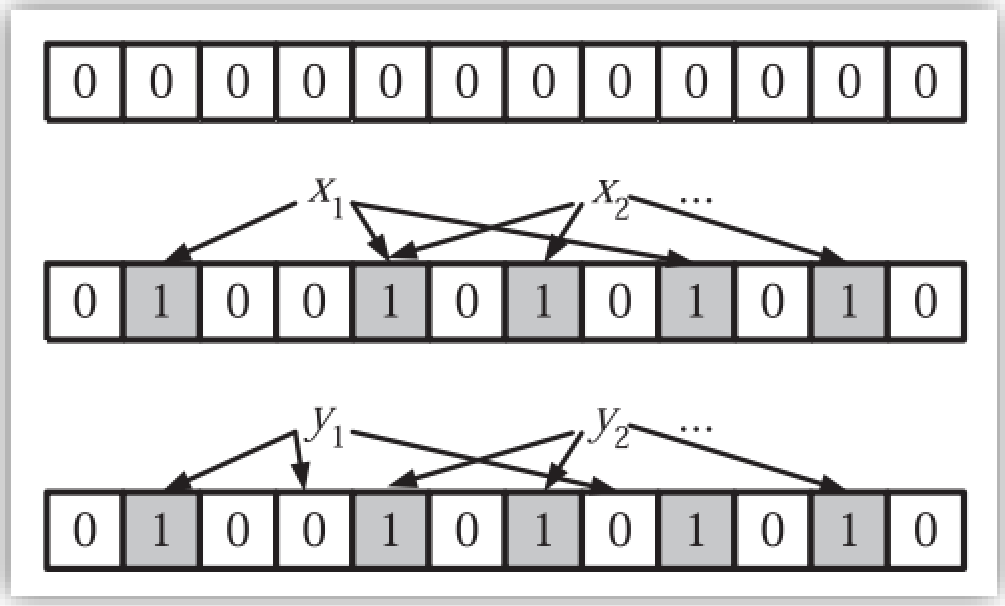
To answer a query: $q \in S$?

Check whether $A[h_i(x)] = 1$ for all $i = 1, 2, \ldots, k$

If yes, answer **Yes**. If no, answer **No**.

No false negatives: Clearly if $x \in S$, we return **Yes**.

But there is some chance that other keys have caused the bits in positions $h_1(x), \ldots, h_k(x)$ to be set even if $x \notin S$.



No false negatives: Clearly if $x \in S$, we return **Yes**.

But there is some chance that other keys have caused the bits in positions $h_1(x), \dots, h_k(x)$ to be set even if $x \notin S$.

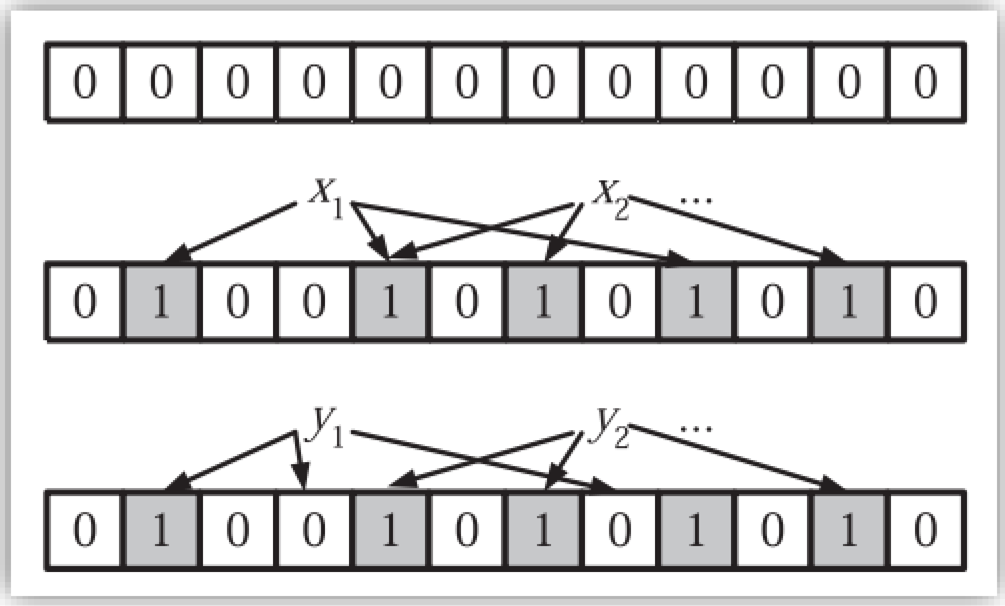
Heuristic analysis:

Let us assume that $|S| = n$.

Compute $\mathbb{P}[A[\ell] = 0]$ for some location $\ell \in [M]$:

$$p(k, N) = \left(1 - \frac{1}{M}\right)^{kN} \approx e^{-\frac{kN}{M}}$$

(Here we use the approximation $\left(1 - \frac{1}{M}\right)^M \approx e^{-1}$ for M large enough.)



No false negatives: Clearly if $x \in S$, we return **Yes**.

But there is some chance that other keys have caused the bits in positions $h_1(x), \dots, h_k(x)$ to be set even if $x \notin S$.

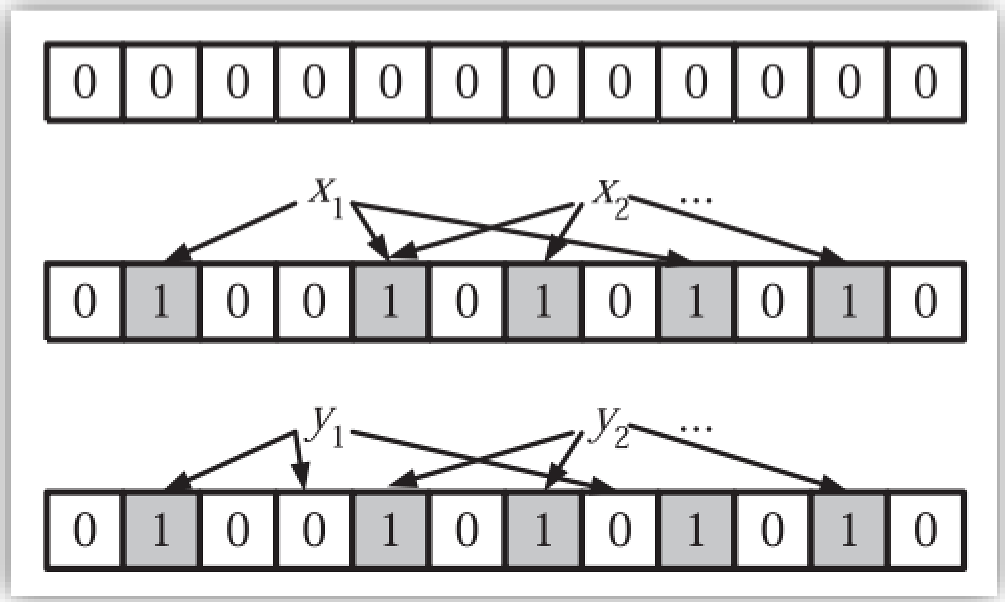
Heuristic analysis:

(Here we use the approximation $\left(1 - \frac{1}{M}\right)^M \approx e^{-1}$ for M large enough.)

Let us assume that $|S| = n$.

Compute $\mathbb{P}[A[\ell] = 0]$ for some location $\ell \in [M]$:

$$p(k, N) = \left(1 - \frac{1}{M}\right)^{kN} \approx e^{-\frac{kN}{M}}$$



If each location in A is 0 with probability $p(k, N)$, then a false positive for $x \notin S$ should happen with probability at most

$$\left(1 - p(k, N)\right)^k \approx \left(1 - e^{-\frac{kN}{M}}\right)^k$$

Heuristic analysis:

If each location in A is 0 with probability $p(k, N)$, then a false positive for $x \notin S$ should happen with probability at most

$$\left(1 - p(k, N)\right)^k \approx \left(1 - e^{-\frac{kN}{M}}\right)^k$$

Heuristic analysis:

If each location in A is 0 with probability $p(k,N)$, then a false positive for $x \notin S$ should happen with probability at most

$$\left(1-p(k,N)\right)^k \approx \left(1-e^{-\frac{kN}{M}}\right)^k$$

But the **actual** fraction of $0's$ in the hash table is a random variable $X_{k,N}$ with **expectation**

$$\mathbb{E}\left[X_{k,N}\right] = p(k,N)$$

To get the analysis right, we need a **concentration bound**: Want to say that $X_{k,N}$ is close to its expected value with **high probability**. [We will return to this in the 2nd half of the lecture]

Heuristic analysis:

If each location in A is 0 with probability $p(k,N)$, then a false positive for $x \notin S$ should happen with probability at most

$$\left(1 - p(k,N)\right)^k \approx \left(1 - e^{-\frac{kN}{M}}\right)^k$$

But the **actual** fraction of $0's$ in the hash table is a random variable $X_{k,N}$ with **expectation**

$$\mathbb{E}\left[X_{k,N}\right] = p(k,N)$$

To get the analysis right, we need a **concentration bound**: Want to say that $X_{k,N}$ is close to its expected value with **high probability**. [We will return to this in the 2nd half of the lecture]

If the heuristic analysis is correct, it gives nice estimates:

For instance, if $M = 8N$, then choosing the optimal value of $k = 7$ gives false positive rate about 2%.

Lecture #2: Advanced hashing and concentration bounds

- Bloom filters
- Cuckoo hashing
- Load balancing
- Tail bounds

Cuckoo hashing is a hash scheme with worst-case constant lookup time. The name derives from the behavior of some species of cuckoo, where the cuckoo chick pushes the other eggs or young out of the nest when it hatches; analogously, inserting a new key into a cuckoo hashing table may push an older key to a different location in the table.

Idea: Simple hashing without errors

Lookups are worst case $O(1)$ time

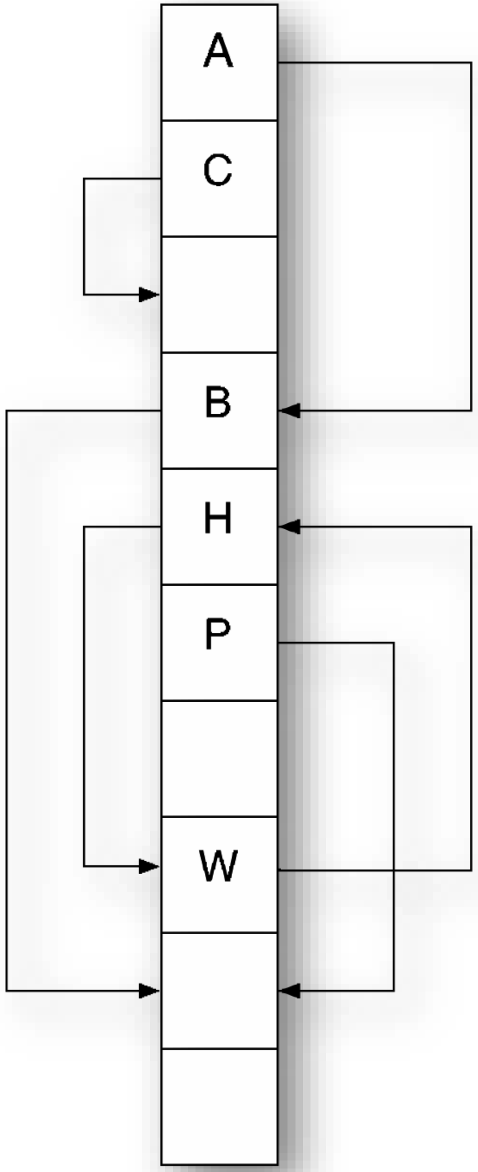
Deletions are worst case $O(1)$ time

Insertions are **expected** $O(1)$ time

Insertion time is $O(1)$ with good probability [will require a concentration bound]

Data structure: Two tables A_1 and A_2 both of size $M = O(N)$
Two hash functions $h_1, h_2 : \mathcal{U} \rightarrow [M]$
(will assume hash functions are fully random)

When an element $x \in S$ is inserted, if either $A_1[h_1(x)]$ or $A_2[h_2(x)]$ is empty, store x there.



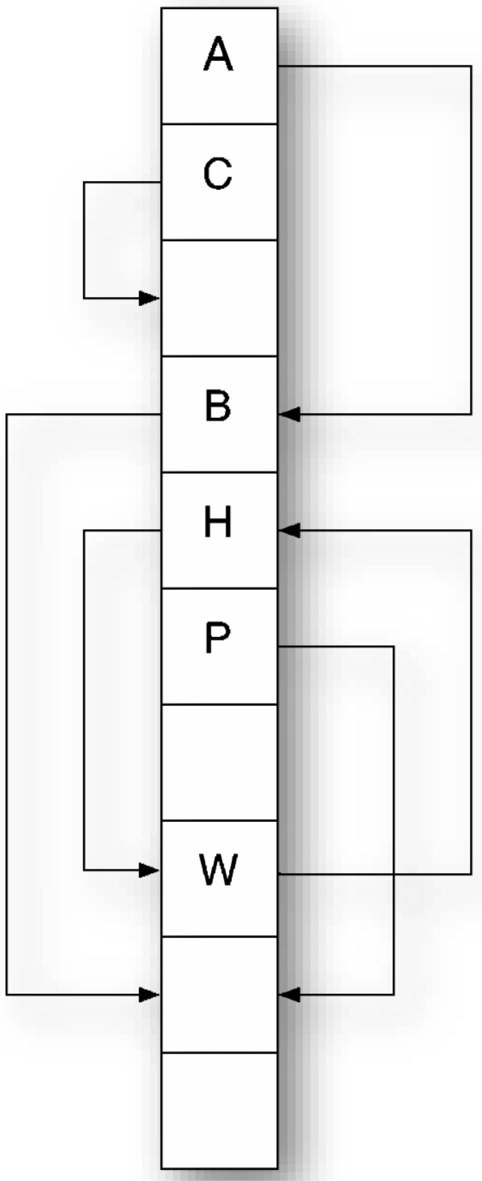
Data structure: Two tables A_1 and A_2 both of size $M = O(N)$
Two hash functions $h_1, h_2 : \mathcal{U} \to [M]$
(will assume hash functions are fully random)

When an element $x \in S$ is inserted, if either $A_1[h_1(x)]$ or $A_2[h_2(x)]$ is empty, store x there.

Bump:

If both locations are occupied, then place x in $A_1[h_1(x)]$ and bump the current occupant.

Whenever an element z is bumped from $A_i[h_i(z)]$, attempt to store it in the other location $A_j[h_j(z)]$
(here $(i, j) = (1,2)$ or $(2,1)$)



Data structure: Two tables A_1 and A_2 both of size $M = O(N)$
Two hash functions $h_1, h_2 : \mathcal{U} \to [M]$
(will assume hash functions are fully random)

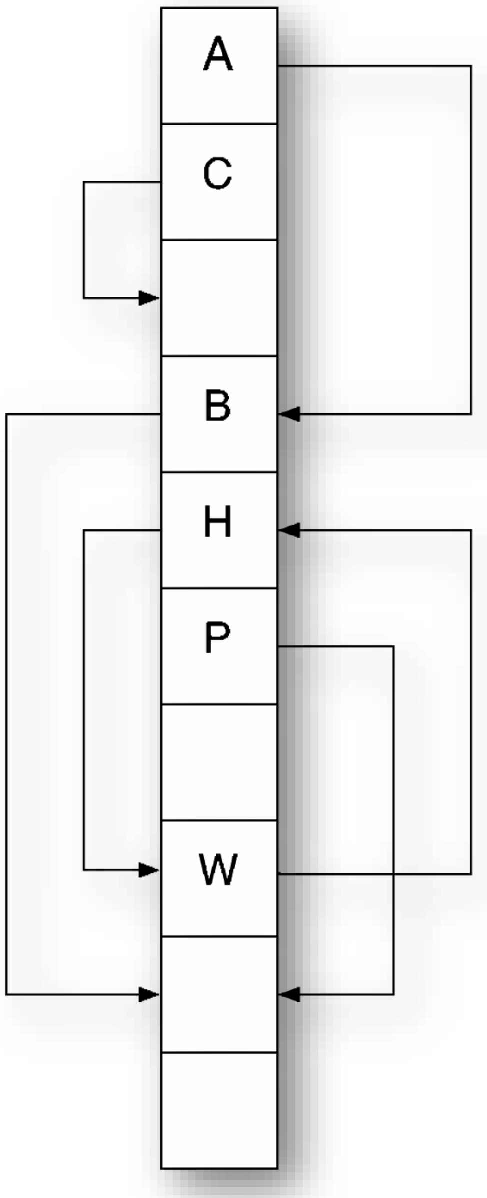
When an element $x \in S$ is inserted, if either $A_1[h_1(x)]$ or $A_2[h_2(x)]$ is empty, store x there.

Bump:

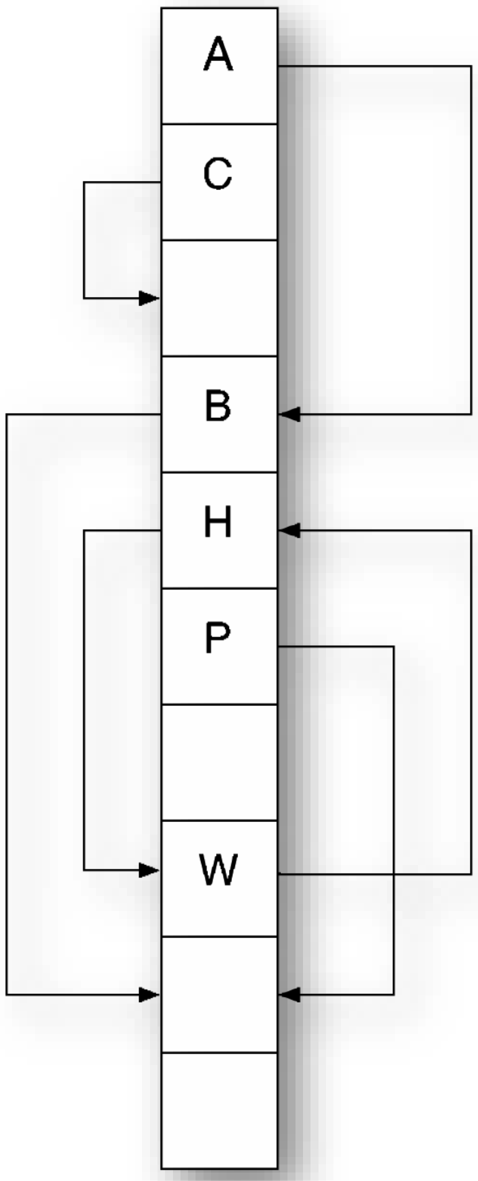
If both locations are occupied, then place x in $A_1[h_1(x)]$ and bump the current occupant.

Whenever an element z is bumped from $A_i[h_i(z)]$, attempt to store it in the other location $A_j[h_j(z)]$
(here $(i, j) = (1,2)$ or $(2,1)$)

Abort: After $6 \log N$ consecutive bumps, stop the process and build a fresh hash table using new random hash functions h_1, h_2.



Alternately (as in the picture), we can use a single table with $2M$ entries and two hash functions $h_1, h_2\colon \mathcal{U} \to [2M]$ (with the same "bumping" algorithm)

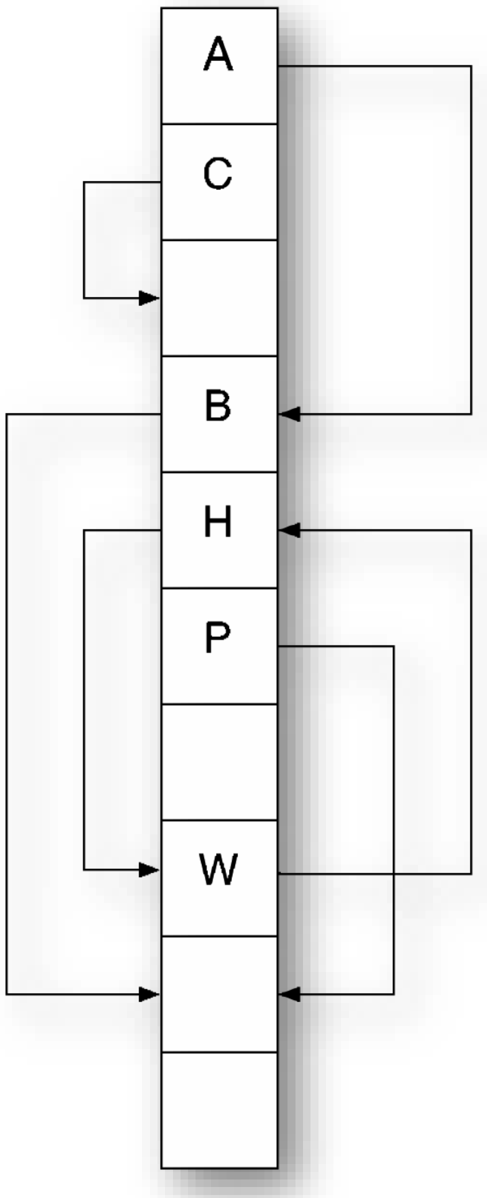


Arrows represent the alternate location for each key.

If we insert an item at the location of A, it will get bumped, thereby bumping B, and then we are done.

Cycles are possible (where the insertion process never completes). What's an example?

Data structure: Two tables A_1 and A_2 both of size $M = O(N)$
Two hash functions $h_1, h_2 : \mathcal{U} \to [M]$
(will assume hash functions are fully random)



Theorem:

Expected time to perform an insert operation is $O(1)$ if $M \geq 4N$.

Data structure: Two tables A_1 and A_2 both of size $M = O(N)$
Two hash functions $h_1, h_2 : \mathcal{U} \to [M]$
(will assume hash functions are fully random)

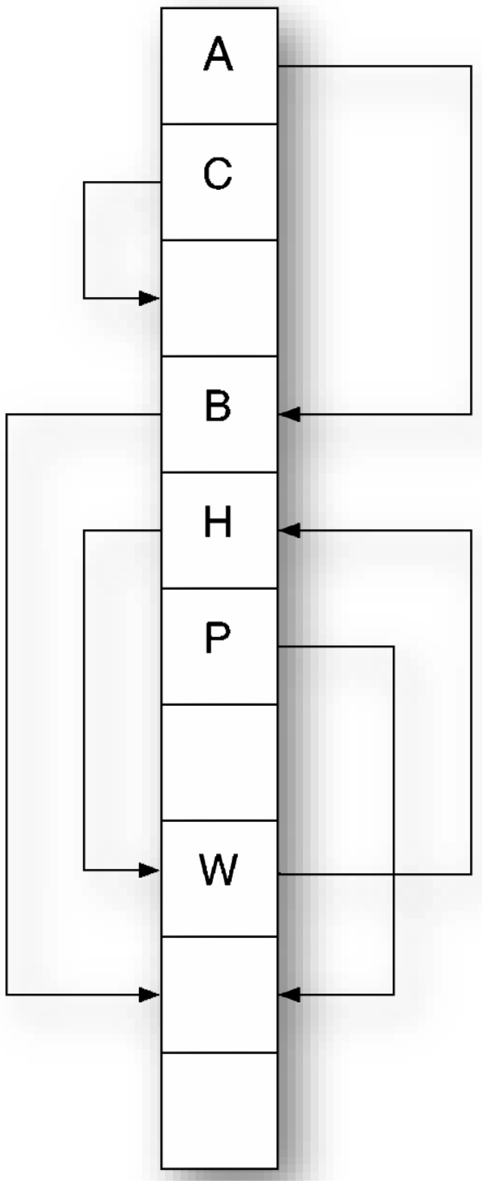
Theorem:

Expected time to perform an insert operation is $O(1)$ if $M \geq 4N$.

Pretty good… but only 25% memory utilization.

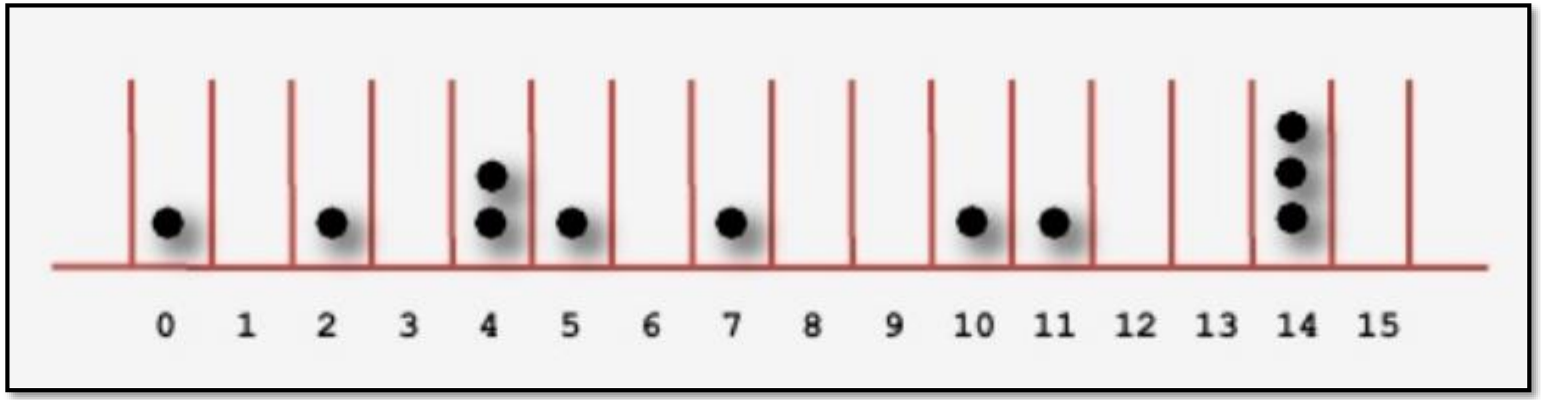
Can actually get about 50% memory utilization.

Experimentally, with 3 hash functions instead of 2, can get $\approx 90\%$ utilization, but it is an open question to provide tight analyses for d hash functions when $d \geq 3$.



Lecture #2: Advanced hashing and concentration bounds

- Bloom filters
- Cuckoo hashing
- Load balancing
- Tail bounds



Suppose we have N jobs to assign to N servers.

Clearly we could achieve a load of one job/server, but this might result in an expensive/hard-to-parallelize allocation rule.

Suppose we have N jobs to assign to N servers.

Clearly we could achieve a load of one job/server, but this might result in an expensive/hard-to-parallelize allocation rule.

We could **hash** the balls into bins. Let's again consider the case of a uniformly random hash function $h : [N] \to [N]$

Suppose we have N jobs to assign to N servers.

Clearly we could achieve a load of one job/server, but this might result in an expensive/hard-to-parallelize allocation rule.

We could **hash** the balls into bins. Let's again consider the case of a uniformly random hash function $h : [N] \to [N]$

Claim: The max-loaded server has < $8 \log N \,/ \log \log N$ jobs with probability at least $1 - 1/N$

Suppose we have N jobs to assign to N servers.

Clearly we could achieve a load of one job/server, but this might result in an expensive/hard-to-parallelize allocation rule.

We could **hash** the balls into bins. Let's again consider the case of a uniformly random hash function $h : [N] \to [N]$

Claim: The max-loaded server has < $8 \log N \,/\, \log \log N$ jobs with probability at least $1 - 1/N$

Proof: Probability that a fixed server $i \in \{1,2, \dots, N\}$ gets at least k jobs is at most

$$\binom{N}{k}\left(\frac{1}{N}\right)^k \leq \frac{N^k}{k!} \cdot \frac{1}{N^k} \leq \frac{1}{k!} \leq k^{-\frac{k}{2}}$$

Suppose we have N jobs to assign to N servers.

Clearly we could achieve a load of one job/server, but this might result in an expensive/hard-to-parallelize allocation rule.

We could **hash** the balls into bins. Let's again consider the case of a uniformly random hash function $h : [N] \to [N]$

Claim: The max-loaded server has < $8 \log N \,/ \log \log N$ jobs with probability at least $1 - 1/N$

Proof: Probability that a fixed server $i \in \{1,2, \dots, N\}$ gets at least k jobs is at most

$$\binom{N}{k}\left(\frac{1}{N}\right)^k \le \frac{N^k}{k!} \cdot \frac{1}{N^k} \le \frac{1}{k!} \le k^{-\frac{k}{2}}$$

If we choose $k = \frac{8 \log N}{\log \log N}$ this is at most $1/N^2$

Explanation: $k^{\frac{k}{2}} \ge \left(\sqrt{\log N}\right)^{\frac{4 \log N}{\log \log N}} \ge 2^{2 \log N} = N^2$

Suppose we have N jobs to assign to N servers.

Clearly we could achieve a load of one job/server, but this might result in an expensive/hard-to-parallelize allocation rule.

We could **hash** the balls into bins. Let's again consider the case of a uniformly random hash function $h : [N] \to [N]$

Claim: The max-loaded server has < $8 \log N \,/\, \log\log N$ jobs with probability at least $1 - 1/N$

Proof: Probability that a fixed server $i \in \{1,2,\ldots,N\}$ gets at least k jobs is at most

$$\binom{N}{k}\left(\frac{1}{N}\right)^k \leq \frac{N^k}{k!} \cdot \frac{1}{N^k} \leq \frac{1}{k!} \leq k^{-\frac{k}{2}}$$

If we choose $k = \frac{8 \log N}{\log \log N}$ this is at most $1/N^2$

Now a **union bound** shows that the probability of **any** server getting at least k jobs is at most $1/N$.

This is an example of a **concentration bound**.

Let X_i be the number of jobs assigned to the ith server.
By linearity of expectation, $\mathbb{E}[X_i] = \sum_{j=1}^{N} \mathbb{P}[\text{job } j \to \text{server } i] = N \cdot (1/N) = 1$.

Claim: The max-loaded server has < $8 \log N \,/\, \log\log N$ jobs with probability at least $1 - 1/N$

Proof: Probability that a fixed server $i \in \{1,2, \ldots, N\}$ gets at least k jobs is at most

$$\binom{N}{k}\left(\frac{1}{N}\right)^k \leq \frac{N^k}{k!} \cdot \frac{1}{N^k} \leq \frac{1}{k!} \leq k^{-\frac{k}{2}}$$

If we choose $k = \frac{8 \log N}{\log\log N}$ this is at most $1/N^2$

Now a **union bound** shows that the probability of **any** server getting at least k jobs is at most $1/N$.

This is an example of a **concentration bound**.

Let X_i be the number of jobs assigned to the ith server.
By linearity of expectation, $\mathbb{E}[X_i] = \sum_{j=1}^{N} \mathbb{P}[\text{job } j \to \text{server } i] = N \cdot (1/N) = 1$.

We showed that $\mathbb{P}\left[X_i \geq \frac{8 \log N}{\log \log N}\right] \leq \frac{1}{N^2}$ and then took a union bound over all N servers.

Claim: The max-loaded server has < $8 \log N \,/\, \log \log N$ jobs with probability at least $1 - 1/N$

Proof: Probability that a fixed server $i \in \{1,2, \ldots , N\}$ gets at least k jobs is at most

$$\binom{N}{k}\left(\frac{1}{N}\right)^k \leq \frac{N^k}{k!} \cdot \frac{1}{N^k} \leq \frac{1}{k!} \leq k^{-\frac{k}{2}}$$

If we choose $k = \frac{8 \log N}{\log \log N}$ this is at most $1/N^2$

Now a **union bound** shows that the probability of **any** server getting at least k jobs is at most $1/N$.

This is an example of a **concentration bound**.

Let X_i be the number of jobs assigned to the ith server.
By linearity of expectation, $\mathbb{E}[X_i] = \sum_{j=1}^{N} \mathbb{P}[\text{job } j \to \text{server } i] = N \cdot (1/N) = 1$.

We showed that $\mathbb{P}\left[X_i \geq \frac{8 \log N}{\log \log N}\right] \leq \frac{1}{N^2}$ and then took a union bound over all N servers.

This is a common analysis technique:

If a random variable (like X_i) depends in a "smooth" way on the outcome of many **independent** events, then it is likely not too far from its expectation.

This is an example of a **concentration bound**.

Let X_i be the number of jobs assigned to the ith server.
By linearity of expectation, $\mathbb{E}[X_i] = \sum_{j=1}^{N} \mathbb{P}[\text{job } j \to \text{server } i] = N \cdot (1/N) = 1$.

We showed that $\mathbb{P}\left[X_i \geq \frac{8 \log N}{\log \log N}\right] \leq \frac{1}{N^2}$ and then took a union bound over all N servers.

This is a common analysis technique:

If a random variable (like X_i) depends in a "smooth" way on the outcome of many **independent** events, then it is likely not too far from its expectation.

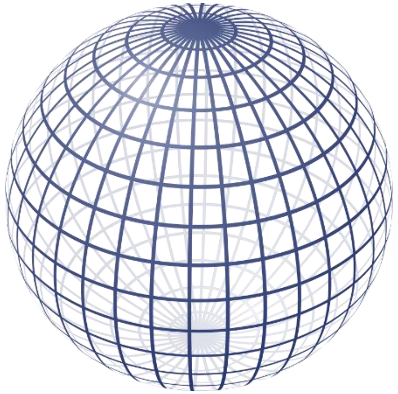
"Smooth" in this case means that the outcome of any decision (where to put job j) does not affect the value of X_i by too much (only by 1).

Is it concentrated? [why or why not?]

#1: Choose a uniformly random vector $X \in \mathbb{R}^n$ with $\|X\| = \sqrt{X_1^2 + X_2^2 + \cdots + X_n^2} = 1$

What is $\mathbb{E}[X_1^2]$?

What is the typical value of the maximum: $\max\ (|X_1|, |X_2|, \dots, |X_n|)$?



#2 Rich get richer: Suppose we have N people. Everyone starts with 1 dollar.

We assign N^2 more dollars in rounds.

***i*th round:** If person j already has n_j dollars, we give them the ith dollar with probability

$$\frac{n_j}{i-1}$$

i.e., with probability to the proportional the amount of money they already have.

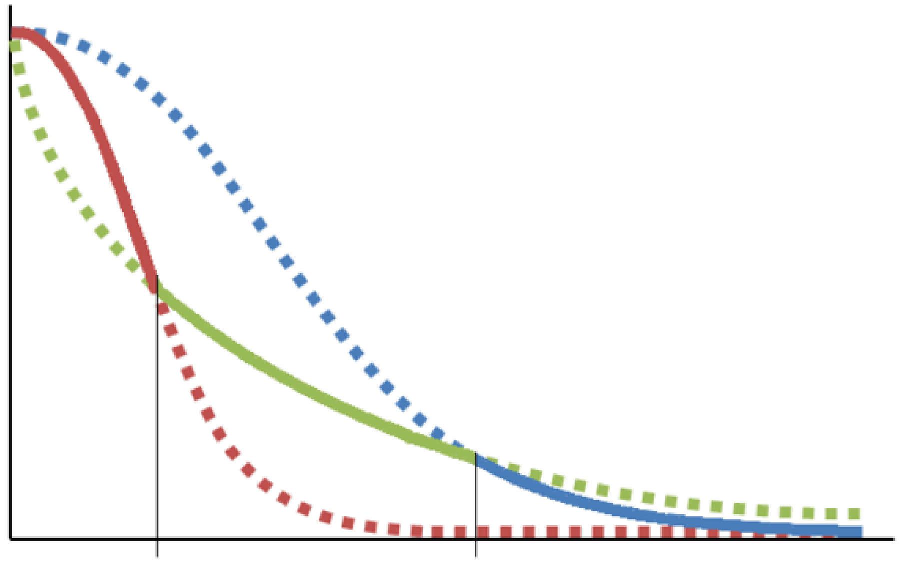
Let X_i be the amount of money person i ends up with.

What is the typical value of X_1? Is X_1 concentrated?

What is the typical value of $\max(X_1, X_2, \dots, X_n)$? Is it concentrated?

Lecture #2: Advanced hashing and concentration bounds

- Bloom filters
- Cuckoo hashing
- Load balancing
- Tail bounds



The more you know: The more information we have about a random variable, the stronger the concentration we can prove.

The more you know: The more information we have about a random variable, the stronger the concentration we can prove.

The most basic concentration bound is **Markov's inequality**. It requires knowing only the expected value:

If X is a non-negative random variable, then for any $\lambda \geq 1$,

$$\mathbb{P}[X \geq \lambda] \leq \frac{\mathbb{E}[X]}{\lambda}$$

Proof? (it's written there)

The more you know: The more information we have about a random variable, the stronger the concentration we can prove.

The most basic concentration bound is **Markov’s inequality**. It requires knowing only the expected value:

If X is a non-negative random variable, then for any $\lambda \geq 1$,

$$\mathbb{P}[X \geq \lambda] \leq \frac{\mathbb{E}[X]}{\lambda}$$

Proof? (it’s written there)

Example:
If your expected revenue is \$10,000, then the probability to make \$1,000 is at most 1/10.

Markov's inequality: If X is a non-negative random variable, then for any $\lambda \geq 1$,

$$\mathbb{P}[X \geq \lambda] \leq \frac{\mathbb{E}[X]}{\lambda}$$

A permutation is an invertible mapping $\pi : \{1,2,\ldots,n\} \rightarrow \{1,2,\ldots,n\}$

A number j is called a **fixed point** of π if $\pi(j) = j$.

Exercise: Prove that if π is a uniformly random permutation, then

$$\mathbb{P}[\pi \text{ has more than } k \text{ fixed points}] \leq \frac{1}{k}$$

Recall that the **variance** of a random variable X is the value

$$\operatorname{var}(X) = \sigma^2 = \mathbb{E}\left(X - \mathbb{E}X\right)^2$$

Recall that the **variance** of a random variable X is the value

$$\mathrm{var}(X) = \sigma^2 = \mathbb{E}\left(X - \mathbb{E}X\right)^2$$

Chebyshev's inequality:

If X is a random variable with $\mathrm{var}(X) = \sigma^2$, then for any $\lambda > 0$,

$$\mathbb{P}[|X - \mathbb{E}X| \geq \lambda\sigma] \leq \frac{1}{\lambda^2}$$

Recall that the **variance** of a random variable X is the value

$$\mathrm{var}(X) = \sigma^2 = \mathbb{E}\,(X - \mathbb{E}X)^2$$

Chebyshev's inequality:
If X is a random variable with $\mathrm{var}(X) = \sigma^2$, then for any $\lambda > 0$,

$$\mathbb{P}[|X - \mathbb{E}X| \geq \lambda\sigma] \leq \frac{1}{\lambda^2}$$

Proof: Apply Markov's inequality to the random variable $Y = (X - \mathbb{E}X)^2$

Recall that the **variance** of a random variable X is the value

$$\mathrm{var}(X) = \sigma^2 = \mathbb{E}\left(X - \mathbb{E}X\right)^2$$

Chebyshev's inequality:

If X is a random variable with $\mathrm{var}(X) = \sigma^2$, then for any $\lambda > 0$,

$$\mathbb{P}[|X - \mathbb{E}X| \geq \lambda\sigma] \leq \frac{1}{\lambda^2}$$

Proof: Apply Markov's inequality to the random variable $Y = (X - \mathbb{E}X)^2$

Application:

Suppose we map N balls into N bins using a 2-universal hash family $\mathcal{H}$. Then with probability at least $1/2$, the maximum load is at most $O(\sqrt{N})$.

Application:

Suppose we map N balls into N bins using a 2-universal hash family $\mathcal{H}$. Then with probability at least $1/2$, the maximum load is at most $O(\sqrt{N})$.

Let L_i be the load of bin i.
Let X_{ij} be the indicator random variable such that $X_{ij} = 1 \leftrightarrow i$th bin gets the jth ball.
Note that $\mathbb{E}[X_{ij}] = 1/N$ for each $i = 1, \ldots, N$.

Application:

Suppose we map N balls into N bins using a 2-universal hash family $\mathcal{H}$. Then with probability at least $1/2$, the maximum load is at most $O(\sqrt{N})$.

Let L_i be the load of bin i.
Let X_{ij} be the indicator random variable such that $X_{ij} = 1 \leftrightarrow i$th bin gets the jth ball.
Note that $\mathbb{E}[X_{ij}] = 1/N$ for each $i = 1, \ldots, N$.

Exercise: For any random variable Y, $\mathrm{var}(Y) = \mathbb{E}(Y^2) - (\mathbb{E}Y)^2$

Application:

Suppose we map N balls into N bins using a 2-universal hash family $\mathcal{H}$. Then with probability at least $1/2$, the maximum load is at most $O(\sqrt{N})$.

Let L_i be the load of bin i.
Let X_{ij} be the indicator random variable such that $X_{ij} = 1 \leftrightarrow i$th bin gets the jth ball.
Note that $\mathbb{E}[X_{ij}] = 1/N$ for each $i = 1, \ldots, N$.

Exercise: For any random variable Y, $\mathrm{var}(Y) = \mathbb{E}(Y^2) - (\mathbb{E}Y)^2$

So write: $\mathrm{var}(L_i) = \mathbb{E}\left[(X_{i1} + \cdots + X_{iN})^2\right] - 1$

We have $\mathbb{E}[X_{ij}^2] = \mathbb{E}[X_{ij}] = 1/N$ and $\mathbb{E}[X_{ij}X_{ik}] = \mathbb{P}[h(j) = h(k) = i] \leq 1/N^2$ using the 2-universal property, so

$$\mathrm{var}(L_i) \leq N \cdot \left(\frac{1}{N}\right) + \frac{N(N-1)}{N^2} - 1 = 1 - \frac{1}{N} \leq 1$$

$$\mathrm{var}(L_i) \leq N \cdot \left(\frac{1}{N}\right) + \frac{N(N-1)}{N^2} - 1 = 1 - \frac{1}{N} \leq 1$$

Chebyshev's inequality:

If X is a random variable with $\mathrm{var}(X) = \sigma^2$, then for any $\lambda > 0$,

$$\mathbb{P}[|X - \mathbb{E}X| \geq \lambda\sigma] \leq \frac{1}{\lambda^2}$$

$$\mathrm{var}(L_i) \le N \cdot \left(\frac{1}{N}\right) + \frac{N(N-1)}{N^2} - 1 = 1 - \frac{1}{N} \le 1$$

Chebyshev's inequality:

If X is a random variable with $\mathrm{var}(X) = \sigma^2$, then for any $\lambda > 0$,

$$\mathbb{P}[|X - \mathbb{E}X| \ge \lambda\sigma] \le \frac{1}{\lambda^2}$$

Apply Chebyshev's inequality to L_i, yielding

$$\mathbb{P}[|L_i - 1| \ge \lambda] \le \frac{1}{\lambda^2}$$

$$\mathrm{var}(L_i) \le N \cdot \left(\frac{1}{N}\right) + \frac{N(N-1)}{N^2} - 1 = 1 - \frac{1}{N} \le 1$$

Chebyshev's inequality:

If X is a random variable with $\mathrm{var}(X) = \sigma^2$, then for any $\lambda > 0$,

$$\mathbb{P}[|X - \mathbb{E}X| \ge \lambda\sigma] \le \frac{1}{\lambda^2}$$

Apply Chebyshev's inequality to L_i, yielding

$$\mathbb{P}[|L_i - 1| \ge \lambda] \le \frac{1}{\lambda^2}$$

Thus $\mathbb{P}\left[|L_i - 1| \ge \sqrt{2N}\right] \le \frac{1}{2N}$, so a union bound yields

$$\mathbb{P}\left[\max(L_1, \dots, L_N) \ge \sqrt{2N} + 1\right] \le \frac{1}{2}$$

Chebyshev's inequality:

If X is a random variable with $\mathrm{var}(X) = \sigma^2$, then for any $\lambda > 0$,

$$\mathbb{P}[|X - \mathbb{E}X| \geq \lambda\sigma] \leq \frac{1}{\lambda^2}$$

Suppose we choose n independent random voters and ask them whether they prefer candidate #1 over candidate #2.

We see outcomes $X_1, X_2, \ldots, X_n \in \{0,1\}$.

Let p be the actual percentage of the population the prefers candidate #1 and let $\hat{p} = (X_1 + \cdots + X_n)/n$ denote the **empirical mean.**

Exercise:

Prove that if we want $|p - \hat{p}| \leq \epsilon$ to hold with 99% probability, then we need only sample $n = O(1/\epsilon^2)$ voters.

Hoeffding's inequality:

Let $X_1, \dots, X_n$ be a sequence of independent random variables where, for each $1 \leq i \leq n$, we have $a_i \leq X_i \leq b_i$. Let $X = (X_1 + \cdots + X_n)/n$. Then:

$$\mathbb{P}[|X - \mathbb{E}X| \geq \lambda] \leq 2\, e^{-\frac{2\lambda^2 n^2}{\sum_{i=1}^{n}(a_i - b_i)^2}}$$

Hoeffding's inequality:

Let $X_1, \dots, X_n$ be a sequence of independent random variables where, for each $1 \leq i \leq n$, we have $a_i \leq X_i \leq b_i$. Let $X = (X_1 + \cdots + X_n)/n$. Then:

$$\mathbb{P}[|X - \mathbb{E}X| \geq \lambda] \leq 2\, e^{-\frac{2\lambda^2 n^2}{\sum_{i=1}^n (a_i - b_i)^2}}$$

Suppose we wanted our poll from the previous slide to be correct with probability at least $1 - \delta$. Chebyshev's inequality would tell us we need at most $O\left(\frac{1}{\epsilon^2 \sqrt{\delta}}\right)$ samples.

Hoeffding's inequality:

Let $X_1, \ldots, X_n$ be a sequence of independent random variables where, for each $1 \le i \le n$, we have $a_i \le X_i \le b_i$. Let $X = (X_1 + \cdots + X_n)/n$. Then:

$$\mathbb{P}[|X - \mathbb{E}X| \ge \lambda] \le 2\, e^{-\frac{2\lambda^2 n^2}{\sum_{i=1}^{n}(a_i - b_i)^2}}$$

Suppose we wanted our poll from the previous slide to be correct with probability at least $1 - \delta$. Chebyshev's inequality would tell us we need at most $O\left(\frac{1}{\epsilon^2\sqrt{\delta}}\right)$ samples.

Setting $a_i = 0$, $b_i = 1$, and $\lambda = \epsilon$ in Hoeffding's inequality gives

$$\mathbb{P}[|\hat{p} - p| \ge \epsilon] \le 2e^{-2\epsilon^2 n}$$

so we only need $n \le O\left(\frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}\right)$ samples.

Chernoff bound (multiplicative):

Let $X_1, \dots, X_n$ be a sequence of independent $\{0,1\}$-valued random variables.

Let $p_i = \mathbb{E}[X_i]$, $X = X_1 + X_2 + \cdots + X_n$, $\mu = \mathbb{E}[X]$. Then for every $\beta \geq 1$:

$$\mathbb{P}[X \geq \beta\mu] \leq \left(\frac{e^{\beta-1}}{\beta^\beta}\right)^\mu \qquad \mathbb{P}[X \leq \mu/\beta] \leq \left(\frac{e^{\frac{1}{\beta}-1}}{\beta^\beta}\right)^\mu$$

Chernoff bound (multiplicative):

Let $X_1, \dots, X_n$ be a sequence of independent $\{0,1\}$-valued random variables.

Let $p_i = \mathbb{E}[X_i]$, $X = X_1 + X_2 + \cdots + X_n$, $\mu = \mathbb{E}[X]$. Then for every $\beta \geq 1$:

$$\mathbb{P}[X \geq \beta\mu] \leq \left(\frac{e^{\beta-1}}{\beta^\beta}\right)^\mu \qquad \mathbb{P}[X \leq \mu/\beta] \leq \left(\frac{e^{\frac{1}{\beta}-1}}{\beta^\beta}\right)^\mu$$

Reproduce balls in bins:

N balls thrown randomly into N bins.

$X_i = 1$ if ith ball ends up in first bin and $X_i = 0$ otherwise.

Then X = # of balls in first bin. As we calculated earlier, $\mathbb{E}[X] = 1$

For $\beta \approx \frac{\log N}{\log \log N}$, the Chernoff bound gives $\mathbb{P}[X \geq \beta] \leq 1/N^2$

Chernoff bound (multiplicative):

Let $X_1, \dots, X_n$ be a sequence of independent $\{0,1\}$-valued random variables.

Let $p_i = \mathbb{E}[X_i]$, $X = X_1 + X_2 + \cdots + X_n$, $\mu = \mathbb{E}[X]$. Then for every $\beta \geq 1$:

$$\mathbb{P}[X \geq \beta\mu] \leq \left(\frac{e^{\beta-1}}{\beta^\beta}\right)^\mu \qquad \mathbb{P}[X \leq \mu/\beta] \leq \left(\frac{e^{\frac{1}{\beta}-1}}{\beta^\beta}\right)^\mu$$

Reproduce balls in bins:

N balls thrown randomly into N bins.

$X_i = 1$ if ith ball ends up in first bin and $X_i = 0$ otherwise.

Then X = # of balls in first bin. As we calculated earlier, $\mathbb{E}[X] = 1$

For $\beta \approx \frac{\log N}{\log \log N}$, the Chernoff bound gives $\mathbb{P}[X \geq \beta] \leq 1/N^2$

This type of analysis works for much more complicated kinds of events (see homework #2)

Heuristic analysis:

If each location in A is 0 with probability $p(k,N)$, then a false positive for $x \notin S$ should happen with probability at most

$$\left(1 - p(k,N)\right)^k \approx \left(1 - e^{-\frac{kN}{M}}\right)^k$$

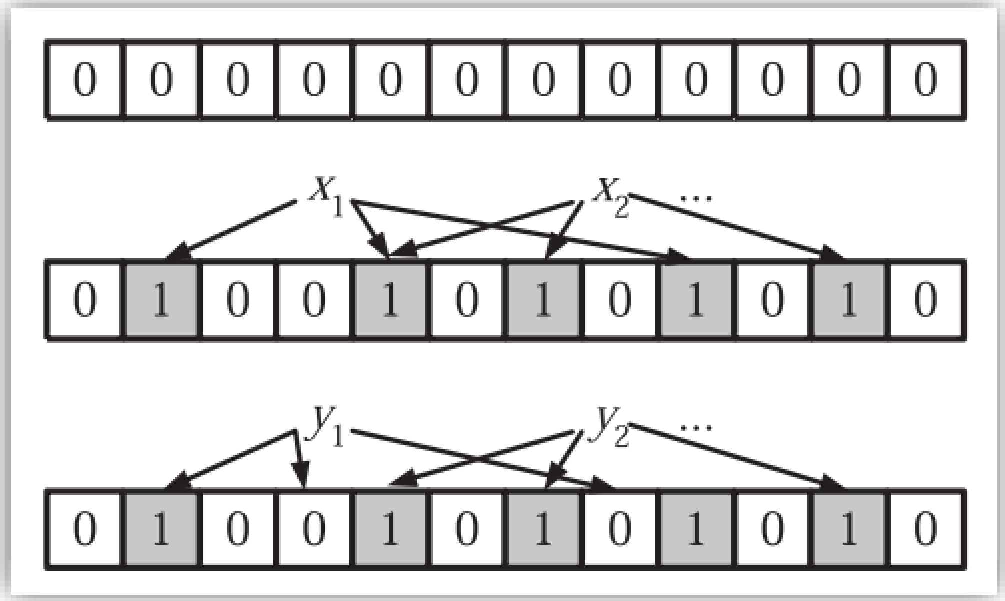
But the **actual** fraction of $0's$ in the hash table is a random variable $X_{k,N}$ with **expectation**

$$\mathbb{E}\left[X_{k,N}\right] = p(k,N)$$

To get the analysis right, we need a **concentration bound**: Want to say that $X_{k,N}$ is close to its expected value with **high probability**.

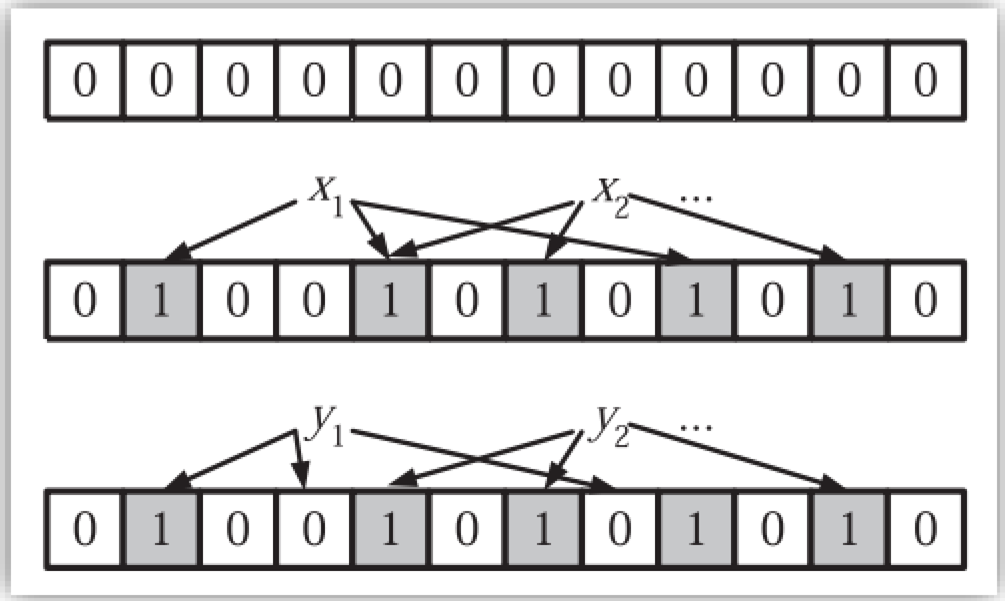
Let's analyze!

We have an array with M bits and to hash an element $x \in \mathcal{U}$, we set the bits in positions $h_1(x), h_2(x), \ldots, h_k(x)$ to 1.



We have an array with M bits and to hash an element $x \in \mathcal{U}$, we set the bits in positions $h_1(x), h_2(x), \dots, h_k(x)$ to 1.

Let X be the # of 0's in the hash table after N elements are hashed.

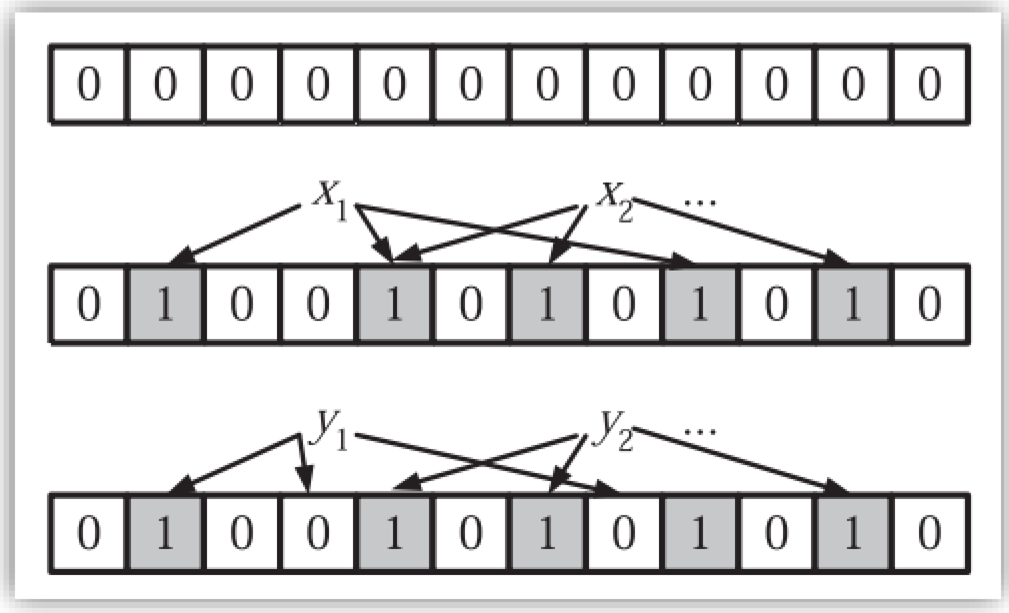


We have an array with M bits and to hash an element $x \in \mathcal{U}$, we set the bits in positions $h_1(x), h_2(x), \dots, h_k(x)$ to 1.

Let X be the # of 0's in the hash table after N elements are hashed.

Consider the N elements to hash:

$$x_1, x_2, \dots, x_N$$



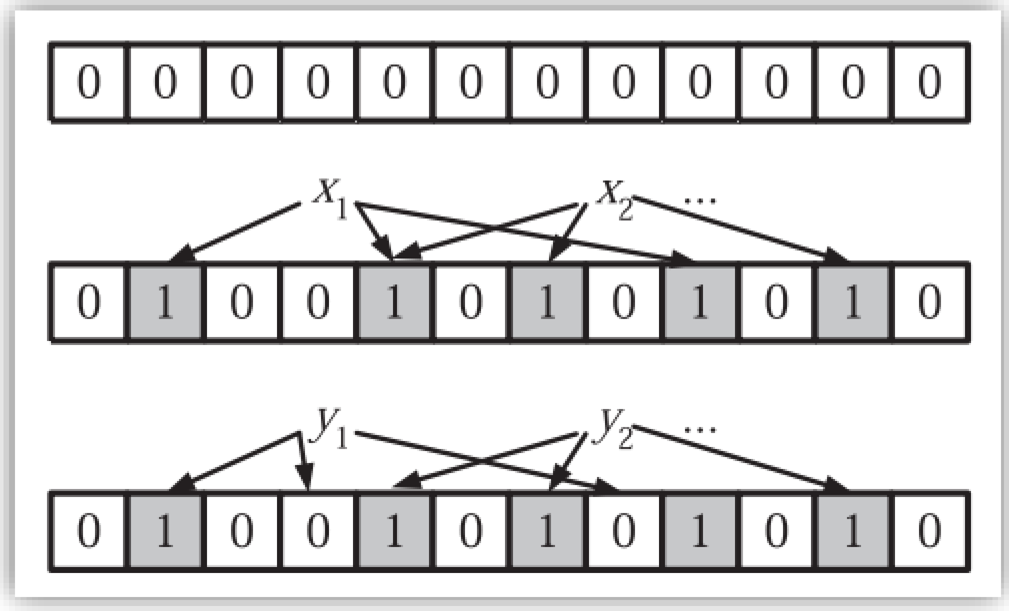
We have an array with M bits and to hash an element $x \in \mathcal{U}$, we set the bits in positions $h_1(x), h_2(x), \ldots, h_k(x)$ to 1.

Let X be the # of 0's in the hash table after N elements are hashed.

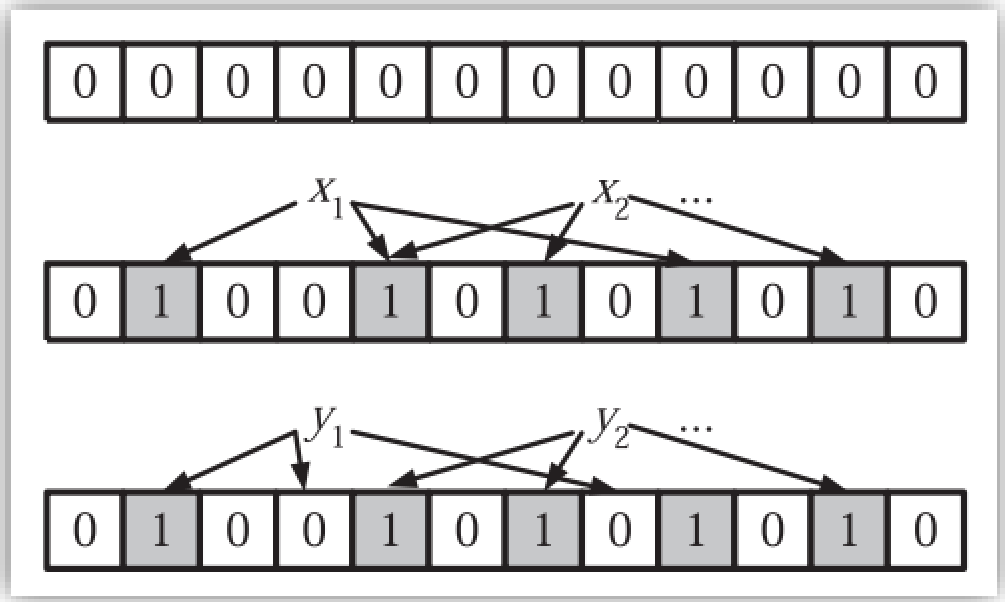
Consider the N elements to hash:

$$x_1, x_2, \ldots, x_N$$

Let $H(x_i) = (h_1(x_i), \ldots, h_k(x_i))$



We have an array with M bits and to hash an element $x \in \mathcal{U}$, we set the bits in positions $h_1(x), h_2(x), \ldots, h_k(x)$ to 1.



Let X be the # of 0's in the hash table after N elements are hashed.

Consider the N elements to hash:

$$x_1, x_2, \ldots, x_N$$

Let $H(x_i) = \left(h_1(x_i), \ldots, h_k(x_i)\right)$

Define $X_j = \mathbb{E}\left[X \mid H(x_1), H(x_2), \ldots, H(x_j)\right]$ to be the expected # of 0's in the hash table after hashing the first j elements.

We have an array with M bits and to hash an element $x \in \mathcal{U}$, we set the bits in positions $h_1(x), h_2(x), \ldots, h_k(x)$ to 1.

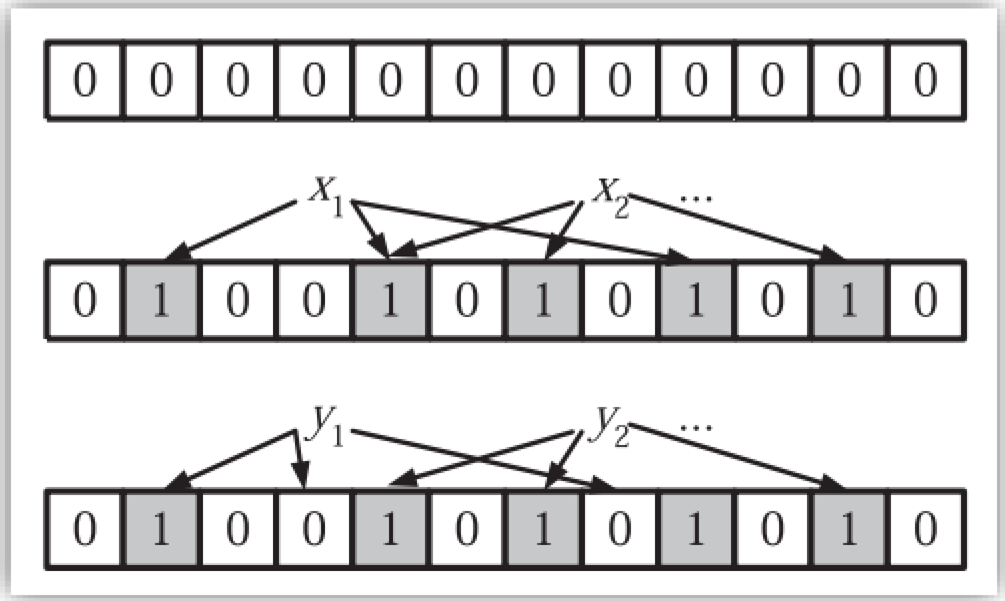
Let X be the # of 0's in the hash table after N elements are hashed.

Consider the N elements to hash:

$$x_1, x_2, \ldots, x_N$$

Let $H(x_i) = \left(h_1(x_i), \ldots, h_k(x_i)\right)$

Define $X_j = \mathbb{E}\left[X \mid H(x_1), H(x_2), \ldots, H(x_j)\right]$ to be the expected # of 0's in the hash table after hashing the first j elements.



Note that $x_1, \ldots, x_N$ are **any** set of keys. The randomness here is all in the choice of the hash functions $h_1, \ldots, h_k$.

Let X be the # of 0's in the hash table after N elements are hashed.

Consider the N elements to hash: $x_1, x_2, \ldots, x_N$

$$H(x_i) = \big(h_1(x_i), \ldots, h_k(x_i)\big)$$

$$X_j = \mathbb{E}\big[\, X \mid H(x_1), H(x_2), \ldots, H\big(x_j\big) \,\big]$$

We calculated before that $X_0 = \mathbb{E}[X] = m\left(1 - \frac{1}{m}\right)^{kN}$ [why?]

Let X be the # of 0's in the hash table after N elements are hashed.

Consider the N elements to hash: $x_1, x_2, \dots, x_N$

$$H(x_i) = \left(h_1(x_i), \dots, h_k(x_i)\right)$$

$$X_j = \mathbb{E}\left[X \mid H(x_1), H(x_2), \dots, H\left(x_j\right)\right]$$

We calculated before that $X_0 = \mathbb{E}[X] = m\left(1 - \frac{1}{m}\right)^{kN}$ [why?]

Now we want to know the probability that X is much different from its expectation X_0.

Let X be the # of 0's in the hash table after N elements are hashed.

Consider the N elements to hash: $x_1, x_2, \dots, x_N$

$$H(x_i) = \left(h_1(x_i), \dots, h_k(x_i)\right)$$

$$X_j = \mathbb{E}\left[X \mid H(x_1), H(x_2), \dots, H\left(x_j\right)\right]$$

We calculated before that $X_0 = \mathbb{E}[X] = m\left(1 - \frac{1}{m}\right)^{kN}$ [why?]

Now we want to know the probability that X is much different from its expectation X_0.

Claim #1: $\left|X_{j+1} - X_j\right| \leq k$ for all $j = 1,2, \dots, N-1$

Let X be the # of 0's in the hash table after N elements are hashed.

Consider the N elements to hash: $x_1, x_2, \ldots, x_N$

$$H(x_i) = \big(h_1(x_i), \ldots, h_k(x_i)\big)$$

$$X_j = \mathbb{E}\big[\, X \mid H(x_1), H(x_2), \ldots, H(x_j) \,\big]$$

We calculated before that $X_0 = \mathbb{E}[X] = m\left(1 - \frac{1}{m}\right)^{kN}$ [why?]

Now we want to know the probability that X is much different from its expectation X_0.

Claim #1: $\left|X_{j+1} - X_j\right| \le k$ for all $j = 1,2, \ldots, N-1$

Claim #2: $\mathbb{E}\big[\, X_{j+1} \mid H(x_1), \ldots, H(x_j) \,\big] = X_j$ for all $j = 1,2, \ldots, N-1$

Let X be the # of 0's in the hash table after N elements are hashed.

Consider the N elements to hash: $x_1, x_2, \dots, x_N$

$$H(x_i) = \left(h_1(x_i), \dots, h_k(x_i)\right)$$

$$X_j = \mathbb{E}\left[\, X \mid H(x_1), H(x_2), \dots, H\left(x_j\right) \right]$$

We calculated before that $X_0 = \mathbb{E}[X] = m\left(1 - \frac{1}{m}\right)^{kN}$ [why?]

Now we want to know the probability that X is much different from its expectation X_0.

Claim #1: $\left|X_{j+1} - X_j\right| \le k$ for all $j = 1,2,\dots,N-1$

Claim #2: $\mathbb{E}\left[\, X_{j+1} \mid H(x_1), \dots, H\left(x_j\right) \right] = X_j$ for all $j = 1,2,\dots,N-1$

Such a sequence of random variables is called a **martingale**

Suppose that $\{X_0, X_1, \dots, X_N\}$ is a **martingale** such that for some constants $\{c_j\}$, $|X_{j+1} - X_j| \le c_j$ for all $j = 0,1,\dots,N-1$. Then for any $\lambda > 0$,

$$\mathbb{P}[|X_N - X_0| \ge \lambda] \le 2\exp\left(-\frac{\lambda^2}{2\left(c_1^2 + \cdots + c_N^2\right)}\right)$$

Suppose that $\{X_0, X_1, \ldots, X_N\}$ is a **martingale** such that for some constants $\{c_j\}$, $|X_{j+1} - X_j| \le c_j$ for all $j = 0,1,\ldots,N-1$. Then for any $\lambda > 0$,

$$\mathbb{P}[|X_N - X_0| \ge \lambda] \le 2\exp\left(-\frac{\lambda^2}{2\left(c_1^2 + \cdots + c_N^2\right)}\right)$$

For our problem: $c_1 = c_2 = \cdots = c_N = k$

So the probability that the # of 0's differs from its expectation by more than λ is at most

$$2\exp(-\lambda^2/2k^2N)$$

So the deviation is $\approx k\sqrt{N}$ and is tightly concentrated in this window.

(take home) EXERCISE

Suppose that $\{X_0, X_1, \dots, X_N\}$ is a **martingale** such that for some constants $\{c_j\}$, $|X_{j+1} - X_j| \le c_j$ for all $j = 0,1,\dots,N-1$. Then for any $\lambda > 0$,

$$\mathbb{P}[|X_N - X_0| \ge \lambda] \le 2\exp\left(-\frac{\lambda^2}{2(c_1^2 + \cdots + c_N^2)}\right)$$

For our problem: $c_1 = c_2 = \cdots = c_N = k$

So the probability that the # of 0's differs from its expectation by more than λ is at most

$$2\exp(-\lambda^2/2k^2N)$$

So the deviation is $\approx k\sqrt{N}$ and is tightly concentrated in this window.

Improve the error probability to $2\exp(-\lambda^2/2kN)$ using a different martingale.