BBM402-Lecture 14: High Probability Analysis and Universal Hashing

Lecturer: Lale Özkahya

Resources for the presentation:
https://courses.engr.illinois.edu/cs473/fa2016/lectures.html
Outline

Randomized **QuickSort** w.h.p.
What is the probability that the algorithm will terminate in $O(n \log n)$ time?

**Balls & Bins**
- Expected bin size.
- Expected max bin size $\rightarrow$ max size w.h.p.
- Analogy to hashing

Hashing
Part I

Randomized **QuickSort** (Contd.)
Randomized **QuickSort**: Recall

**Input:** Array $A$ of $n$ distinct numbers. **Output:** Numbers in sorted order.

**Randomized QuickSort**

1. Pick a pivot element *uniformly at random* from $A$.
2. Split array into 2 subarrays: those smaller than pivot (L), and those larger than pivot (R).
3. Recursively sort the subarrays, and concatenate them.

Note: On every input randomized QuickSort takes $O(n \log n)$ time in expectation. On every input it may take $\Omega(n^2)$ time with some small probability.

Question: With what probability it takes $O(n \log n)$ time?
**Randomized QuickSort: Recall**

**Input:** Array $A$ of $n$ distinct numbers.  **Output:** Numbers in sorted order.

**Randomized QuickSort**

1. Pick a pivot element *uniformly at random* from $A$.
2. Split array into 2 subarrays: those smaller than pivot (L), and those larger than pivot (R).
3. Recursively sort the subarrays, and concatenate them.

**Note:** On every input randomized QuickSort takes $O(n \log n)$ time in expectation. On every input it may take $\Omega(n^2)$ time with some small probability.
Randomized **QuickSort**: Recall

**Input:** Array $A$ of $n$ distinct numbers. **Output:** Numbers in sorted order.

**Randomized QuickSort**

1. Pick a pivot element *uniformly at random* from $A$.
2. Split array into 2 subarrays: those smaller than pivot (L), and those larger than pivot (R).
3. Recursively sort the subarrays, and concatenate them.

**Note:** On every input randomized **QuickSort** takes $O(n \log n)$ time in expectation. On every input it may take $\Omega(n^2)$ time with some small probability.

Question: With what probability it takes $O(n \log n)$ time?
Randomized **QuickSort**: High Probability Analysis

**Informal Statement**

Random variable $Q(A) = \#$ comparisons done by the algorithm.

We will show that $\Pr[Q(A) \leq 32n \ln n] \geq 1 - \frac{1}{n^3}$.

If $n = 100$ then this gives $\Pr[Q(A) \leq 32n \ln n] \geq 0.99999$. 
Randomized **QuickSort**: High Probability Analysis

**Informal Statement**

Random variable $Q(A) = \#$ comparisons done by the algorithm.

We will show that $\Pr[Q(A) \leq 32n \ln n] \geq 1 - \frac{1}{n^3}$.

If $n = 100$ then this gives $\Pr[Q(A) \leq 32n \ln n] \geq 0.99999$. 
Randomized QuickSort: High Probability Analysis

Informal Statement
We will show that \(\Pr[Q(A) \leq 32n \ln n] \geq 1 - \frac{1}{n^3}\).

Outline of the proof
- If depth of recursion is \(k\) then \(Q(A) \leq kn\).
- Prove that depth of recursion \(\leq 32 \ln n\) with high probability (w.h.p.) . This will imply the result.
Informal Statement

We will show that \( \Pr[Q(A) \leq 32n \ln n] \geq 1 - \frac{1}{n^3} \).

Outline of the proof

- If depth of recursion is \( k \) then \( Q(A) \leq kn \).
- Prove that depth of recursion \( \leq 32 \ln n \) with high probability (w.h.p.) . This will imply the result.
  1. Focus on a single element. Prove that it “participates” in \( > 32 \ln n \) levels with probability (w.p.) at most \( \frac{1}{n^4} \).
  2. By union bound, any of the \( n \) elements participates in \( > 32 \ln n \) levels w.p. at most...
Randomized **QuickSort**: High Probability Analysis

**Informal Statement**
We will show that $\Pr[Q(A) \leq 32n \ln n] \geq 1 - \frac{1}{n^3}$.

**Outline of the proof**
- If depth of recursion is $k$ then $Q(A) \leq kn$.
- Prove that depth of recursion $\leq 32 \ln n$ with high probability (w.h.p.) . This will imply the result.
  1. Focus on a single element. Prove that it “participates” in $> 32 \ln n$ levels with probability (w.p.) at most $\frac{1}{n^4}$.
  2. By union bound, any of the $n$ elements participates in $> 32 \ln n$ levels w.p. at most $\frac{1}{n^3}$.
Informal Statement

We will show that $\Pr[Q(A) \leq 32n \ln n] \geq 1 - \frac{1}{n^3}$.

Outline of the proof

- If depth of recursion is $k$ then $Q(A) \leq kn$.
- Prove that depth of recursion $\leq 32 \ln n$ with high probability (w.h.p.) . This will imply the result.
  1. Focus on a single element. Prove that it “participates” in $> 32 \ln n$ levels with probability (w.p.) at most $\frac{1}{n^4}$.
  2. By union bound, any of the $n$ elements participates in $> 32 \ln n$ levels w.p. at most $\frac{1}{n^3}$.
  3. Therefore, all elements participate in $\leq 32 \ln n$ w.p. $\left(1 - \frac{1}{n^3}\right)$.
Randomized **QuickSort**: High Probability Analysis

**Informal Statement**
An element participates in $> 32 \ln n$ w.p. $\leq 1/n^4$.

**Intuition**

1. When we pick a pivot from an array of size $n$ uniformly at random, what is the probability that its rank is between $n/4$ and $3n/4$?
Informal Statement

An element participates in $> 32 \ln n$ w.p. $\leq 1/n^4$.

Intuition

When we pick a pivot from an array of size $n$ uniformly at random, what is the probability that its rank is between $n/4$ and $3n/4$? $1/2$. 

If $32 \ln n$ splits, then $E[\text{Balanced-split}] = 16 \ln n$. Out of these there are $<4 \ln n$ balanced split w.p. $\leq 1/n^4$. 

Chandra & Ruta (UIUC) CS473 Fall 2016 8 / 41
Randomized **QuickSort**: High Probability Analysis

### Informal Statement
An element participates in $> 32 \ln n$ w.p. $\leq 1/n^4$.

### Intuition
1. When we pick a pivot from an array of size $n$ uniformly at random, what is the probability that its rank is between $n/4$ and $3n/4$? $1/2$.

2. If we pick such a pivot then the size of $L$ and $R$ is at most?
Informal Statement

An element participates in $> 32 \ln n$ w.p. $\leq 1/n^4$.

Intuition

1. When we pick a pivot from an array of size $n$ uniformly at random, what is the probability that its rank is between $n/4$ and $3n/4$? $1/2$.

2. If we pick such a pivot then the size of $L$ and $R$ is at most? $3n/4$. (Balanced split)
Randomized **QuickSort**: High Probability Analysis

**Informal Statement**

An element participates in $> 32 \ln n$ w.p. $\leq 1/n^4$.

**Intuition**

1. When we pick a pivot from an array of size $n$ uniformly at random, what is the probability that its rank is between $n/4$ and $3n/4$? $1/2$.

2. If we pick such a pivot then the size of $L$ and $R$ is at most? $3n/4$. (Balanced split)

3. If an array is reduced to at least its $3/4$th size every time, then after how many rounds only one element remains?
Randomized **QuickSort**: High Probability Analysis

**Informal Statement**

An element participates in $> 32 \ln n$ w.p. $\leq 1/n^4$.

**Intuition**

1. When we pick a pivot from an array of size $n$ uniformly at random, what is the probability that its rank is between $n/4$ and $3n/4$? $1/2$.

2. If we pick such a pivot then the size of $L$ and $R$ is at most? $3n/4$. (Balanced split)

3. If an array is reduced to at least its $3/4$th size every time, then after how many rounds only one element remains? $\leq 4 \ln n$. 
Randomized **QuickSort**: High Probability Analysis

**Informal Statement**

An element participates in $> 32 \ln n$ w.p. $\leq 1/n^4$.

**Intuition**

1. When we pick a pivot from an array of size $n$ uniformly at random, what is the probability that its rank is between $n/4$ and $3n/4$? $1/2$.

2. If we pick such a pivot then the size of $L$ and $R$ is at most? $3n/4$. (Balanced split)

3. If an array is reduced to at least its $3/4$th size every time, then after how many rounds only one element remains? $\leq 4 \ln n$.

4. If $32 \ln n$ splits, then $E[\text{Balanced-split}] = 16 \ln n$. Out of these there are $< 4 \ln n$ balanced split w.p. $\leq 1/n^4$. 

Chandra & Ruta (UIUC)  
CS473  
Fall 2016
Randomized **QuickSort**: High Probability Analysis

- If \( k \) levels of recursion then \( kn \) comparisons.

Fix an element \( s \in A \). We will track it at each level. Let \( S_i \) be the partition containing \( s \) at \( i \)th level. \( S_1 = A \) and \( S_k = \{s\} \).

We call \( s \) lucky in \( i \)th iteration, if balanced split:

\[
|S_{i+1}| \leq \left(\frac{3}{4}\right)|S_i|
\]

and

\[
|S_i \setminus S_{i+1}| \leq \left(\frac{3}{4}\right)|S_i|
\]

If \( \rho = \) \#lucky rounds in first \( k \) rounds, then

\[
|S_k| \leq \left(\frac{3}{4}\right)\rho n.
\]

For \( |S_k| = 1 \), \( \rho = 4 \ln n \geq \log_{\frac{4}{3}} n \) suffices.
If $k$ levels of recursion then $kn$ comparisons.

Fix an element $s \in A$. We will track it at each level.

Let $S_i$ be the partition containing $s$ at $i^{th}$ level.

$S_1 = A$ and $S_k = \{s\}$. 
Randomized QuickSort: High Probability Analysis

- If $k$ levels of recursion then $kn$ comparisons.
- Fix an element $s \in A$. We will track it at each level.
- Let $S_i$ be the partition containing $s$ at $i^{th}$ level.
- $S_1 = A$ and $S_k = \{s\}$.

We call $s$ lucky in $i^{th}$ iteration, if balanced split:

$|S_{i+1}| \leq \frac{3}{4}|S_i|$ and $|S_i \setminus S_{i+1}| \leq \frac{3}{4}|S_i|$. 
Randomized **QuickSort**: High Probability Analysis

- If **k** levels of recursion then **kn** comparisons.
- Fix an element **s ∈ A**. We will track it at each level.
- Let **S_i** be the partition containing **s** at **i^{th}** level.
- **S_1 = A** and **S_k = {s}**.

We call **s** lucky in **i^{th}** iteration, if *balanced split*:

\[ |S_{i+1}| \leq (\frac{3}{4})|S_i| \text{ and } |S_i \setminus S_{i+1}| \leq (\frac{3}{4})|S_i| \]

- If **ρ = #lucky rounds in first **k** rounds**, then

\[ |S_k| \leq (\frac{3}{4})^{\rho n} \]
Randomized QuickSort: High Probability Analysis

- If \( k \) levels of recursion then \( kn \) comparisons.
- Fix an element \( s \in A \). We will track it at each level.
- Let \( S_i \) be the partition containing \( s \) at \( i^{th} \) level.
- \( S_1 = A \) and \( S_k = \{s\} \).

We call \( s \) lucky in \( i^{th} \) iteration, if balanced split:
\[
|S_{i+1}| \leq \frac{3}{4}|S_i| \quad \text{and} \quad |S_i \setminus S_{i+1}| \leq \frac{3}{4}|S_i|.
\]

- If \( \rho = \# \)lucky rounds in first \( k \) rounds, then
\[
|S_k| \leq \left(\frac{3}{4}\right)^\rho n.
\]

- For \( |S_k| = 1 \), \( \rho = 4 \ln n \geq \log_{4/3} n \) suffices.
How may rounds before $4 \ln n$ lucky rounds?

- $X_i = 1$ if $s$ is lucky in $i^{th}$ iteration.
How may rounds before $4 \ln n$ lucky rounds?

- $X_i = 1$ if $s$ is lucky in $i^{th}$ iteration.
- Observation: $X_1, \ldots, X_k$ are independent variables.
- $\Pr[X_i = 1] = \frac{1}{2}$ Why?
How may rounds before $4 \ln n$ lucky rounds?

- $X_i = 1$ if $s$ is lucky in $i^{th}$ iteration.
- **Observation:** $X_1, \ldots, X_k$ are independent variables.
- $\Pr[X_i = 1] = \frac{1}{2}$ **Why?**
- Clearly, $\rho = \sum_{i=1}^{k} X_i$. Let $\mu = \mathbb{E}[\rho] = \frac{k}{2}$. 

$\text{(Chernoff)} \leq e^{-\frac{\delta^2 \mu}{2}} = e^{-\frac{9k}{64}} = e^{-4.5 \ln n} \leq \frac{1}{n^4}$
How may rounds before $4 \ln n$ lucky rounds?

- $X_i = 1$ if $s$ is lucky in $i^{th}$ iteration.
- **Observation:** $X_1, \ldots, X_k$ are independent variables.
- $\Pr[X_i = 1] = \frac{1}{2}$  Why?
- Clearly, $\rho = \sum_{i=1}^{k} X_i$. Let $\mu = \mathbb{E}[\rho] = \frac{k}{2}$.
- Set $k = 32 \ln n$ and $\delta = \frac{3}{4}$. $(1 - \delta) = \frac{1}{4}$.
How may rounds before $4 \ln n$ lucky rounds?

- $X_i = 1$ if $s$ is lucky in $i^{th}$ iteration.
- **Observation:** $X_1, \ldots, X_k$ are independent variables.
- $\Pr[X_i = 1] = \frac{1}{2}$ Why?
- Clearly, $\rho = \sum_{i=1}^{k} X_i$. Let $\mu = \mathbb{E}[\rho] = \frac{k}{2}$.
- Set $k = 32 \ln n$ and $\delta = \frac{3}{4}$. $(1 - \delta) = \frac{1}{4}$.

Probability of $\leq 4 \ln n$ lucky rounds out of $32 \ln n$ rounds is,
How may rounds before $4 \ln n$ lucky rounds?

- $X_i = 1$ if $s$ is lucky in $i^{\text{th}}$ iteration.
- **Observation:** $X_1, \ldots, X_k$ are independent variables.
- $\Pr[X_i = 1] = \frac{1}{2}$  Why?

Clearly, $\rho = \sum_{i=1}^{k} X_i$. Let $\mu = E[\rho] = \frac{k}{2}$.

- Set $k = 32 \ln n$ and $\delta = \frac{3}{4}$. $(1 - \delta) = \frac{1}{4}$.

Probability of $\leq 4 \ln n$ lucky rounds out of $32 \ln n$ rounds is,

$$
\Pr[\rho \leq 4 \ln n] = \Pr[\rho \leq \frac{k}{8}]
= \Pr[\rho \leq (1 - \delta)\mu]
$$
How may rounds before $4 \ln n$ lucky rounds?

- $X_i = 1$ if $s$ is lucky in $i^{th}$ iteration.
- Observation: $X_1, \ldots, X_k$ are independent variables.
- $\Pr[X_i = 1] = \frac{1}{2}$  
  Why?
- Clearly, $\rho = \sum_{i=1}^{k} X_i$. Let $\mu = \mathbb{E}[\rho] = \frac{k}{2}$.
- Set $k = 32 \ln n$ and $\delta = \frac{3}{4}$. $(1 - \delta) = \frac{1}{4}$.

Probability of $\leq 4 \ln n$ lucky rounds out of $32 \ln n$ rounds is,

$$\Pr[\rho \leq 4 \ln n] = \Pr[\rho \leq \frac{k}{8}] = \Pr[\rho \leq (1 - \delta)\mu]$$

(Chebyshev)

$$\leq e^{-\frac{\delta^2 \mu}{2}} = e^{-\frac{9k}{64}} = e^{-4.5 \ln n} \leq \frac{1}{n^4}$$
Randomized **QuickSort** w.h.p. Analysis

- n input elements. Probability that depth of recursion in **QuickSort** $> 32 \ln n$ is at most $\frac{1}{n^4} \ast n = \frac{1}{n^3}$. 

Q: How to increase the probability?
Randomized **QuickSort** w.h.p. Analysis

- \( n \) input elements. Probability that depth of recursion in **QuickSort** \( > 32 \ln n \) is at most \( \frac{1}{n^4} \times n = \frac{1}{n^3} \).

**Theorem**

*With high probability (i.e., \( 1 - \frac{1}{n^3} \)) the depth of the recursion of **QuickSort** is \( \leq 32 \ln n \). Due to \( n \) comparisons in each level, with high probability, the running time of **QuickSort** is \( O(n \ln n) \).*
Randomized QuickSort w.h.p. Analysis

- $n$ input elements. Probability that depth of recursion in QuickSort $> 32 \ln n$ is at most $\frac{1}{n^4} \times n = \frac{1}{n^3}$.

**Theorem**

*With high probability (i.e., $1 - \frac{1}{n^3}$) the depth of the recursion of QuickSort is $\leq 32 \ln n$. Due to $n$ comparisons in each level, with high probability, the running time of QuickSort is $O(n \ln n)$.**

Q: How to increase the probability?
Part II

Balls and Bins
Expected Bin Size

Problem
If $n$ balls are thrown independently and uniformly into $n$ bins, how many balls lend in a bin in expectation (expected size of a bin)?
Expected Bin Size

Problem
If \( n \) balls are thrown independently and uniformly into \( n \) bins, how many balls lend in a bin in expectation (expected size of a bin)?

Solution
- Fix a bin, say \( j \).
Expected Bin Size

Problem
If $n$ balls are thrown independently and uniformly into $n$ bins, how many balls lend in a bin in expectation (expected size of a bin)?

Solution
- Fix a bin, say $j$.
- Random variable $X_{ij}$ is 1 if $i$th balls falls in $j$th bin, otherwise 0.

$$E[Y_j] = \sum_{i=1}^{n} E[X_{ij}] = n \cdot \frac{1}{n} = 1.$$
Expected Bin Size

Problem
If \( n \) balls are thrown independently and uniformly into \( n \) bins, how many balls lend in a bin in expectation (expected size of a bin)?

Solution
- Fix a bin, say \( j \).
- Random variable \( X_{ij} \) is 1 if \( i \)th balls falls in \( j \)th bin, otherwise 0.
- \( \mathbb{E}[X_{ij}] = \Pr[X_{ij} = 1] = \)
Expected Bin Size

Problem
If $n$ balls are thrown independently and uniformly into $n$ bins, how many balls lend in a bin in expectation (expected size of a bin)?

Solution
- Fix a bin, say $j$.
- Random variable $X_{ij}$ is 1 if $i$th balls falls in $j$th bin, otherwise 0.
- $E[X_{ij}] = \Pr[X_{ij} = 1] = 1/n$. 

$Y_j = \text{# balls in } j\text{th bin} = \sum_{n}^{i=1} X_{ij}$.

$E[Y_j] = \sum_{n}^{i=1} E[X_{ij}] = n \cdot \frac{1}{n} = 1$. 
Expected Bin Size

**Problem**

If \( n \) balls are thrown independently and uniformly into \( n \) bins, how many balls lend in a bin in expectation (expected size of a bin)?

**Solution**

- Fix a bin, say \( j \).
- Random variable \( X_{ij} \) is \( 1 \) if \( i \)th balls falls in \( j \)th bin, otherwise \( 0 \).
- \( E[X_{ij}] = Pr[X_{ij} = 1] = \frac{1}{n} \).
- R.V. \( Y_j = \# \) balls in \( j \)th bin = \( \sum_{i=1}^{n} X_{ij} \).
Expected Bin Size

Problem
If \( n \) balls are thrown independently and uniformly into \( n \) bins, how many balls lend in a bin in expectation (expected size of a bin)?

Solution
- Fix a bin, say \( j \).
- Random variable \( X_{ij} \) is 1 if \( i \)th balls falls in \( j \)th bin, otherwise 0.
- \( E[X_{ij}] = \Pr[X_{ij} = 1] = \frac{1}{n} \).
- R.V. \( Y_j = \# \) balls in \( j \)th bin = \( \sum_{i=1}^{n} X_{ij} \).
- \( E[Y_j] = \sum_{i=1}^{n} E[X_{ij}] = n \cdot \frac{1}{n} = 1 \).
Expected Max Bin Size

Problem
If \( n \) balls are thrown independently and uniformly into \( n \) bins, what is the expected maximum bin size?

Possible Solution
R.V. \( Z = \max_{1 \leq j \leq n} Y_j \).

\[ E[Z] = \sum_{k=1}^{n} \Pr[Z = k] k. \]

How to compute \( \Pr[Z = k] \), i.e., count configurations where no bin has more than \( k \) balls and at least one has \( k \) balls.

Too many to count!!
Expected Max Bin Size

Problem
If $n$ balls are thrown independently and uniformly into $n$ bins, what is the expected maximum bin size?

$$E \left[ \max_{j=1}^{n} Y_j \right]$$
Expected Max Bin Size

**Problem**

If $n$ balls are thrown independently and uniformly into $n$ bins, what is the expected maximum bin size?

$$E \left[ \max_{j=1}^{n} Y_j \right]$$

**Possible Solution**

- R.V. $Z = \max_{j=1}^{n} Y_j$. $E[Z] = \sum_{k=1}^{n} \Pr[Z = k] k$. 

How to compute $\Pr[Z = k]$, i.e., count configurations where no bin has more than $k$ balls and at least one has $k$ balls. Too many to count!!
Problem
If \( n \) balls are thrown independently and uniformly into \( n \) bins, what is the expected maximum bin size?

\[
E \left[ \max_{j=1}^{n} Y_j \right]
\]

Possible Solution
- R.V. \( Z = \max_{j=1}^{n} Y_j \). \( E[Z] = \sum_{k=1}^{n} \Pr[Z = k] k \).
- How to compute \( \Pr[Z = k] \), i.e., count configurations where no bin has more than \( k \) balls and at least one has \( k \) balls.
Expected Max Bin Size

Problem
If \( n \) balls are thrown independently and uniformly into \( n \) bins, what is the expected maximum bin size?

\[
E \left[ \max_{j=1}^{n} Y_j \right]
\]

Possible Solution
- R.V. \( Z = \max_{j=1}^{n} Y_j \). \( E[Z] = \sum_{k=1}^{n} \Pr[Z = k] k \).
- How to compute \( \Pr[Z = k] \), i.e., count configurations where no bin has more than \( k \) balls and at least one has \( k \) balls.
- Too many to count!!
Problem
What is the expected maximum bin size?
R.V. $Z = \max_{j=1}^{n} Y_j$. Show $E[Z] \leq O\left(\frac{\ln n}{\ln \ln n}\right)$?

Possible Solution
If $\Pr[Z > \frac{8\ln n}{\ln \ln n}] \leq 1/n^2$, then: define $A = \frac{8\ln n}{\ln \ln n}$.
Problem

What is the expected maximum bin size?

R.V. $Z = \max_{j=1}^{n} Y_j$. Show $E[Z] \leq O\left(\frac{\ln n}{\ln \ln n}\right)$?

Possible Solution

If $\Pr[Z > \frac{8\ln n}{\ln \ln n}] \leq \frac{1}{n^2}$, then: define $A = \frac{8\ln n}{\ln \ln n}$.

\[
E[Z] \leq \sum_{k=1}^{A} Pr[Z = k] A + \sum_{k=A+1}^{n} Pr[Z = k] n \\
\leq A \cdot Pr[Z \leq A] + n \cdot Pr[Z > A] \\
\leq A \cdot (1) + n \cdot (1/n^2) = O(A) = O\left(\frac{\ln n}{\ln \ln n}\right)
\]
Expected Max Bin Size (Contd.)

Problem
What is the expected maximum bin size?

R.V. \( Z = \max_{j=1}^{n} Y_j \). Show \( E[Z] \leq O \left( \frac{\ln n}{\ln \ln n} \right) \).

Possible Solution

If \( \Pr \left[ Z > \frac{8 \ln n}{\ln \ln n} \right] \leq \frac{1}{n^2} \), then: define \( A = \frac{8 \ln n}{\ln \ln n} \).

\[
E[Z] \leq \sum_{k=1}^{A} \Pr[Z = k] A + \sum_{k=A+1}^{n} \Pr[Z = k] n \\
\leq A \cdot \Pr[Z \leq A] + n \cdot \Pr[Z > A] \\
\leq A \cdot (1) + n \cdot (1/n^2) = O(A) = O \left( \frac{\ln n}{\ln \ln n} \right)
\]

Bound \( \Pr[Z > \frac{8 \ln n}{\ln \ln n}] \).
Bound $\Pr[Z > \frac{8 \ln n}{\ln \ln n}]$ using Chernoff inequality.

**Chernoff Ineq. We Saw**

Let $X_1, \ldots, X_k$ be independent binary random variables, and let $X = \sum_{i=1}^{k} X_i$, $\mu = \mathbb{E}[X]$, then for $0 < \delta < 1$

$$
\Pr[X \geq (1 + \delta)\mu] \leq e^{-\delta^2 \mu/3} \quad \& \quad \Pr[X \leq (1 - \delta)\mu] \leq e^{-\delta^2 \mu/2}
$$
**Expected Max Bin Size (Contd.)**

Bound $\Pr[Z > \frac{8 \ln n}{\ln \ln n}]$ using Chernoff inequality.

**Chernoff Ineq. We Saw**

$X_1, \ldots, X_k$ independent binary R.V., and $X = \sum_{i=1}^{k} X_i$, $\mu = \mathbb{E}[X]$, then for $0 < \delta < 1$

$$\Pr[X \geq (1 + \delta)\mu] \leq e^{-\delta^2 \mu / 3} \quad \& \quad \Pr[X \leq (1 - \delta)\mu] \leq e^{-\delta^2 \mu / 2}$$

**Stronger Versions**

- For $\delta > 0$, $\Pr[X > (1 + \delta)\mu] < \left( \frac{e^{\delta}}{(1+\delta)(1+\delta)} \right)^{\mu}$.
- For $0 < \delta < 1$ $\Pr[X < (1 - \delta)\mu] < \left( \frac{e^{-\delta}}{(1-\delta)(1-\delta)} \right)^{\mu}$. 
Expected Max Bin Size (Contd.)

Problem
What is the expected maximum bin size? Let $Z = \max_{j=1}^{n} Y_j$.
Show $E[Z] \leq O(\frac{\ln n}{\ln \ln n})$.
Show $Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq \frac{1}{n^2}$.
Problem
What is the expected maximum bin size? Let $Z = \max_{j=1}^n Y_j$.
Show $E[Z] \leq O\left(\frac{\ln n}{\ln \ln n}\right)$. → Show $\Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq 1/n^2$.

Solution
Recall: $Y_j = \#$ balls in bin $j$, $E[Y_j] = 1$, and $A = \frac{8 \ln n}{\ln \ln n}$

$$
\Pr[Y_j > A] = \Pr[Y_j \geq A \cdot E[Y]] < \left(\frac{e^{A-1}}{A^A}\right) < \left(\frac{n^6/\ln \ln n}{A^A}\right)
$$
Expected Max Bin Size (Contd.)

Problem
What is the expected maximum bin size? Let $Z = \max_{j=1}^n Y_j$.
Show $E[Z] \leq O\left(\frac{\ln n}{\ln \ln n}\right)$.

Show $\Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq \frac{1}{n^2}$.

Solution
Recall: $Y_j = \#$ balls in bin $j$, $E[Y_j] = 1$, and $A = \frac{8 \ln n}{\ln \ln n}$

$$\Pr[Y_j > A] = \Pr[Y_j \geq A \cdot E[Y]] < \left(\frac{e^{A-1}}{A^A}\right) < \left(\frac{n^6/\ln \ln n}{A^A}\right)$$

$$A^A = \left(\frac{8 \ln n}{\ln \ln n}\right)^{8 \ln n/\ln \ln n} \geq (\sqrt{\ln n})^{8 \ln n/\ln \ln n} = (\ln n)^{4 \ln n/\ln \ln n} = e^{4\ln n} = n^4$$
Expected Max Bin Size (Contd.)

**Problem**

What is the expected maximum bin size? Let \( Z = \max_{j=1}^{n} Y_j \).

Show \( E[Z] \leq O\left(\frac{\ln n}{\ln \ln n}\right) \).

Show \( \Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq 1/n^2 \).

**Solution**

Recall: \( Y_j = \# \text{ balls in bin } j, \ E[Y_j] = 1, \) and \( A = \frac{8 \ln n}{\ln \ln n} \)

\[
\Pr[Y_j > A] = \Pr[Y_j \geq A E[Y]] < \left(\frac{e^{A-1}}{A^A}\right) < \left(\frac{n^6/\ln \ln n}{A^A}\right)
\]

\[
A^A = \left(\frac{8 \ln n}{\ln \ln n}\right)^{\frac{8 \ln n}{\ln \ln n}} \geq (\sqrt{\ln n})^{\frac{8 \ln n}{\ln \ln n}} = (\ln n)^{\frac{4 \ln n}{\ln \ln n}} = e^{4\ln n} = n^4
\]

\[
\Pr \left[ Y_j > \frac{8 \ln n}{\ln \ln n} \right] < 1/n^3
\]
**Problem**

What is the expected maximum bin size? Let $Z = \max_{j=1}^{n} Y_j$. Show $E[Z] \leq O\left(\frac{\ln n}{\ln \ln n}\right)$. → Show $\Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq 1/n^2$.

**Solution**


$\Pr[Y_j > \frac{8 \ln n}{\ln \ln n}] \leq 1/n^3$ (Using Chernoff)
Expected Max Bin Size (Contd.)

**Problem**

What is the expected maximum bin size? Let $Z = \max_{j=1}^{n} Y_j$.

Show $E[Z] \leq O\left(\frac{\ln n}{\ln \ln n}\right)$. → Show $\Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq \frac{1}{n^2}$.

**Solution**

Recall: $Y_j = \# \text{ balls in bin } j$. $E[Y_j] = 1$.

$\Pr[Y_j > \frac{8 \ln n}{\ln \ln n}] \leq \frac{1}{n^3}$ (Using Chernoff)

(Union bound)

$\Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq \sum_{j=1}^{n} \Pr[Y_j > \frac{8 \ln n}{\ln \ln n}] \leq n \cdot \frac{1}{n^3} = \frac{1}{n^2}$. 

Max bin size is at most $O\left(\frac{\ln n}{\ln \ln n}\right)$ with probability $1 - \frac{1}{n^2}$.

Ω(\ln n \ln \ln n) is a lower bound as well!

Chandra & Ruta (UIUC) CS473 18 Fall 2016 18 / 41
Problem

What is the expected maximum bin size? Let \( Z = \max_{j=1}^{n} Y_j \).
Show \( E[Z] \leq O\left(\frac{\ln n}{\ln \ln n}\right) \). → Show \( \Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq \frac{1}{n^2} \).

Solution

Recall: \( Y_j = \# \) balls in bin \( j \). \( E[Y_j] = 1 \).
\[ \Pr[Y_j > \frac{8 \ln n}{\ln \ln n}] \leq \frac{1}{n^3} \quad \text{(Using Chernoff)} \]

(Union bound)
\[ \Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq \sum_{j=1}^{n} \Pr[Y_j > \frac{8 \ln n}{\ln \ln n}] \leq n \cdot \frac{1}{n^3} = \frac{1}{n^2}. \]

Max bin size is at most \( O\left(\frac{\ln n}{\ln \ln n}\right) \) with probability \( 1 - \frac{1}{n^2} \).
Problem

What is the expected maximum bin size? Let $Z = \max_{j=1}^n Y_j$.

Show $E[Z] \leq O(\frac{\ln n}{\ln \ln n})$.

→ Show $\Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq 1/n^2$.

Solution


$\Pr[Y_j > 8 \ln n/ \ln \ln n] \leq 1/n^3$ (Using Chernoff)

(Union bound)

$\Pr[Z > \frac{8 \ln n}{\ln \ln n}] \leq \sum_{j=1}^n \Pr[Y_j > \frac{8 \ln n}{\ln \ln n}] \leq n \cdot 1/n^3 = 1/n^2$.

Max bin size is at most $O(\frac{\ln n}{\ln \ln n})$ with probability $1 - 1/n^2$.

$\Omega(\frac{\ln n}{\ln \ln n})$ is a lower bound as well!
Balls n Bins → Hashing

Hashing

Storing elements in a table such that look up is $O(1)$-time.
Hashing

Storing elements in a table such that look up is $O(1)$-time.

Throwing numbered balls

Imagine that $n$ balls have numbers coming from a universe $\mathcal{U}$. $|\mathcal{U}| \gg n$. 
Hashing
Storing elements in a table such that look up is \( O(1) \)-time.

Throwing numbered balls
Imagine that \( n \) balls have numbers coming from a universe \( U \).
\(|U| \gg n\).

Hashing: throw balls (elements) randomly into \( n \) bins such that bin sizes are small.
Balls n Bins → Hashing

Hashing

Storing elements in a table such that look up is $O(1)$-time.

Throwing numbered balls

Imagine that $n$ balls have numbers coming from a universe $U$. $|U| \gg n$.

Hashing: throw balls (elements) randomly into $n$ bins such that bin sizes are small and also lookup is easy!
Part III

Hash Tables
Dictionary Data Structure

1. **U**: universe of keys with total order: numbers, strings, etc.
2. Data structure to store a subset $S \subseteq U$
3. **Operations:**
   1. **Search/lookup**: given $x \in U$ is $x \in S$?
   2. **Insert**: given $x \not\in S$ add $x$ to $S$.
   3. **Delete**: given $x \in S$ delete $x$ from $S$
4. **Static** structure: $S$ given in advance or changes very infrequently, main operations are lookups.
5. **Dynamic** structure: $S$ changes rapidly so inserts and deletes as important as lookups.
Dictionary Data Structures

Common solutions:

1. Static:
   1. Store $S$ as a *sorted* array
   2. **Lookup**: Binary search in $O(\log |S|)$ time (comparisons)

2. Dynamic:
   1. Store $S$ in a *balanced* binary search tree
   2. Lookup, Insert, Delete in $O(\log |S|)$ time (comparisons)
Question: “Should Tables be Sorted?”
(also title of famous paper by Turing award winner Andy Yao)
Dictionary Data Structures

Question: “Should Tables be Sorted?”
(also title of famous paper by Turing award winner Andy Yao)

Hashing is a widely used & powerful technique for dictionaries.

Motivation:

1. Universe $U$ may not be (naturally) totally ordered.
2. Keys correspond to large objects (images, graphs etc) for which comparisons are very expensive.
3. Want to improve “average” performance of lookups to $O(1)$ even at cost of extra space or errors with small probability: many applications for fast lookups in networking, security, etc.
Hashing and Hash Tables

Hash Table data structure:

1. A (hash) table/array $T$ of size $m$ (the table size).
2. A hash function $h : \mathcal{U} \rightarrow \{0, \ldots, m - 1\}$.
3. Item $x \in \mathcal{U}$ hashes to slot $h(x)$ in $T$. 

Given $S \subseteq \mathcal{U}$. How do we store $S$ and how do we do lookups?

Ideal situation:

1. Each element $x \in S$ hashes to a distinct slot in $T$. Store $x$ in slot $h(x)$.
2. Lookup: Given $y \in \mathcal{U}$ check if $T[h(y)] = y$. $O(1)$ time!

Collisions unavoidable if $|T| < |\mathcal{U}|$. Several techniques to handle them.

Chandra & Ruta (UIUC)
Hashing and Hash Tables

Hash Table data structure:
1. A (hash) table/array $T$ of size $m$ (the table size).
2. A hash function $h : \mathcal{U} \rightarrow \{0, \ldots, m - 1\}$.
3. Item $x \in \mathcal{U}$ hashes to slot $h(x)$ in $T$.

Given $S \subseteq \mathcal{U}$. How do we store $S$ and how do we do lookups?
Hashing and Hash Tables

Hash Table data structure:
1. A (hash) table/array $T$ of size $m$ (the table size).
2. A hash function $h : U \rightarrow \{0, \ldots, m - 1\}$.
3. Item $x \in U$ hashes to slot $h(x)$ in $T$.

Given $S \subseteq U$. How do we store $S$ and how do we do lookups?

**Ideal situation:**
1. Each element $x \in S$ hashes to a distinct slot in $T$. Store $x$ in slot $h(x)$
2. Lookup: Given $y \in U$ check if $T[h(y)] = y$. $O(1)$ time!
Hashing and Hash Tables

Hash Table data structure:
1. A (hash) table/array $T$ of size $m$ (the table size).
2. A hash function $h : \mathcal{U} \rightarrow \{0, \ldots, m - 1\}$.
3. Item $x \in \mathcal{U}$ hashes to slot $h(x)$ in $T$.

Given $S \subseteq \mathcal{U}$. How do we store $S$ and how do we do lookups?

**Ideal situation:**
1. Each element $x \in S$ hashes to a distinct slot in $T$. Store $x$ in slot $h(x)$
2. **Lookup:** Given $y \in \mathcal{U}$ check if $T[h(y)] = y$. $O(1)$ time!

Collisions unavoidable if $|T| < |\mathcal{U}|$. Several techniques to handle them.
Handling Collisions: Chaining

Collision: \( h(x) = h(y) \) for some \( x \neq y \).

Chaining to handle collisions:

1. For each slot \( i \) store all items hashed to slot \( i \) in a linked list. \( T[i] \) points to the linked list.
2. **Lookup**: to find if \( y \in \mathcal{U} \) is in \( T \), check the linked list at \( T[h(y)] \). Time proportion to size of linked list.

This is also known as **Open hashing**.
Handling Collisions

Several other techniques:

1. Cuckoo hashing.
   Every value has two possible locations. When inserting, insert in one of the locations, otherwise, kick stored value to its other location. Repeat till stable. if no stability then rebuild table.

2. ...

3. Others.
Understanding Hashing

Does hashing give $O(1)$ time per operation for dictionaries?

Questions:
1. Complexity of evaluating $h$ on a given element?
2. Relative sizes of the universe $U$ and the set to be stored $S$.
3. Size of table relative to size of $S$.
4. Worst-case vs average-case vs randomized (expected) time?
5. How do we choose $h$?
Understanding Hashing

Does hashing give $\mathcal{O}(1)$ time per operation for dictionaries?

Questions:

1. Complexity of evaluating $h$ on a given element?
2. Relative sizes of the universe $U$ and the set to be stored $S$.
3. Size of table relative to size of $S$.
4. Worst-case vs average-case vs randomized (expected) time?
5. How do we choose $h$?
Understanding Hashing

1. Complexity of evaluating $h$ on a given element? Should be small.

2. Relative sizes of the universe $U$ and the set to be stored $S$: typically $|U| \gg |S|$.

3. Size of table relative to size of $S$. The load factor of $T$ is the ratio $n/m$ where $n = |S|$ and $m = |T|$. Typically $n/m$ is a small constant smaller than 1. Also known as the fill factor.
Understanding Hashing

1. Complexity of evaluating $h$ on a given element? Should be small.
2. Relative sizes of the universe $U$ and the set to be stored $S$: typically $|U| \gg |S|$.
3. Size of table relative to size of $S$. The load factor of $T$ is the ratio $n/m$ where $n = |S|$ and $m = |T|$. Typically $n/m$ is a small constant smaller than 1. Also known as the fill factor.

Main and interrelated questions:

1. Worst-case vs average-case vs randomized (expected) time?
2. How do we choose $h$?
Single hash function

1. \( \mathcal{U} \): universe (very large).

2. Assume \( N = |\mathcal{U}| \gg m \) where \( m \) is size of table \( T \). In particular assume \( N \geq m^2 \) (very conservative).

3. Fix hash function \( h : \mathcal{U} \rightarrow \{0, \ldots, m-1\} \).

4. \( N \) items hashed to \( m \) slots. By pigeon hole principle there is some \( i \in \{0, \ldots, m-1\} \) such that \( N/m \geq m \) elements of \( \mathcal{U} \) get hashed to \( i \) (!).

5. Implies that there is a set \( S \subseteq \mathcal{U} \) where \( |S| = m \) such that all of \( S \) hashes to same slot. Ooops.
Single hash function

1. \( U \): universe (very large).

2. Assume \( N = |U| \gg m \) where \( m \) is size of table \( T \). In particular assume \( N \geq m^2 \) (very conservative).

3. Fix hash function \( h : U \rightarrow \{0, \ldots, m-1\} \).

4. \( N \) items hashed to \( m \) slots. By pigeon hole principle there is some \( i \in \{0, \ldots, m-1\} \) such that \( N/m \geq m \) elements of \( U \) get hashed to \( i \) (!).

5. Implies that there is a set \( S \subseteq U \) where \( |S| = m \) such that all of \( S \) hashes to same slot. Ooops.

**Lesson:** For every hash function there is a very bad set. Bad set. Bad.
How many hash functions are there, anyway?

Let $\mathcal{H}$ be the set of all functions from $\mathcal{U} = \{1, \ldots, U\}$ to $\{1, \ldots, m\}$. The number of functions in $\mathcal{H}$ is

(A) $U + m$.
(B) $U^m$.
(C) $U^m$.
(D) $m^U$.
(E) $\binom{U+m}{m}$.
(F) The answer is blowing in the wind.
How many bits one need?

Let $\mathcal{H}$ be a set of functions from $\mathcal{U} = \{1, \ldots, U\}$ to $\{1, \ldots, m\}$. Specifying a function in $\mathcal{H}$ requires:

(A) $O(U + m)$ bits.
(B) $O(Um)$ bits.
(C) $O(U^m)$ bits.
(D) $O(m^U)$ bits.
(E) $O(\log |\mathcal{H}|)$ bits.
(F) Many many bits. At least two.
Picking a hash function

1. Hash functions are often chosen in an ad hoc fashion. Implicit assumption is that input behaves well.

2. May work well for aircraft control. Susceptible to denial of service attack in routing.
Picking a hash function

1. Hash function are often chosen in an ad hoc fashion. Implicit assumption is that input behaves well.
2. May work well for aircraft control. Susceptible to denial of service attack in routing.

Parameters: \( N = |U|, \ m = |T|, \ n = |S| \)

1. \( H \) is a family of hash functions: each function \( h \in H \) should be efficient to evaluate (that is, to compute \( h(x) \)).
2. \( h \) is chosen randomly from \( H \) (typically uniformly at random). Implicitly assumes that \( H \) allows an efficient sampling.
3. Randomized guarantee: should have the property that for any fixed set \( S \subseteq U \) of size \( m \) the expected number of collisions for a function chosen from \( H \) should be “small”. Here the expectation is over the randomness in choice of \( h \).
**Question:** Why not let $\mathcal{H}$ be the set of all functions from $\mathcal{U}$ to $\{0, 1, \ldots, m - 1\}$?

Too many functions! A random function has high complexity!

$M = m^{\left|\mathcal{U}\right|}$.

Bits to encode such a function $\approx \log M = \left|\mathcal{U}\right| \log m$.

**Question:** Are there good and compact families $\mathcal{H}$?

Yes... But what it means for $\mathcal{H}$ to be good and compact.
Question: Why not let $\mathcal{H}$ be the set of all functions from $\mathcal{U}$ to $\{0, 1, \ldots, m - 1\}$?

Too many functions! A random function has high complexity!

- Number of functions: $M = m^{|\mathcal{U}|}$.
- Bits to encode such a function $\approx \log M = |\mathcal{U}| \log m$. 

Yes... But what it means for $\mathcal{H}$ to be good and compact.
Question: Why not let $\mathcal{H}$ be the set of all functions from $\mathcal{U}$ to $\{0, 1, \ldots, m - 1\}$?

Too many functions! A random function has high complexity!

# of functions: $M = m^{|\mathcal{U}|}$.

Bits to encode such a function $\approx \log M = |\mathcal{U}| \log m$.

Question: Are there good and compact families $\mathcal{H}$?
Picking a hash function

**Question:** Why not let \( \mathcal{H} \) be the set of all functions from \( \mathcal{U} \) to \( \{0, 1, \ldots, m - 1\} \)?

- Too many functions! A random function has high complexity!
  
  # of functions: \( M = m^{\left|\mathcal{U}\right|} \).
  
  Bits to encode such a function \( \approx \log M = \left|\mathcal{U}\right| \log m \).

**Question:** Are there good and compact families \( \mathcal{H} \)?

- Yes... But what it means for \( \mathcal{H} \) to be good and compact.
Uniform hashing

**Question:** What are good properties of $\mathcal{H}$ in distributing data?

1. Consider any element $x \in U$. Then if $h \in \mathcal{H}$ is picked randomly then $x$ should go into a random slot in $T$. In other words $\Pr[h(x) = i] = 1/m$ for every $0 \leq i < m$. (Uniform)

2. Consider any two distinct elements $x, y \in U$. Then if $h \in \mathcal{H}$ is picked randomly then the probability of a collision between $x$ and $y$ should be at most $1/m$. In other words $\Pr[h(x) = h(y)] = 1/m$ (cannot be smaller).

3. Second property is stronger than the first and the crucial issue.

Definition: A family hash function $\mathcal{H}$ is $(2-)$universal if for all distinct $x, y \in U$, $\Pr[h(x) = h(y)] = 1/m$ where $m$ is the table size.

Note: The set of all hash functions satisfies stronger properties!
Question: What are good properties of $\mathcal{H}$ in distributing data?

1. Consider any element $x \in U$. Then if $h \in \mathcal{H}$ is picked randomly then $x$ should go into a random slot in $T$. In other words $\Pr[h(x) = i] = 1/m$ for every $0 \leq i < m$. (Uniform)
Uniform hashing

**Question:** What are good properties of $\mathcal{H}$ in distributing data?

1. Consider any element $x \in \mathcal{U}$. Then if $h \in \mathcal{H}$ is picked randomly then $x$ should go into a random slot in $T$. In other words $\Pr[h(x) = i] = 1/m$ for every $0 \leq i < m$. (Uniform)

2. Consider any two distinct elements $x, y \in \mathcal{U}$. Then if $h \in \mathcal{H}$ is picked randomly then the probability of a collision between $x$ and $y$ should be at most $1/m$. In other words $\Pr[h(x) = h(y)] = 1/m$ (cannot be smaller).
Uniform hashing

**Question:** What are good properties of $\mathcal{H}$ in distributing data?

1. Consider any element $x \in \mathcal{U}$. Then if $h \in \mathcal{H}$ is picked randomly then $x$ should go into a random slot in $T$. In other words $\Pr[h(x) = i] = 1/m$ for every $0 \leq i < m$. (Uniform)

2. Consider any two distinct elements $x, y \in \mathcal{U}$. Then if $h \in \mathcal{H}$ is picked randomly then the probability of a collision between $x$ and $y$ should be at most $1/m$. In other words $\Pr[h(x) = h(y)] = 1/m$ (cannot be smaller).

3. Second property is stronger than the first and the crucial issue.

**Definition**

A family hash function $\mathcal{H}$ is **(2-)universal** if for all distinct $x, y \in \mathcal{U}$, $\Pr_h[h(x) = h(y)] = 1/m$ where $m$ is the table size.
Uniform hashing

Question: What are good properties of $\mathcal{H}$ in distributing data?

1. Consider any element $x \in U$. Then if $h \in \mathcal{H}$ is picked randomly then $x$ should go into a random slot in $T$. In other words $\Pr[h(x) = i] = 1/m$ for every $0 \leq i < m$. (Uniform)

2. Consider any two distinct elements $x, y \in U$. Then if $h \in \mathcal{H}$ is picked randomly then the probability of a collision between $x$ and $y$ should be at most $1/m$. In other words $\Pr[h(x) = h(y)] = 1/m$ (cannot be smaller).

3. Second property is stronger than the first and the crucial issue.

Definition

A family hash function $\mathcal{H}$ is (2-)universal if for all distinct $x, y \in U$, $\Pr_h[h(x) = h(y)] = 1/m$ where $m$ is the table size.

Note: The set of all hash functions satisfies stronger properties!
Analyzing Universal Hashing

1. \( T \) is hash table of size \( m \).
2. \( S \subseteq U \) is a fixed set of size \( \leq m \).
3. \( h \) is chosen randomly from a universal hash family \( \mathcal{H} \).
4. \( x \) is a fixed element of \( U \).

**Question:** What is the expected time to look up \( x \) in \( T \) using \( h \) assuming chaining used to resolve collisions?
Question: What is the expected time to look up $x$ in $T$ using $h$ assuming chaining used to resolve collisions?

1. The time to look up $x$ is the size of the list at $T[h(x)]$: same as the number of elements in $S$ that collide with $x$ under $h$.

2. Let $\ell(x)$ be this number. We want $E[\ell(x)]$.

3. For $y \in S$ let $A_y$ be the event that $x, y$ collide and $D_y$ be the corresponding indicator variable.
Analyzing Universal Hashing

Continued...

Number of elements colliding with $x$: $\ell(x) = \sum_{y \in S} D_y$.

$$\Rightarrow E[\ell(x)] = \sum_{y \in S} E[D_y]$$

linearity of expectation

$$= \sum_{y \in S} \Pr[h(x) = h(y)]$$

$$= \sum_{y \in S} \frac{1}{m}$$

since $\mathcal{H}$ is a universal hash family

$$= \frac{|S|}{m}$$

$$\leq 1 \text{ if } |S| \leq m$$
Analyzing Universal Hashing

**Question:** What is the *expected* time to look up $x$ in $T$ using $h$ assuming chaining used to resolve collisions?

**Answer:** $O(n/m)$. 

1. $O(1)$ expected time also holds for insertion.
2. Analysis assumes static set $S$ but holds as long as $S$ is a set formed with at most $O(m)$ insertions and deletions.
3. Worst-case: look up time can be large! How large? $\Omega(\log n / \log \log n)$ 
   
   [Lower bound holds even under stronger assumptions.]
Analyzing Universal Hashing

**Question:** What is the *expected* time to look up \( x \) in \( T \) using \( h \) assuming chaining used to resolve collisions?

**Answer:** \( O(n/m) \).

**Comments:**

1. \( O(1) \) expected time also holds for insertion.
2. Analysis assumes static set \( S \) but holds as long as \( S \) is a set formed with at most \( O(m) \) insertions and deletions.
3. **Worst-case:** look up time can be large! How large? \( \Omega(\log n / \log \log n) \)  
   [Lower bound holds even under stronger assumptions.]
Desired Hash Family $\mathcal{H}$

$p > |\mathcal{U}|$ be a prime. Define $h_{a,b}(x) = (ax + b \mod p) \mod m$.

$$\mathcal{H} = \{h_{a,b} \mid a \in \{1, \ldots, p - 1\}, b \in \{0, \ldots, p - 1\}\}$$
Desired Hash Family $\mathcal{H}$

$p > |\mathcal{U}|$ be a prime. Define $h_{a,b}(x) = (ax + b \mod p) \mod m$.

$$
\mathcal{H} = \{ h_{a,b} \mid a \in \{1, \ldots, p - 1\}, b \in \{0, \ldots, p - 1\} \}
$$

1. $h_{a,b}$ can be evaluated in $O(1)$ time.
2. Easy to sample.
3. **Universal!**
So far we assumed fixed $S$ of size $\sim m$.

**Question:** What happens as items are inserted and deleted?

1. If $|S|$ grows to more than $cm$ for some constant $c$ then hash table performance clearly degrades.

2. If $|S|$ stays around $\sim m$ but incurs many insertions and deletions then the initial random hash function is no longer random enough!

**Solution:** Rebuild hash table periodically!

1. Choose a new table size based on current number of elements in table.
2. Choose a new random hash function and rehash the elements.
3. Discard old table and hash function.

**Question:** When to rebuild? How expensive?
So far we assumed fixed $S$ of size $\approx m$.

**Question:** What happens as items are inserted and deleted?

1. If $|S|$ grows to more than $cm$ for some constant $c$ then hash table performance clearly degrades.
2. If $|S|$ stays around $\approx m$ but incurs many insertions and deletions then the initial random hash function is no longer random enough!

**Solution:** Rebuild hash table periodically!

1. Choose a new table size based on current number of elements in table.
2. Choose a new random hash function and rehash the elements.
3. Discard old table and hash function.

**Question:** When to rebuild? How expensive?
Rebuilding the hash table

1. Start with table size $m$ where $m$ is some estimate of $|S|$ (can be some large constant).

2. If $|S|$ grows to more than twice current table size, build new hash table (choose a new random hash function) with double the current number of elements. Can also use similar trick if table size falls below quarter the size.

3. If $|S|$ stays roughly the same but more than $c|S|$ operations on table for some chosen constant $c$ (say 10), rebuild.

The amortize cost of rebuilding to previously performed operations. Rebuilding ensures $O(1)$ expected analysis holds even when $S$ changes. Hence $O(1)$ expected look up/insert/delete time dynamic data dictionary data structure!
Part I

Hash Tables
Dictionary Data Structure

1. $\mathcal{U}$: universe of keys with total order: numbers, strings, etc.
2. Data structure to store a subset $S \subseteq \mathcal{U}$
3. Operations:
   1. Search/look up: given $x \in \mathcal{U}$ is $x \in S$?
   2. Insert: given $x \not\in S$ add $x$ to $S$.
   3. Delete: given $x \in S$ delete $x$ from $S$
4. Static structure: $S$ given in advance or changes very infrequently, main operations are lookups.
5. Dynamic structure: $S$ changes rapidly so inserts and deletes as important as lookups.

Can we do everything in $O(1)$ time?
Hashing and Hash Tables

Hash Table data structure:
1. A (hash) table/array $T$ of size $m$ (the table size).
2. A hash function $h : \mathcal{U} \to \{0, \ldots, m-1\}$.
3. Item $x \in \mathcal{U}$ hashes to slot $h(x)$ in $T$. 

Given $S \subseteq \mathcal{U}$. How do we store $S$ and how do we do lookups?

Ideal situation:
1. Each element $x \in S$ hashes to a distinct slot in $T$. Store $x$ in slot $h(x)$.
2. Lookup: Given $y \in \mathcal{U}$ check if $T[h(y)] = y$. $O(1)$ time!

Collisions unavoidable if $|T| < |\mathcal{U}|$. Several techniques to handle them.

Chandra & Ruta (UIUC) CS473 5 Fall 2016 5 / 32
Hashing and Hash Tables

Hash Table data structure:
1. A (hash) table/array $T$ of size $m$ (the table size).
2. A hash function $h : U \rightarrow \{0, \ldots, m - 1\}$.
3. Item $x \in U$ hashes to slot $h(x)$ in $T$.

Given $S \subseteq U$. How do we store $S$ and how do we do lookups?
Hashing and Hash Tables

Hash Table data structure:
1. A (hash) table/array $T$ of size $m$ (the table size).
2. A hash function $h : \mathcal{U} \rightarrow \{0, \ldots, m - 1\}$.
3. Item $x \in \mathcal{U}$ hashes to slot $h(x)$ in $T$.

Given $S \subseteq \mathcal{U}$. How do we store $S$ and how do we do lookups?

**Ideal situation:**
1. Each element $x \in S$ hashes to a distinct slot in $T$. Store $x$ in slot $h(x)$
2. **Lookup**: Given $y \in \mathcal{U}$ check if $T[h(y)] = y$. $O(1)$ time!
Hashing and Hash Tables

Hash Table data structure:
1. A (hash) table/array $T$ of size $m$ (the table size).
2. A hash function $h : U \rightarrow \{0, \ldots, m - 1\}$.
3. Item $x \in U$ hashes to slot $h(x)$ in $T$.

Given $S \subseteq U$. How do we store $S$ and how do we do lookups?

**Ideal situation:**
1. Each element $x \in S$ hashes to a distinct slot in $T$. Store $x$ in slot $h(x)$
2. **Lookup**: Given $y \in U$ check if $T[h(y)] = y$. $O(1)$ time!

Collisions unavoidable if $|T| < |U|$. Several techniques to handle them.
Handling Collisions: Chaining

Collision: \( h(x) = h(y) \) for some \( x \neq y \).

Chaining/Open hashing to handle collisions:

1. For each slot \( i \) store all items hashed to slot \( i \) in a linked list. \( T[i] \) points to the linked list.

2. Lookup: to find if \( y \in U \) is in \( T \), check the linked list at \( T[h(y)] \). Time proportion to size of linked list.

Does hashing give \( \mathcal{O}(1) \) time per operation for dictionaries?
Hash Functions

Parameters: $N = |\mathcal{U}|$ (very large), $m = |T|$, $n = |S|$
Goal: $\mathcal{O}(1)$-time lookup, insertion, deletion.

Single hash function

If $N \geq m^2$, then for any hash function $h : \mathcal{U} \rightarrow T$ there exists $i < m$ such that at least $N/m \geq m$ elements of $\mathcal{U}$ get hashed to slot $i$. 
Hash Functions

Parameters: \( N = |\mathcal{U}| \) (very large), \( m = |T| \), \( n = |S| \)
Goal: \( O(1) \)-time lookup, insertion, deletion.

Single hash function

If \( N \geq m^2 \), then for any hash function \( h : \mathcal{U} \rightarrow T \) there exists \( i < m \) such that at least \( N/m \geq m \) elements of \( \mathcal{U} \) get hashed to slot \( i \). Any \( S \) containing all of these is a very very bad set for \( h \)!
Hash Functions

Parameters: $N = |U|$ (very large), $m = |T|$, $n = |S|$
Goal: $O(1)$-time lookup, insertion, deletion.

Single hash function

If $N \geq m^2$, then for any hash function $h : U \rightarrow T$ there exists $i < m$ such that at least $N/m \geq m$ elements of $U$ get hashed to slot $i$. Any $S$ containing all of these is a very very bad set for $h$!

Such a bad set may lead to $O(m)$ lookup time!
Hash Functions

Parameters: \( N = |U| \) (very large), \( m = |T| \), \( n = |S| \)
Goal: \( O(1) \)-time lookup, insertion, deletion.

Single hash function

If \( N \geq m^2 \), then for any hash function \( h : U \rightarrow T \) there exists \( i < m \) such that at least \( N/m \geq m \) elements of \( U \) get hashed to slot \( i \). Any \( S \) containing all of these is a very very bad set for \( h \)!

Such a bad set may lead to \( O(m) \) lookup time!

Lesson:

- Consider a family \( \mathcal{H} \) of hash functions with good properties and choose \( h \) uniformly at random.
- Guarantees: small \# collisions in expectation for a given \( S \).
- \( \mathcal{H} \) should allow efficient sampling.
Question: What are good properties of $\mathcal{H}$ in distributing data?

1. Uniform: Consider any element $x \in U$. Then if $h \in \mathcal{H}$ is picked randomly then $x$ should go into a random slot in $T$. In other words $\Pr[h(x) = i] = 1/m$ for every $0 \leq i < m$.

2. Universal: Consider any two distinct elements $x, y \in U$. Then if $h \in \mathcal{H}$ is picked randomly then the probability of a collision between $x$ and $y$ should be at most $1/m$. In other words $\Pr[h(x) = h(y)] = 1/m$ (cannot be smaller).

3. Second property is stronger than the first and the crucial issue.

Definition: A family of hash function $\mathcal{H}$ is ($2$-)universal if for all distinct $x, y \in U$, $\Pr[h \in \mathcal{H}[h(x) = h(y)] = 1/m$ where $m$ is the table size.
Universal Hashing

Question: What are good properties of $\mathcal{H}$ in distributing data?

1 Uniform: Consider any element $x \in U$. Then if $h \in \mathcal{H}$ is picked randomly then $x$ should go into a random slot in $T$. In other words $\Pr[h(x) = i] = 1/m$ for every $0 \leq i < m$. 

2 Universal: Consider any two distinct elements $x, y \in U$. Then if $h \in \mathcal{H}$ is picked randomly then the probability of a collision between $x$ and $y$ should be at most $1/m$. In other words $\Pr[h(x) = h(y)] = 1/m$ (cannot be smaller).

Definition. A family of hash functions $\mathcal{H}$ is $(\frac{1}{2})$ universal if for all distinct $x, y \in U$, $\Pr[h \sim \mathcal{H}[h(x) = h(y)] = 1/m$ where $m$ is the table size.
Universal Hashing

**Question:** What are good properties of $\mathcal{H}$ in distributing data?

1. **Uniform:** Consider any element $x \in \mathcal{U}$. Then if $h \in \mathcal{H}$ is picked randomly then $x$ should go into a random slot in $T$. In other words $\Pr[h(x) = i] = 1/m$ for every $0 \leq i < m$.

2. **Universal:** Consider any two distinct elements $x, y \in \mathcal{U}$. Then if $h \in \mathcal{H}$ is picked randomly then the probability of a collision between $x$ and $y$ should be at most $1/m$. In other words $\Pr[h(x) = h(y)] = 1/m$ (cannot be smaller).
Universal Hashing

Question: What are good properties of $\mathcal{H}$ in distributing data?

1. **Uniform**: Consider any element $x \in \mathcal{U}$. Then if $h \in \mathcal{H}$ is picked randomly then $x$ should go into a random slot in $T$. In other words $\Pr[h(x) = i] = 1/m$ for every $0 \leq i < m$.

2. **Universal**: Consider any two distinct elements $x, y \in \mathcal{U}$. Then if $h \in \mathcal{H}$ is picked randomly then the probability of a collision between $x$ and $y$ should be at most $1/m$. In other words $\Pr[h(x) = h(y)] = 1/m$ (cannot be smaller).

3. Second property is stronger than the first and the crucial issue.

**Definition**

A family of hash function $\mathcal{H}$ is (2-)universal if for all distinct $x, y \in \mathcal{U}$, $\Pr_{h \sim \mathcal{H}}[h(x) = h(y)] = 1/m$ where $m$ is the table size.
Question: Fixing set $S$, what is the expected time to look up $x \in S$ when $h$ is picked uniformly at random from $\mathcal{H}$?

1. $\ell(x)$: the size of the list at $T[h(x)]$. We want $E[\ell(x)]$

2. For $y \in S$ let $D_y$ be one if $h(y) = h(x)$, else zero.

$\ell(x) = \sum_{y \in S} D_y$
Analyzing Universal Hashing

**Question:** Fixing set $S$, what is the *expected* time to look up $x \in S$ when $h$ is picked uniformly at random from $\mathcal{H}$?

1. $\ell(x)$: the size of the list at $T[h(x)]$. We want $E[\ell(x)]$

2. For $y \in S$ let $D_y$ be one if $h(y) = h(x)$, else zero.

   $\ell(x) = \sum_{y \in S} D_y$

   

   $$
   E[\ell(x)] = \sum_{y \in S} E[D_y] = \sum_{y \in S} Pr[h(x) = h(y)]
   $$

   $$
   = \sum_{y \in S} \frac{1}{m} \quad \text{(since } \mathcal{H} \text{ is a universal hash family)}
   $$

   $$
   = \frac{|S|}{m} \leq 1 \quad \text{if } |S| \leq m
   $$
Analyzing Universal Hashing

**Question:** What is the *expected* time to look up $x$ in $T$ using $h$ assuming chaining used to resolve collisions?

**Answer:** $O(n/m)$.
Question: What is the expected time to look up \( x \) in \( T \) using \( h \) assuming chaining used to resolve collisions?

Answer: \( O(n/m) \).

Comments:

1. \( O(1) \) expected time also holds for insertion.
2. Analysis assumes static set \( S \) but holds as long as \( S \) is a set formed with at most \( O(m) \) insertions and deletions.
3. **Worst-case**: look up time can be large! How large?
   \( \Omega(\log n / \log \log n) \)
Universal Hash Family

Universal: $\mathcal{H}$ such that $\Pr[h(x) = h(y)] = 1/m$.

All functions

$\mathcal{H}$: Set of all possible functions $h : \mathcal{U} \to \{0, \ldots, m - 1\}$.

- Universal.
Universal Hash Family

Universal: \( \mathcal{H} \) such that \( \Pr[h(x) = h(y)] = 1/m \).

All functions

\( \mathcal{H} : \) Set of all possible functions \( h : \mathcal{U} \rightarrow \{0, \ldots, m - 1\} \).

- Universal.
- \( |\mathcal{H}| = m^{|\mathcal{U}|} \)
- representing \( h \) requires \( |\mathcal{U}| \log m \) – Not \( O(1) \)!
Universal Hash Family

Universal: $\mathcal{H}$ such that $\Pr[h(x) = h(y)] = 1/m$.

All functions

$\mathcal{H}$: Set of all possible functions $h: \mathcal{U} \rightarrow \{0, \ldots, m-1\}$.

- Universal.
- $|\mathcal{H}| = m^{|\mathcal{U}|}$
- representing $h$ requires $|\mathcal{U}| \log m$ – Not $O(1)$!

We need compactly representable universal family.
Compact Universal Hash Family

Parameters: $N = |\mathcal{U}|$, $m = |\mathcal{T}|$, $n = |\mathcal{S}|$

1. Choose a prime number $p \geq N$. $\mathbb{Z}_p = \{0, 1, \ldots, p - 1\}$ is a field.
2. For $a, b \in \mathbb{Z}_p$, $a \neq 0$, define the hash function $h_{a,b}$ as $h_{a,b}(x) = ((ax + b) \mod p) \mod m$.
3. Let $\mathcal{H} = \{h_{a,b} \mid a, b \in \mathbb{Z}_p, a \neq 0\}$. Note that $|\mathcal{H}| = p(p - 1)$. 

Comments:
1. Hash family is of small size, easy to sample from.
2. Easy to store a hash function ($a, b$ have to be stored) and evaluate it.
Compact Universal Hash Family

Parameters: $N = |U|$, $m = |T|$, $n = |S|$

1. Choose a prime number $p \geq N$. $\mathbb{Z}_p = \{0, 1, \ldots, p - 1\}$ is a field.

2. For $a, b \in \mathbb{Z}_p$, $a \neq 0$, define the hash function $h_{a,b}$ as $h_{a,b}(x) = ((ax + b) \mod p) \mod m$.

3. Let $\mathcal{H} = \{h_{a,b} \mid a, b \in \mathbb{Z}_p, a \neq 0\}$. Note that $|\mathcal{H}| = p(p - 1)$.

**Theorem**

$\mathcal{H}$ is a universal hash family.
Compact Universal Hash Family

Parameters: $N = |\mathcal{U}|$, $m = |T|$, $n = |S|$

1. Choose a **prime** number $p \geq N$. $\mathbb{Z}_p = \{0, 1, \ldots, p - 1\}$ is a field.

2. For $a, b \in \mathbb{Z}_p$, $a \neq 0$, define the hash function $h_{a,b}$ as
   
   \[ h_{a,b}(x) = ((ax + b) \mod p) \mod m. \]

3. Let $\mathcal{H} = \{h_{a,b} \mid a, b \in \mathbb{Z}_p, a \neq 0\}$. Note that $|\mathcal{H}| = p(p - 1)$.

**Theorem**

$\mathcal{H}$ is a **universal hash family**.

**Comments:**

1. Hash family is of small size, easy to sample from.
2. Easy to store a hash function ($a, b$ have to be stored) and evaluate it.
Lemma (LemmaUnique)

Let $p$ be a prime number, $x$: an integer number in $\{1, \ldots, p - 1\}$.

$\implies$ There exists a unique $y$ s.t. $xy = 1 \mod p$.

In other words: For every element there is a unique inverse.

$\implies\mathbb{Z}_p = \{0, 1, \ldots, p - 1\}$ when working modulo $p$ is a field.
Proof of Lemma Unique

Claim

Let $p$ be a prime number. For any $x, y, z \in \{1, \ldots, p - 1\}$ s.t. $y \neq z$, we have that $xy \mod p \neq xz \mod p$.

Proof.

Assume for the sake of contradiction $xy \mod p = xz \mod p$. Then

$$x(y - z) = 0 \mod p$$

$$\implies p \text{ divides } x(y - z)$$

$$\implies p \text{ divides } y - z$$

$$\implies y - z = 0$$

$$\implies y = z.$$
Lemma (LemmaUnique)

Let $p$ be a prime number, $x$: an integer number in $\{1, \ldots, p - 1\}$. $\implies$ There exists a unique $y$ s.t. $xy = 1 \pmod{p}$.

Proof.

By the above claim if $xy = 1 \pmod{p}$ and $xz = 1 \pmod{p}$ then $y = z$. Hence uniqueness follows.
Lemma (LemmaUnique)

Let \( p \) be a prime number,

\( x: \) an integer number in \( \{1, \ldots, p - 1\} \).

\[ \implies \] There exists a unique \( y \) s.t. \( xy = 1 \mod p \).

Proof.

By the above claim if \( xy = 1 \mod p \) and \( xz = 1 \mod p \) then \( y = z \). Hence uniqueness follows.

Existence. For any \( x \in \{1, \ldots, p - 1\} \) we have that

\( \{x \times 1 \mod p, x \times 2 \mod p, \ldots, x \times (p - 1) \mod p\} = \{1, 2, \ldots, p - 1\} \).

\[ \implies \] There exists a number \( y \in \{1, \ldots, p - 1\} \) such that \( xy = 1 \mod p \).
Proof of the Theorem: Outline

\[ h_{a,b}(x) = ((ax + b) \mod p) \mod m). \]

**Theorem**

\[ \mathcal{H} = \{ h_{a,b} \mid a, b \in \mathbb{Z}_p, a \neq 0 \} \text{ is universal.} \]

**Proof.**

Fix \( x, y \in U \). We need to show that

\[ \Pr_{h_{a,b} \sim \mathcal{H}}[h_{a,b}(x) = h_{a,b}(y)] \leq 1/m. \]

Note that \( |\mathcal{H}| = p(p - 1) \).
Proof of the Theorem: Outline

\[ h_{a,b}(x) = ((ax + b) \mod p) \mod m). \]

**Theorem**

\[ \mathcal{H} = \{ h_{a,b} \mid a, b \in \mathbb{Z}_p, a \neq 0 \} \text{ is universal.} \]

**Proof.**

Fix \( x, y \in \mathcal{U} \). We need to show that

\[ \Pr_{h_{a,b} \sim \mathcal{H}}[h_{a,b}(x) = h_{a,b}(y)] \leq 1/m. \]

Note that \( |\mathcal{H}| = p(p - 1) \).

1. Let \((a, b)\) (equivalently \( h_{a,b} \)) be bad for \( x, y \) if \( h_{a,b}(x) = h_{a,b}(y) \).

2. **Claim:** Number of bad \((a, b)\) is at most \( p(p - 1)/m \).

3. Total number of hash functions is \( p(p - 1) \) and hence probability of a collision is \( \leq 1/m \).
Intuition for the Claim

\[ g_{a,b}(x) = (ax + b) \mod p, \quad h_{a,b}(x) = (g_{a,b}(x)) \mod m \]

First map \( x \neq y \) to \( r = g_{a,b}(x) \) and \( s = g_{a,b}(y) \). \( r \neq s \) (LemmaUnique)
Intuition for the Claim

\[ g_{a,b}(x) = (ax + b) \mod p, \quad h_{a,b}(x) = (g_{a,b}(x)) \mod m \]
First map \( x \neq y \) to \( r = g_{a,b}(x) \) and \( s = g_{a,b}(y) \). \( r \neq s \)

(LemmaUnique)

As \( (a, b) \) varies, \( (r, s) \) takes all possible \( p(p - 1) \) values. Since \( (a, b) \) is picked u.a.r., every value of \( (r, s) \) has equal probability.
Intuition for the Claim

\[ g_{a,b}(x) = (ax + b) \mod p, \quad h_{a,b}(x) = (g_{a,b}(x)) \mod m \]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
\hline
\hline
r & & & \\
\hline
s & & & \\
\hline
(r, s) & & & \\
\hline
\end{array}
\]
Intuition for the Claim

\[ g_{a,b}(x) = (ax + b) \mod p, \quad h_{a,b}(x) = (g_{a,b}(x)) \mod m \]
Intuition for the Claim

\[ g_{a,b}(x) = (ax + b) \mod p, \quad h_{a,b}(x) = (g_{a,b}(x)) \mod m \]

1. First part of mapping maps \((x, y)\) to a random location \((g_{a,b}(x), g_{a,b}(y))\) in the “matrix”.

2. \((g_{a,b}(x), g_{a,b}(y))\) is not on main diagonal.

3. All blue locations are “bad” – map by \(\mod m\) to a location of collusion.

4. But... at most \(1/m\) fraction of allowable locations in the matrix are bad.
We need to show at most $1/m$ fraction of bad $h_{a,b}$

$$h_{a,b}(x) = (((ax + b) \mod p) \mod m)$$

2 lemmas ...

Fix $x \neq y \in \mathbb{Z}_p$, and let $r = (ax + b) \mod p$ and $s = (ay + b) \mod p$. 

Out of all possible $p(p-1)$ pairs of $(r, s)$, at most $p(p-1)/m$ fraction satisfies $r \mod m = s \mod m$. 

Chandra & Ruta (UIUC)  CS473  18 Fall 2016  18 / 32
We need to show at most $1/m$ fraction of bad $h_{a,b}$

$$h_{a,b}(x) = (((ax + b) \mod p) \mod m)$$

2 lemmas ...

Fix $x \neq y \in \mathbb{Z}_p$, and let $r = (ax + b) \mod p$ and $s = (ay + b) \mod p$.

- 1-to-1 correspondence between $p(p-1)$ pairs of $(a, b)$ (equivalently $h_{a,b}$) and $p(p-1)$ pairs of $(r, s)$. 
We need to show at most $1/m$ fraction of bad $h_{a,b}$

$$h_{a,b}(x) = (((ax + b) \mod p) \mod m)$$

2 lemmas ...

Fix $x \neq y \in \mathbb{Z}_p$, and let $r = (ax + b) \mod p$ and $s = (ay + b) \mod p$.

1. 1-to-1 correspondence between $p(p-1)$ pairs of $(a,b)$ (equivalently $h_{a,b}$) and $p(p-1)$ pairs of $(r,s)$.

2. Out of all possible $p(p-1)$ pairs of $(r,s)$, at most $p(p-1)/m$ fraction satisfies $r \mod m = s \mod m$. 
Lemma

If $x \neq y$ then for any $a, b \in \mathbb{Z}_p$ such that $a \neq 0$, we have
$$ax + b \mod p \neq ay + b \mod p.$$ 

Proof.

If $ax + b \mod p = ay + b \mod p$ then $a(x - y) \mod p = 0$
and $a \neq 0$ and $(x - y) \neq 0$. However, $a$ and $(x - y)$ cannot divide
$p$ since $p$ is prime and $a < p$ and $(x - y) < p$. 

Chandra & Ruta (UIUC) CS473 19 Fall 2016 19 / 32
Lemma

If \( x \neq y \) then for each \((r, s)\) such that \( r \neq s \) and \( 0 \leq r, s \leq p - 1 \) there is exactly one \( a, b \) such that

\[
ax + b \mod p = r \quad \text{and} \quad ay + b \mod p = s.
\]

Proof.

Solve the two equations:

\[
ax + b = r \mod p \quad \text{and} \quad ay + b = s \mod p.
\]

We get \( a = \frac{r - s}{x - y} \mod p \) and \( b = r - ax \mod p \).

One-to-one correspondence between \((a, b)\) and \((r, s)\)
Understanding the hashing

Once we fix \( a \) and \( b \), and we are given a value \( x \), we compute the hash value of \( x \) in two stages:

1. **Compute**: \( r \leftarrow (ax + b) \mod p \).
2. **Fold**: \( r' \leftarrow r \mod m \)

Collision...

Given two distinct values \( x \) and \( y \) they might collide only because of folding.

Lemma

\[ \# \text{ not equal pairs } (r, s) \text{ of } \mathbb{Z}_p \times \mathbb{Z}_p \text{ that are folded to the same number is } p(p - 1)/m. \]
Folding numbers

Lemma

\# pairs \((r, s) \in \mathbb{Z}_p \times \mathbb{Z}_p\) such that \(r \neq s\) and \(r \mod m = s \mod m\) (folded to the same number) is \(p(p - 1)/m\).

Proof.

Consider a pair \((r, s) \in \{0, 1, \ldots, p - 1\}^2\) s.t. \(r \neq s\). Fix \(r:\)

1. \(a = r \mod m\).
2. There are \(\lceil p/m \rceil\) values of \(s\) that fold into \(a\). That is \(r \mod m = s \mod m\).
3. One of them is when \(r = s\).
4. \(\implies \) \# of colliding pairs \((\lceil p/m \rceil - 1)p \leq (p - 1)p/m\)
Proof of Claim

# of bad pairs is $p(p - 1)/m$

Proof.

Let $a, b \in \mathbb{Z}_p$ such that $a \neq 0$ and $h_{a,b}(x) = h_{a,b}(y)$.

1. Let $r = ax + b \mod p$ and $s = ay + b \mod p$.
2. Collision if and only if $r \mod m = s \mod m$.
3. (Folding error): Number of pairs $(r, s)$ such that $r \neq s$ and $0 \leq r, s \leq p - 1$ and $r \mod m = s \mod m$ is $p(p - 1)/m$.
4. From previous lemma there is one-to-one correspondence between $(a, b)$ and $(r, s)$. Hence total number of bad $(a, b)$ pairs is $p(p - 1)/m$. 

Proof of Claim

# of bad pairs is $\frac{p(p - 1)}{m}$

Proof.

Let $a, b \in \mathbb{Z}_p$ such that $a \neq 0$ and $h_{a,b}(x) = h_{a,b}(y)$.

1. Let $r = ax + b \mod p$ and $s = ay + b \mod p$.

2. Collision if and only if $r \mod m = s \mod m$.

3. (Folding error): Number of pairs $(r, s)$ such that $r \neq s$ and $0 \leq r, s \leq p - 1$ and $r \mod m = s \mod m$ is $\frac{p(p - 1)}{m}$.

4. From previous lemma there is one-to-one correspondence between $(a, b)$ and $(r, s)$. Hence total number of bad $(a, b)$ pairs is $\frac{p(p - 1)}{m}$.

Prob of $x$ and $y$ to collide: $\frac{\# \text{ bad } (a, b) \text{ pairs}}{\#(a, b) \text{ pairs}} = \frac{p(p-1)/m}{p(p-1)} = \frac{1}{m}$. 
Look up Time

Say $|S| = |T| = m$. For $0 \leq i \leq m - 1$, $\ell(i)$ : number of elements hashed to slot $i$ in $T$.

Expected look up time

Since for $x \neq y$, $\Pr[h_{a,b}(x) = h_{a,b}(y)] = 1/m$, we get $E[\ell(i)] = |S|/m = 1$. 

Claim: If $|T| = m^2$, then $E[\max_{0 \leq i \leq m-1} \ell(i)] = O(1)$.
Look up Time

Say \(|S| = |T| = m\).
For \(0 \leq i \leq m - 1\), \(\ell(i)\) : number of elements hashed to slot \(i\) in \(T\).

**Expected look up time**

Since for \(x \neq y\), \(\Pr[h_{a,b}(x) = h_{a,b}(y)] = 1/m\), we get
\[E[\ell(i)] = |S|/m = 1.\]

**Expected worst case look up time**

Like in Balls & Bins, \(E\left[\max_{i=0}^{m-1} \ell(i)\right] \geq O(\ln n / \ln \ln n)\).
Look up Time

Say $|S| = |T| = m$.
For $0 \leq i \leq m - 1$, $\ell(i)$: number of elements hashed to slot $i$ in $T$.

Expected look up time

Since for $x \neq y$, $\Pr[h_{a,b}(x) = h_{a,b}(y)] = 1/m$, we get $E[\ell(i)] = |S|/m = 1$.

Expected worst case look up time

Like in Balls & Bins, $E\left[\max_{i=0}^{m-1} \ell(i)\right] \geq O(\ln n / \ln \ln n)$.

What if $|T| = m^2$ (# Bins is $m^2$)

Claim: If $|T| = m^2$, then $E\left[\max_{i=0}^{m-1} \ell(i)\right] = O(1)$. 
Perfect Hashing
Two levels of hash tables

Question: Can we make look up time $O(1)$ in worst case?

Perfect Hashing for Static Data
- Do hashing once.
- If $Y_i = |\ell(i)| > 10$ then hash elements of $\ell(i)$ to a table of $Y_i^2$ size.
Question: Can we make look up time $O(1)$ in worst case?

Perfect Hashing for Static Data
- Do hashing once.
- If $Y_i = |\ell(i)| > 10$ then hash elements of $\ell(i)$ to a table of $Y_i^2$ size.

Lemma
Worst case expected look up time is $O(1)$. 
Perfect Hashing
Two levels of hash tables

**Question:** Can we make look up time $O(1)$ in worst case?

**Perfect Hashing for Static Data**

- Do hashing once.
- If $Y_i = |\ell(i)| > 10$ then hash elements of $\ell(i)$ to a table of $Y_i^2$ size.

**Lemma**

*Worst case expected look up time is $O(1)$.*

**Lemma**

*If $|S| = O(m)$ then space usage of perfect hashing is $O(m)$.***
Intuition: Throwing $m$ Balls into $m^2$ Bins

- $\Pr[\text{ith ball lands in jth bin}] = \frac{1}{m^2}$

For a fixed bin $j$, $Y_j = \# \text{balls in bin } j$.

$E[Y_j] = \frac{1}{m}$.

For $c \geq 3$, let $\delta = cm - 1$.

$\Pr[Y_j > c] = \Pr[Y_j > \frac{cm}{m}] = \Pr[Y_j > (1 + \delta) E[Y_j]]$ (Chernoff)

$\leq (e^\delta (1 + \delta))(1 + \delta))^{\frac{1}{m}} \leq (e/c)(1/mc) \leq 1/m^3$.

$\Pr[\max_j m^2 Y_j > c] \leq 1/m$ (Union bound).

$\Pr[\max_j m^2 Y_j \leq c] \geq 1 - 1/m$ – (w.h.p.)

$E[\max_j Y_j] \leq c + 1 = O(1)$.
Intuition: Throwing \( m \) Balls in to \( m^2 \) Bins

- \( \Pr[\text{ith ball lands in } j\text{th bin}] = 1/m^2 \)
- For a fixed bin \( j \), \( Y_j = \# \text{ balls in bin } j \).
Intuition: Throwing \( m \) Balls in to \( m^2 \) Bins

- \( \text{Pr}[\text{ith ball lands in jth bin}] = \frac{1}{m^2} \)
- For a fixed bin \( j \), \( Y_j = \# \) balls in bin \( j \). \( E[Y_j] = \frac{1}{m} \).
Intuition: Throwing \( m \) Balls in \( m^2 \) Bins

- \( \Pr[\text{ith ball lands in jth bin}] = 1/m^2 \)
- For a fixed bin \( j \), \( Y_j = \# \) balls in bin \( j \). \( E[Y_j] = 1/m. \)
- For \( c \geq 3 \), let \( \delta = cm - 1 \). \( \Pr[Y_j > c] \)

\[
\begin{align*}
\Pr[Y_j > cm/m] &= \Pr[Y_j > (1 + \delta) E[Y_j]] \\
&(\text{Chernoff}) < \left( \frac{e^\delta}{(1+\delta)^{(1+\delta)}} \right)^{\mu} \\
&= \left( \frac{e^{(cm-1)}}{(cm)^{cm}} \right)^{1/m} \leq (e/c)^c(1/m^c) \\
&\leq 1/m^3
\end{align*}
\]
**Intuition: Throwing \( m \) Balls in to \( m^2 \) Bins**

- \( \Pr[\text{ith ball lands in \( j \)th bin}] = \frac{1}{m^2} \)
- For a fixed bin \( j \), \( Y_j = \#\) balls in bin \( j \). \( \mathbb{E}[Y_j] = \frac{1}{m} \).
- For \( c \geq 3 \), let \( \delta = cm - 1 \). \( \Pr[Y_j > c] \)

\[
\begin{align*}
\Pr[Y_j > cm/m] &= \Pr[Y_j > (1 + \delta) \mathbb{E}[Y_j]] \\
\text{(Chernoff)} &< \left( \frac{e^{\delta}}{(1+\delta)(1+\delta)} \right)^{\mu} \\
&= \left( \frac{e^{(cm-1)}}{(cm)(cm)} \right)^{1/m} \leq \left( \frac{e}{c} \right)^c \left( \frac{1}{m^c} \right) \\
&\leq 1/m^3
\end{align*}
\]

- \( \Pr\left[ \max_{j=1}^{m^2} Y_j > c \right] \leq 1/m \) (Union bound).
- \( \Pr\left[ \max_{j=1}^{m^2} Y_j \leq c \right] \geq 1 - 1/m - (w.h.p.) \)
- \( \mathbb{E}[\max_j Y_j] \leq c + 1 = O(1) \).
So far we assumed fixed $S$ of size $\sim m$.

**Question:** What happens as items are inserted and deleted?

1. If $|S|$ grows to more than $cm$ for some constant $c$ then hash table performance clearly degrades.

2. If $|S|$ stays around $\sim m$ but incurs many insertions and deletions then the initial random hash function is no longer random enough!
Rehashing, amortization and...
... making the hash table dynamic

So far we assumed fixed $S$ of size $\sim m$.

**Question:** What happens as items are inserted and deleted?

1. If $|S|$ grows to more than $cm$ for some constant $c$ then hash table performance clearly degrades.
2. If $|S|$ stays around $\sim m$ but incurs many insertions and deletions then the initial random hash function is no longer random enough!

**Solution:** Rebuild hash table periodically!

1. Choose a new table size based on current number of elements in table.
2. Choose a new random hash function and rehash the elements.
3. Discard old table and hash function.

**Question:** When to rebuild? How expensive?
Rebuilding the hash table

1. Start with table size $m$ where $m$ is some estimate of $|S|$ (can be some large constant).

2. If $|S|$ grows to more than twice current table size, build new hash table (choose a new random hash function) with double the current number of elements. Can also use similar trick if table size falls below quarter the size.

3. If $|S|$ stays roughly the same but more than $c|S|$ operations on table for some chosen constant $c$ (say $10$), rebuild.

The **amortize** cost of rebuilding to previously performed operations. Rebuilding ensures $O(1)$ expected analysis holds even when $S$ changes. Hence $O(1)$ expected look up/insert/delete time *dynamic* data dictionary data structure!
Bloom Filters

Hashing:

1. To insert $x$ in dictionary store $x$ in table in location $h(x)$
2. To lookup $y$ in dictionary check contents of location $h(y)$
Bloom Filters

Hashing:
1. To insert \( x \) in dictionary store \( x \) in table in location \( h(x) \)
2. To lookup \( y \) in dictionary check contents of location \( h(y) \)

Bloom Filter: tradeoff space for false positives
1. Storing items in dictionary expensive in terms of memory, especially if items are unwieldy objects such as long strings, images, etc with non-uniform sizes.
2. To insert \( x \) in dictionary set \( bit \) to 1 in location \( h(x) \) (initially all bits are set to 0)
3. To lookup \( y \) if bit in location \( h(y) \) is 1 say yes, else no.
Bloom Filters

Bloom Filter: tradeoff space for false positives

1 To insert $x$ in dictionary set bit to 1 in location $h(x)$ (initially all bits are set to 0)

2 To lookup $y$ if bit in location $h(y)$ is 1 say yes, else no

3 No false negatives but false positives possible due to collisions

Reducing false positives:

1 Pick $k$ hash functions $h_1, h_2, ..., h_k$ independently

2 To insert $x$ for $1 \leq i \leq k$ set bit in location $h_i(x)$ in table $i$ to 1

3 To lookup $y$ compute $h_i(y)$ for $1 \leq i \leq k$ and say yes only if each bit in the corresponding location is 1, otherwise say no. If probability of false positive for one hash function is $\alpha < 1$ then with $k$ independent hash function it is $\alpha^k$. 

Chandra & Ruta (UIUC) CS473 30 Fall 2016 30 / 32
Bloom Filters

**Bloom Filter:** tradeoff space for false positives

1. To insert \( x \) in dictionary set *bit* to \( 1 \) in location \( h(x) \) (initially all bits are set to \( 0 \))

2. To lookup \( y \) if bit in location \( h(y) \) is \( 1 \) say yes, else no

3. No false negatives but false positives possible due to collisions

Reducing false positives:

1. Pick \( k \) hash functions \( h_1, h_2, \ldots, h_k \) *independently*

2. To insert \( x \) for \( 1 \leq i \leq k \) set bit in location \( h_i(x) \) in table \( i \) to \( 1 \)

3. To lookup \( y \) compute \( h_i(y) \) for \( 1 \leq i \leq k \) and say yes only if each bit in the corresponding location is \( 1 \), otherwise say no. If probability of false positive for one hash function is \( \alpha < 1 \) then with \( k \) independent hash function it is \( \alpha^k \).
Take away points

1. Hashing is a powerful and important technique for dictionaries. Many practical applications.
2. Randomization fundamental to understanding hashing.
3. Good and efficient hashing possible in theory and practice with proper definitions (universal, perfect, etc).
4. Related ideas of creating a compact fingerprint/sketch for objects is very powerful in theory and practice.
Practical Issues

Hashing used typically for integers, vectors, strings etc.

- Universal hashing is defined for integers. To implement for other objects need to map objects in some fashion to integers (via representation)
- Practical methods for various important cases such as vectors, strings are studied extensively. See http://en.wikipedia.org/wiki/Universal_hashing for some pointers.
- Cryptographic hash functions have a different motivation and