CS 473: Algorithms

Chandra Chekuri   Ruta Mehta

University of Illinois, Urbana-Champaign

Fall 2016

High Probability Analysis & Universal Hashing

Lecture 09

September 21, 2016

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 $\longrightarrow$ max size w.h.p.
- Analogy to hashing

Hashing

Part I

Randomized **QuickSort** (Contd.)

Randomized **QuickSort**: Recall

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

Randomized **QuickSort**

1. Pick a pivot element *uniformly at random* from $\mathbf{A}$.
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.

Randomized **QuickSort**: Recall

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

Randomized **QuickSort**

1. Pick a pivot element *uniformly at random* from $\mathbf{A}$.
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 $\mathbf{O(n \log n)}$ time in expectation. On *every* input it may take $\mathbf{\Omega(n^2)}$ time with some small probability.

Randomized **QuickSort**: Recall

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

Randomized QuickSort

1. Pick a pivot element *uniformly at random* from $\mathbf{A}$.
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 $\mathbf{O(n \log n)}$ time in expectation. On *every* input it may take $\mathbf{\Omega(n^2)}$ time with some small probability.
Question: With what probability it takes $\mathbf{O(n \log n)}$ time?

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

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

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

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

If $n = 100$ then this gives $\Pr[\mathbf{Q(A)} \leq 32 n \ln n] \geq 0.99999$.

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

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.

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

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 $1/n^4$.
  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 $\Pr[Q(A) \leq 32n \ln n] \geq 1 - 1/n^3$.

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 $1/n^4$.
  2. By union bound, any of the n elements participates in $> 32 \ln n$ levels w.p. at most $1/n^3$.

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

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 $1/n^4$.
  2. By union bound, any of the n elements participates in $> 32 \ln n$ levels w.p. at most $1/n^3$.
  3. Therefore, all elements participate in $\leq 32 \ln n$ w.p. $(1 - 1/n^3)$.

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 $3n/4$?

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

Intuition

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

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

Intuition

1. When we pick a pivot from an array of size $\mathbf{n}$ uniformly at random, what is the probability that its rank is between $\mathbf{n/4}$ and $\mathbf{3n/4}$? $\mathbf{1/2}$.
2. If we pick such a pivot then the size of L and R is at most?

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

Intuition

1. When we pick a pivot from an array of size $\mathbf{n}$ uniformly at random, what is the probability that its rank is between $\mathbf{n/4}$ and $\mathbf{3n/4}$? $\mathbf{1/2}$.
2. If we pick such a pivot then the size of L and R is at most? $\mathbf{3n/4}$. (Balanced split)

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

Intuition

1. When we pick a pivot from an array of size $\mathbf{n}$ uniformly at random, what is the probability that its rank is between $\mathbf{n/4}$ and $\mathbf{3n/4}$? $\mathbf{1/2}$.
2. If we pick such a pivot then the size of L and R is at most? $\mathbf{3n/4}$. (Balanced split)
3. If an array is reduced to at least its $\mathbf{3/4}$th size every time, then after how many rounds only one element remains?

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

Intuition

1. When we pick a pivot from an array of size $\mathbf{n}$ uniformly at random, what is the probability that its rank is between $\mathbf{n/4}$ and $\mathbf{3n/4}$? $\mathbf{1/2}$.
2. If we pick such a pivot then the size of L and R is at most? $\mathbf{3n/4}$. (Balanced split)
3. If an array is reduced to at least its $\mathbf{3/4}$th size every time, then after how many rounds only one element remains? $\leq \mathbf{4 \ln n}$.

Randomized **QuickSort**: High Probability Analysis

Informal Statement

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

Intuition

1. When we pick a pivot from an array of size $\mathbf{n}$ uniformly at random, what is the probability that its rank is between $\mathbf{n/4}$ and $\mathbf{3n/4}$? $\mathbf{1/2}$.
2. If we pick such a pivot then the size of L and R is at most? $\mathbf{3n/4}$. (Balanced split)
3. If an array is reduced to at least its $\mathbf{3/4}$th size every time, then after how many rounds only one element remains? $\leq \mathbf{4 \ln n}$.
4. If $\mathbf{32 \ln n}$ splits, then $\mathbf{E}[\text{Balanced-split}] = \mathbf{16 \ln n}$. Out of these there are $< \mathbf{4 \ln n}$ balanced split w.p. $\leq {}^{1}/_{n^4}$.

Randomized **QuickSort**: High Probability Analysis

- If **k** levels of recursion then **kn** comparisons.

Randomized **QuickSort**: High Probability Analysis

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

Randomized **QuickSort**: High Probability Analysis

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

- We call $\mathbf{s}$ lucky in $\mathbf{i^{th}}$ iteration, if *balanced split*: $\mathbf{|S_{i+1}| \leq (3/4)|S_i|}$ and $\mathbf{|S_i \setminus S_{i+1}| \leq (3/4)|S_i|}$.

Randomized **QuickSort**: High Probability Analysis

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

- We call $\mathbf{s}$ lucky in $\mathbf{i}^{\mathbf{th}}$ iteration, if *balanced split*: $\mathbf{|S_{i+1}| \leq (3/4)|S_i|}$ and $\mathbf{|S_i \setminus S_{i+1}| \leq (3/4)|S_i|}$.

- If $\rho =$#lucky rounds in first $\mathbf{k}$ rounds, then $\mathbf{|S_k| \leq (3/4)^{\rho} n.}$

Randomized **QuickSort**: High Probability Analysis

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

- We call $\mathbf{s}$ lucky in $\mathbf{i^{th}}$ iteration, if *balanced split*: $\mathbf{|S_{i+1}| \leq (3/4)|S_i|}$ and $\mathbf{|S_i \setminus S_{i+1}| \leq (3/4)|S_i|}$.

- If $\rho =$#lucky rounds in first $\mathbf{k}$ rounds, then $\mathbf{|S_k| \leq (3/4)^{\rho} n.}$

- For $\mathbf{|S_k| = 1}$, $\rho = \mathbf{4 \ln n \geq \log_{4/3} n}$ 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 $\mu = E[\rho] = \frac{k}{2}$.

How may rounds before $\mathbf{4 \ln n}$ 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 $\mu = \mathbf{E}[\rho] = \frac{k}{2}$.
- Set $k = 32 \ln n$ and $\delta = \frac{3}{4}$. $(1 - \delta) = \frac{1}{4}$.

How may rounds before $\mathbf{4 \ln n}$ 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 $\mu = E[\rho] = \frac{k}{2}$.
- 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^{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,

$$\begin{aligned} \Pr[\rho \leq 4 \ln n] &= \Pr[\rho \leq k/8] \\ &= \Pr[\rho \leq (1-\delta)\mu] \end{aligned}$$

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 $\mu = E[\rho] = \frac{k}{2}$.
- 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,

$$\begin{aligned}
\Pr[\rho \leq 4 \ln n] &= \Pr[\rho \leq {}^{k}/_{8}] \\
&= \Pr[\rho \leq (1-\delta)\mu] \\
\text{(Chernoff)} \quad &\leq e^{\frac{-\delta^2 \mu}{2}} \\
&= e^{-\frac{9k}{64}} \\
&= e^{-4.5 \ln n} \leq \frac{1}{n^4}
\end{aligned}$$

Randomized **QuickSort** w.h.p. Analysis

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

Randomized **QuickSort** w.h.p. Analysis

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

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 $\frac{1}{n^4} * n = \frac{1}{n^3}$.

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

Expected Bin Size

Problem

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

Solution

- Fix a bin, say $\mathbf{j}$.

Expected Bin Size

Problem

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

Solution

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

Expected Bin Size

Problem

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

Solution

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

Expected Bin Size

Problem

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

Solution

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

Expected Bin Size

Problem

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

Solution

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

Expected Bin Size

Problem

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

Solution

- Fix a bin, say $\mathbf{j}$.
- Random variable $\mathbf{X_{ij}}$ is $\mathbf{1}$ if $\mathbf{i}$th balls falls in $\mathbf{j}$th bin, otherwise $\mathbf{0}$.
- $\mathbf{E[X_{ij}] = Pr[X_{ij} = 1] = 1/n}$.
- R.V. $\mathbf{Y_j} = \#$ balls in $\mathbf{j}$th bin $= \sum_{i=1}^{n} \mathbf{X_{ij}}$.
- $\mathbf{E[Y_j] = \sum_{i=1}^{n} E[X_{ij}] = n \cdot 1/n = 1}$.

Expected Max Bin Size

Problem

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

Expected Max Bin Size

Problem

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

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

Expected Max Bin Size

Problem

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

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

Possible Solution

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

Expected Max Bin Size

Problem

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

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

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 $\mathbf{k}$ balls and at least one has $\mathbf{k}$ balls.

Expected Max Bin Size

Problem

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

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

Possible Solution

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

Problem

What is the expected maximum bin size?

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

Possible Solution

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

Expected Max Bin Size (Contd.)

Problem

What is the expected maximum bin size?

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

Possible Solution

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

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

Expected Max Bin Size (Contd.)

Problem

What is the expected maximum bin size?

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

Possible Solution

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

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

Bound $\mathbf{Pr}\left[\mathbf{Z > \frac{8\ln n}{\ln\ln n}}\right]$.

Expected Max Bin Size (Contd.)

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

Chernoff Ineq. We Saw

$X_1, \ldots, X_k$ independent binary R.V., and $X = \sum_{i=1}^{k} X_i$, $\mu = E[X]$, 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 $\Pr\left[Z > \frac{8 \ln n}{\ln \ln n}\right]$ using Chernoff inequality.

Chernoff Ineq. We Saw

$X_1, \ldots, X_k$ independent binary R.V., and $X = \sum_{i=1}^{k} X_i$, $\mu = E[X]$, 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 $\mathbf{Z = \max_{j=1}^{n} Y_j}$.

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

Expected Max Bin Size (Contd.)

Problem

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

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

Solution

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

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

Expected Max Bin Size (Contd.)

Problem

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

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

Solution

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

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

$$\mathbf{A^A} = \left(\frac{\mathbf{8 \ln n}}{\mathbf{\ln \ln n}}\right)^{\frac{8 \ln n}{\ln \ln n}} \geq (\sqrt{\mathbf{\ln n}})^{\frac{8 \ln n}{\ln \ln n}} = (\mathbf{\ln n})^{\frac{4 \ln n}{\ln \ln n}} = \mathbf{e^{4 \lg n}} = \mathbf{n^4}$$

Expected Max Bin Size (Contd.)

Problem

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

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

Solution

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

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

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

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

Expected Max Bin Size (Contd.)

Problem

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

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

Solution

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

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

Expected Max Bin Size (Contd.)

Problem

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

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

Solution

- Recall: $\mathbf{Y}_j = \#$ balls in bin $\mathbf{j}$. $\mathbf{E}[\mathbf{Y}_j] = 1$.

  $$\mathbf{Pr}[\mathbf{Y}_j > 8 \ln n / \ln \ln n] \leq 1/n^3 \qquad \text{(Using Chernoff)}$$

- (Union bound)
  $\mathbf{Pr}\left[\mathbf{Z} > \frac{8 \ln n}{\ln \ln n}\right] \leq \sum_{j=1}^{n} \mathbf{Pr}\left[\mathbf{Y}_j > \frac{8 \ln n}{\ln \ln n}\right] \leq n \cdot 1/n^3 = 1/n^2$.

Expected Max Bin Size (Contd.)

Problem

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

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

Solution

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

  $\mathbf{Pr[Y_j > 8 \ln n / \ln \ln n] \le 1/n^3}$ **(**Using Chernoff**)**

- (Union bound)

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

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

Expected Max Bin Size (Contd.)

Problem

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

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

Solution

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

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

- (Union bound)

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

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

$\mathbf{\Omega(\frac{\ln n}{\ln \ln n})}$ is a lower bound as well!

Balls n Bins $\longrightarrow$ Hashing

Hashing

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

Balls n Bins $\longrightarrow$ Hashing

Hashing

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

Throwing numbered balls

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

Balls n Bins $\longrightarrow$ Hashing

Hashing

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

Throwing numbered balls

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

Hashing: throw balls (elements) randomly into $\mathbf{n}$ bins such that **bin sizes are small**

Balls n Bins $\longrightarrow$ Hashing

Hashing

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

Throwing numbered balls

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

Hashing: throw balls (elements) randomly into $\mathbf{n}$ bins such that **bin sizes are small** and also **lookup is easy!**.

Part III

Hash Tables

Dictionary Data Structure

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

Dictionary Data Structures

Common solutions:

1. Static:
    1. Store $\mathbf{S}$ as a *sorted* array
    2. **Lookup**: Binary search in $\mathbf{O(\log |S|)}$ time (comparisons)
2. Dynamic:
    1. Store $\mathbf{S}$ in a *balanced* binary search tree
    2. Lookup, Insert, Delete in $\mathbf{O(\log |S|)}$ time (comparisons)

Dictionary Data Structures

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

Hashing and Hash Tables

Hash Table data structure:

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

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

Hashing and Hash Tables

Hash Table data structure:

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

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

Ideal situation:

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

Hashing and Hash Tables

Hash Table data structure:

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

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

Ideal situation:

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

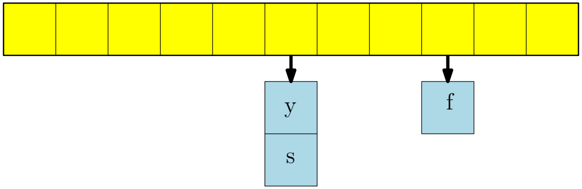
Collisions unavoidable if $|\mathbf{T}| < |\mathcal{U}|$. Several techniques to handle them.

Handling Collisions: Chaining

Collision: $\mathbf{h(x) = h(y)}$ for some $\mathbf{x \neq y}$.

Chaining to handle collisions:

1. For each slot $\mathbf{i}$ store all items hashed to slot $\mathbf{i}$ in a linked list. $\mathbf{T[i]}$ points to the linked list
2. **Lookup**: to find if $\mathbf{y} \in \mathcal{U}$ is in $\mathbf{T}$, check the linked list at $\mathbf{T[h(y)]}$. 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 $\mathbf{O(1)}$ time per operation for dictionaries?

Understanding Hashing

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

Questions:

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

Understanding Hashing

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

Understanding Hashing

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

Single hash function

1. $\mathcal{U}$: universe (very large).
2. Assume $\mathbf{N} = |\mathcal{U}| \gg \mathbf{m}$ where $\mathbf{m}$ is size of table $\mathbf{T}$. In particular assume $\mathbf{N} \geq \mathbf{m}^2$ (very conservative).
3. Fix hash function $\mathbf{h} : \mathcal{U} \rightarrow \{0, \ldots, m-1\}$.
4. $\mathbf{N}$ items hashed to $\mathbf{m}$ slots. By pigeon hole principle there is some $\mathbf{i} \in \{0, \ldots, m-1\}$ such that $\mathbf{N}/\mathbf{m} \geq \mathbf{m}$ elements of $\mathcal{U}$ get hashed to $\mathbf{i}$ (!).
5. Implies that there is a set $\mathbf{S} \subseteq \mathcal{U}$ where $|\mathbf{S}| = \mathbf{m}$ such that all of $\mathbf{S}$ hashes to same slot. Ooops.

Single hash function

1. $\mathcal{U}$: universe (very large).
2. Assume $\mathbf{N} = |\mathcal{U}| \gg \mathbf{m}$ where $\mathbf{m}$ is size of table $\mathbf{T}$. In particular assume $\mathbf{N} \geq \mathbf{m}^2$ (very conservative).
3. Fix hash function $\mathbf{h} : \mathcal{U} \rightarrow \{\mathbf{0}, \ldots, \mathbf{m} - \mathbf{1}\}$.
4. $\mathbf{N}$ items hashed to $\mathbf{m}$ slots. By pigeon hole principle there is some $\mathbf{i} \in \{\mathbf{0}, \ldots, \mathbf{m} - \mathbf{1}\}$ such that $\mathbf{N}/\mathbf{m} \geq \mathbf{m}$ elements of $\mathcal{U}$ get hashed to $\mathbf{i}$ (!).
5. Implies that there is a set $\mathbf{S} \subseteq \mathcal{U}$ where $|\mathbf{S}| = \mathbf{m}$ such that all of $\mathbf{S}$ 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) Um.

(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) $O\left(m^U\right)$ bits.

(E) $O(\log |\mathcal{H}|)$ bits.

(F) Many many bits. At least two.

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.

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

1. $\mathcal{H}$ is a **family** of hash functions: each function $\mathbf{h} \in \mathcal{H}$ should be efficient to evaluate (that is, to compute $\mathbf{h(x)}$).
2. $\mathbf{h}$ is chosen **randomly** from $\mathcal{H}$ (typically uniformly at random). Implicitly assumes that $\mathcal{H}$ allows an efficient sampling.
3. Randomized guarantee: should have the property that for any *fixed* set $\mathbf{S} \subseteq \mathcal{U}$ of size $\mathbf{m}$ the expected number of collisions for a function chosen from $\mathcal{H}$ should be "small". Here the expectation is over the randomness in choice of $\mathbf{h}$.

Picking a hash function

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

Picking a hash function

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

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

Picking a hash function

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

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 $\{0, 1, \ldots, m-1\}$?

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 $\mathcal{H}$?

1. 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?

Uniform hashing

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

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

Uniform hashing

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

1. Consider any element $\mathbf{x} \in \mathcal{U}$. Then if $\mathbf{h} \in \mathcal{H}$ is picked randomly then $\mathbf{x}$ should go into a random slot in $\mathbf{T}$. In other words $\Pr[\mathbf{h(x) = i}] = \mathbf{1/m}$ for every $\mathbf{0 \leq i < m}$. (Uniform)
2. Consider any two distinct elements $\mathbf{x, y} \in \mathcal{U}$. Then if $\mathbf{h} \in \mathcal{H}$ is picked randomly then the probability of a collision between $\mathbf{x}$ and $\mathbf{y}$ should be at most $\mathbf{1/m}$. In other words $\Pr[\mathbf{h(x) = h(y)}] = \mathbf{1/m}$ (cannot be smaller).

Uniform hashing

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

1. Consider any element $\mathbf{x} \in \mathcal{U}$. Then if $\mathbf{h} \in \mathcal{H}$ is picked randomly then $\mathbf{x}$ should go into a random slot in $\mathbf{T}$. In other words $\Pr[\mathbf{h(x) = i}] = \mathbf{1/m}$ for every $\mathbf{0 \leq i < m}$. (Uniform)
2. Consider any two distinct elements $\mathbf{x, y} \in \mathcal{U}$. Then if $\mathbf{h} \in \mathcal{H}$ is picked randomly then the probability of a collision between $\mathbf{x}$ and $\mathbf{y}$ should be at most $\mathbf{1/m}$. In other words $\Pr[\mathbf{h(x) = h(y)}] = \mathbf{1/m}$ (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 $\mathbf{x, y} \in \mathcal{U}$, $\Pr_{\mathbf{h}}[\mathbf{h(x) = h(y)}] = \mathbf{1/m}$ where $\mathbf{m}$ is the table size.

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 $0 \le i < m$. (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.

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

Analyzing Universal Hashing

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

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

Analyzing Universal Hashing

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

1. The time to look up $\mathbf{x}$ is the size of the list at $\mathbf{T[h(x)]}$: same as the number of elements in $\mathbf{S}$ that collide with $\mathbf{x}$ under $\mathbf{h}$.
2. Let $\ell(\mathbf{x})$ be this number. We want $\mathrm{E}[\ell(\mathbf{x})]$
3. For $\mathbf{y} \in \mathbf{S}$ let $\mathbf{A_y}$ be the event that $\mathbf{x}, \mathbf{y}$ collide and $\mathbf{D_y}$ be the corresponding indicator variable.

Analyzing Universal Hashing

Continued...

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

$$\begin{aligned}
\Rightarrow \mathrm{E}[\ell(\mathbf{x})] &= \sum_{\mathbf{y} \in \mathbf{S}} \mathrm{E}[\mathbf{D}_{\mathbf{y}}] \qquad \text{linearity of expectation} \\
&= \sum_{\mathbf{y} \in \mathbf{S}} \Pr[\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{y})] \\
&= \sum_{\mathbf{y} \in \mathbf{S}} \frac{1}{\mathbf{m}} \qquad \text{since } \mathcal{H} \text{ is a universal hash family} \\
&= |\mathbf{S}|/\mathbf{m} \\
&\le 1 \quad \text{if } |\mathbf{S}| \le \mathbf{m}
\end{aligned}$$

Analyzing Universal Hashing

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

Answer: $\mathbf{O(n/m)}$.

Analyzing Universal Hashing

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

Answer: $\mathbf{O(n/m)}$.

Comments:

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

Next Lecture

Desired Hash Family $\mathcal{H}$

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

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

Next Lecture

Desired Hash Family $\mathcal{H}$

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

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

1. $h_{a,b}$ can be evaluated in $O(1)$ time.
2. Easy to sample.
3. **Universal!**

Rehashing, amortization and...

... making the hash table dynamic

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

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 $\simeq m$ 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 $\simeq m$.

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 $\simeq m$ 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 $\mathbf{m}$ where $\mathbf{m}$ is some estimate of $|\mathbf{S}|$ (can be some large constant).
2. If $|\mathbf{S}|$ 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 $|\mathbf{S}|$ stays roughly the same but more than $\mathbf{c}|\mathbf{S}|$ operations on table for some chosen constant $\mathbf{c}$ (say $\mathbf{10}$), rebuild.

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