Good Afternoon.

Advanced algorithms and data structures

Lecture 5: Hashing

Jacob Holm (jaho@di.ku.dk)

December 2nd 2024

Today's Lecture

Hashing

- Hashing fundamentals
- Application: Unordered sets/Hashing with chaining
- Application: Signatures
- Practical hash functions
- Application: Coordinated sampling

Preliminaries

Notation:
For $n \in \mathbb{N}$:

$$[n] = \{0, \dots, n-1\}$$
$$[n]_+ = \{1, \dots, n-1\}$$

Iverson bracket:

$$[\text{condition}] = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{if condition is false} \end{cases}$$

For a random variable X:

$$\mu_X = \mathbb{E}[X] \quad \text{(expectation)}$$
$$\text{Var}[X] = \mathbb{E}[(X - \mu_X)^2] \quad \text{(variance)}$$
$$\sigma_X = \sqrt{\text{Var}[X]} \quad \text{(std. deviation)}$$

Inequalities:
Expectation of indicator variable X:

$$\mathbb{E}[X] = \Pr[X = 1]$$

Linearity of expectation:

$$\mathbb{E}\Big[\sum_i X_i\Big] = \sum_i \mathbb{E}[X_i]$$

Sum of pairwise indep. variances:

$$\text{Var}\Big[\sum_i X_i\Big] = \sum_i \text{Var}[X_i]$$

Union bound:

$$\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$$

Markovs Inequality: For $X \geq 0, t > 0$

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t} = \frac{\mu_X}{t}$$

Chebyshevs Inequality: For $t > 0$

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

AADS Lecture 5 (Hashing), Part 1

Hashing fundamentals

Hash function

Given a (typically large) universe U of keys, and a positive integer m.

Definition

A *random hash function* $h : U \to [m]$ is a randomly chosen function from some family of functions mapping U to $[m]$. Equivalently, it is a function h such that for each $x \in U$, $h(x) \in [m]$ is a random variable.

Cryptographic "hash functions" such as MD5, SHA-1, and SHA-256 are not *random* hash functions, and do not have most of the properties we want here. Do not confuse them!

Hash function

When discussing random hash functions, we usually care about

1. Space (*seed size*) needed to represent h.
2. Time needed to calculate $h(x)$ given $x \in U$.
3. Properties of the random variable.

Hash function types

Definition

A hash function $h : U \to [m]$ is *truly random* if the variables $h(x)$ for $x \in U$ are independent and uniform in $[m]$.

Impractical, why? Space! Require $|U| \log_2 m$ bits to represent.

Definition

A random hash function $h : U \to [m]$ is *universal* if, for all $x \neq y \in U$: $\Pr_h[h(x) = h(y)] \leq \frac{1}{m}$.

Definition

A random hash function $h : U \to [m]$ is *strongly universal* (a.k.a. 2-independent) if,

- Each key is hashed uniformly into $[m]$.
  (i.e. $\forall x \in U, q \in [m] : \Pr_h[h(x) = q] = \frac{1}{m}$)
- Any two distinct keys hash independently.

Or equivalently, if for all $x \neq y \in U$, and $q, r \in [m]$:
$\Pr_h[h(x) = q \wedge h(y) = r] = \frac{1}{m^2}$.

There are $m^{|U|}$ possible functions from U to $[m]$, so it takes at least $\log_2(m^{|U|}) = |U| \log_2 m$ bits to store which one we picked.

We use $\Pr_h[\cdots]$ or $\Pr_h[\cdots]$ instead of just $\Pr[\cdots]$ to make it clear that the probability is based on the random choice of h, and *not* on e.g. x, y being chosen at random.

For many purposes c-approximately universal hash functions for some small constant c are enough. We will see examples of such functions a little later today.

Hash function types

Definition

A hash function $h : U \to [m]$ is *truly random* if the variables $h(x)$ for $x \in U$ are independent and uniform in $[m]$.

Impractical, why? Space! Require $|U| \log_2 m$ bits to represent.

There are $m^{|U|}$ possible functions from U to $[m]$, so it takes at least $\log_2(m^{|U|}) = |U| \log_2 m$ bits to store which one we picked.

Definition

A random hash function $h : U \to [m]$ is *c-approximately universal* if, for all $x \neq y \in U$: $\Pr_h[h(x) = h(y)] \leq \frac{c}{m}$.

We use $\Pr_h[\cdots]$ or $\Pr_h[\cdots]$ instead of just $\Pr[\cdots]$ to make it clear that the probability is based on the random choice of h, and *not* on e.g. x, y being chosen at random.

Definition

A random hash function $h : U \to [m]$ is *c-approximately strongly universal* if,

- Each key is hashed *c*-approximately uniformly into $[m]$. (i.e. $\forall x \in U, q \in [m] : \Pr_h[h(x) = q] \leq \frac{c}{m}$)
- Any two distinct keys hash independently.

For many purposes *c*-approximately universal hash functions for some small constant c are enough. We will see examples of such functions a little later today.

Implying that for all $x \neq y \in U$, and $q, r \in [m]$:
$\Pr_h[h(x) = q \wedge h(y) = r] \leq ?$ (See Assignment 3 exercise 3.2).

AADS Lecture 5 (Hashing), Part 2

Application: Unordered sets/Hashing with chaining

Unordered sets/Dictionaries

Maintain a set S of at most n keys from some unordered universe U, under

INSERT(x, S) Insert key x into S.

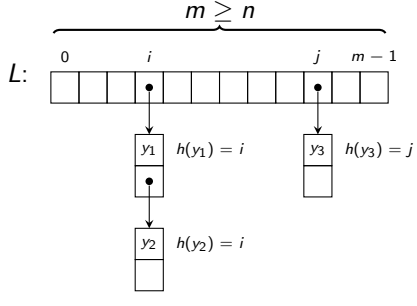
DELETE(x, S) Delete key x from S.

MEMBER(x, S) Return $x \in S$.

We could use some form of balanced tree to store S, but they usually take $\mathcal{O}(\log n)$ or $\mathcal{O}(\log \log U)$ time per operation, and we want each operation to run in expected constant time.

Hashing with chaining

Idea: Pick $m \geq n$ and a *universal* $h : U \to [m]$.
Store array L, where
$L[i] =$ linked list over $\{y \in S \mid h(y) = i\}$.



Then $x \in S \iff x \in L[h(x)]$.

Each operation takes $\mathcal{O}(|L[h(x)]| + 1)$ time.

Hashing with chaining

Theorem

For $x \notin S$, $\mathop{\mathbb{E}}_{h}\Big[\big|L[h(x)]\big|\Big] \leq \frac{n}{m} \qquad \Big(\leq 1\Big)$

Proof.

$$\begin{aligned}
\mathop{\mathbb{E}}_{h}\Big[\big|L[h(x)]\big|\Big] &= \mathop{\mathbb{E}}_{h}\Big[\big|\{y \in S \mid h(y) = h(x)\}\big|\Big] \\
&= \mathop{\mathbb{E}}_{h}\Big[\sum_{y \in S}[h(y) = h(x)]\Big] \\
&= \sum_{y \in S}\mathop{\mathbb{E}}_{h}\Big[[h(y) = h(x)]\Big] \\
&= \sum_{y \in S}\Pr_{h}[h(y) = h(x)] \\
&\leq |S|\tfrac{1}{m} \leq \tfrac{n}{m} \leq 1
\end{aligned}$$

□

By definition of $L[i] := \{y \in S \mid h(y) = i\}$.
Here we use the *Iverson Bracket* notation

$$[P] := \begin{cases} 1 & \text{if } P \text{ is true} \\ 0 & \text{if } P \text{ is false} \end{cases}$$

This can often be used as a shorthand for an indicator variable.
In this case $[h(y) = h(x)]$ becomes an indicator variable for the event $h(y) = h(x)$.
Linearity of expectation.
Expectation of indicator variable
Since $x \notin S$ and $y \in S$, we have $x \neq y$.
Then by definition of a universal hash function $h : U \to [m]$, $\Pr_h[h(y) = h(x)] \leq \frac{1}{m}$.

Hashing with chaining

Similarly, one can show that:

Theorem (Exercise)

For $x \in S$, $\mathbb{E}_h\Big[\big|L[h(x)]\big|\Big] \leq 1 + \frac{n-1}{m} \qquad \Big(\leq 2\Big)$

So we can conclude that each of the operations $\textsc{insert}(x, S)$, $\textsc{delete}(x, S)$, and $\textsc{member}(x, S)$ takes expected constant time when using hashing with chaining with a universal hash function.

AADS Lecture 5 (Hashing), Part 3

Application: Signatures

Application: Signatures

Problem: Assign a unique "signature" to each $x \in S \subseteq U$, $|S| = n$.

Solution: Use universal hash function $s : U \to [n^3]$.

Then we fail (by having a collision) with probability

$$\begin{aligned}
&\Pr_s[\exists \{x, y\} \subseteq S \text{ with } s(x) = s(y)] \\
&\quad = \Pr_s\left[\bigcup_{\{x,y\}\subseteq S} \text{event } s(x) = s(y)\right] \\
&\quad \leq \sum_{\{x,y\}\subseteq S} \Pr_s[s(x) = s(y)] && \text{(Union bound)} \\
&\quad \leq \frac{\binom{n}{2}}{n^3} && (s \text{ universal}) \\
&\quad < \frac{1}{2n}
\end{aligned}$$

Thus with "high probability" we have no collisions.

It is tempting to try to calculate a probability of no collisions more directly as a product over all pairs of the probability of that pair not colliding, i.e. as:

$$\begin{aligned}
\prod_{\{x,y\}\in S} \Pr_s[s(x) \neq s(y)] &\geq \left(1 - \frac{1}{n^3}\right)^{\binom{n}{2}} \\
&= \left(1 + \frac{1}{n^3 - 1}\right)^{-\binom{n}{2}} \\
&\geq e^{\left(\frac{-\binom{n}{2}}{n^3-1}\right)} \\
&\geq e^{-\frac{1}{2n}} \\
&> 1 - \frac{1}{2n}
\end{aligned}$$

While this happens to give the right answer in this case, the calculation is invalid because the events are not independent.

We use the union bound instead because that does not require independence.

AADS Lecture 5 (Hashing), Part 4

Practical hash functions

Multiply-mod-prime

Let $U = [u]$ and pick prime $p \geq u$. For any $a, b \in [p]$, and $m < u$, let $h^m_{a,b} : U \to [m]$ be

$$h^m_{a,b}(x) = ((ax + b) \bmod p) \bmod m$$

Is this a random hash function? NO!

Multiply-mod-prime

Let $U = [u]$ and pick prime $p \geq u$. For any $a, b \in [p]$, and $m < u$, let $h^m_{a,b} : U \to [m]$ be

$$h^m_{a,b}(x) = ((ax + b) \bmod p) \bmod m$$

Choose $a, b \in [p]$ independently and uniformly at random, and let $h(x) := h^m_{a,b}(x)$.
Then $h : U \to [m]$ is a 2-approximately strongly universal hash function.

Multiply-shift

Let $U = [2^w]$ and $m = 2^\ell$. For any odd $a \in [2^w]$ define

$$h_a(x) := \left\lfloor \frac{(ax) \bmod 2^w}{2^{w-\ell}} \right\rfloor$$

Choose odd $a \in [2^w]$ uniformly at random, and let $h(x) := h_a(x)$.
Then $h : U \to [m]$ is a 2-approximately universal hash function.

(Assignment 3 exercise 3.4 asks you to show that it is not c-approximately strongly universal for any constant c).

Multiply-shift, C

For $U = [2^{64}]$ the C code looks like this:

```
#include<stdint.h>
uint64_t hash(uint64_t x, uint64_t l, uint64_t a)
{
  return (a*x) >> (64-l);
}
```

Reminder: Strongly universal

Recall:

Definition

A random hash function $h : U \to [m]$ is *strongly universal* (a.k.a. 2-independent) if,

- Each key is hashed uniformly into $[m]$.
  (i.e. $\forall x \in U, q \in [m] : \Pr_h[h(x) = q] = \frac{1}{m}$)
- Any two distinct keys hash independently.

Reminder: Strongly universal

Recall:

Definition

A random hash function $h : U \to [m]$ is *c-approximately strongly universal* if,

- Each key is hashed *c*-approximately uniformly into $[m]$.
  (i.e. $\forall x \in U, q \in [m] : \Pr_h[h(x) = q] \leq \frac{c}{m}$)
- Any two distinct keys hash independently.

Strong Multiply-shift

Let $U = [2^w]$ and $m = 2^\ell$, and pick $\bar{w} \geq w + \ell - 1$. For any pair $(a, b) \in [2^{\bar{w}}]^2$ define

$$h_{a,b}(x) := \left\lfloor \frac{(ax + b) \bmod 2^{\bar{w}}}{2^{\bar{w}-\ell}} \right\rfloor$$

Choose $a, b \in [2^{\bar{w}}]$ independently and uniformly at random, and let $h(x) := h_{a,b}(x)$.
Then $h : U \to [m]$ is a strongly universal hash function.

Strong Multiply-shift, C

For $\ell \leq w = 32$ and $\bar{w} = 64$ we have $U = [2^{32}]$ and the C code looks like this:

```
#include<stdint.h>
uint32_t hash(uint32_t x, uint32_t l,
              uint64_t a, uint64_t b)
{
  return (a*x+b) >> (64-l);
}
```

AADS Lecture 5 (Hashing), Part 5

Application: Coordinated sampling

Application: Coordinated sampling

Suppose we have a bunch of *agents* that each observe some set of events from some universe U. Let $A_i \subseteq U$ denote the set of events seen by agent i, and suppose $|A_i|$ is large so only a small sample $S_i \subseteq A_i$ is actually stored.

At some later point we are interested in a subset $A^\star \subseteq \bigcup_i A_i$ (not known in advance), and in particular we want to estimate the size $|A^\star|$.

Application: Coordinated sampling

If each agent independently just samples a random subset of the seen events, there is very little chance that two agents that see an event make the same decision.
$\Longrightarrow$ The samples are incomparable (almost useless).

Coordinated sampling means that all agents that see an event make the same decision about whether to store it.
$\Longrightarrow$ Samples can be combined, i.e.

- $S_i \cup S_j$ is a sample of $A_i \cup A_j$
- $S_i \cap S_j$ is a sample of $A_i \cap A_j$

In particular, $A^\star \cap \bigcup_i S_i$ is a sample of $A^\star$ and can be computed from $\bigcup_i S_i$ just by determining for each element whether it belongs to $A^\star$.

Application: Coordinated sampling

Let $h : U \to [m]$ be a strongly universal hash function, and let $t \in \{0, \ldots, m\}$. Send h and t to all the agents.

Each agent samples $x \in U$ iff $h(x) < t$.

Thus if an agent sees the set $A_i \subseteq U$, the set $S_{h,t}(A_i) := \{x \in A_i \mid h(x) < t\}$ is sampled. Note that

- $S_{h,t}(A_i) \cup S_{h,t}(A_j) = S_{h,t}(A_i \cup A_j)$
- $S_{h,t}(A_i) \cap S_{h,t}(A_j) = S_{h,t}(A_i \cap A_j)$

Each $x \in U$ (if seen) is sampled with probability $\Pr_h[h(x) < t] = \frac{t}{m}$. Why? Strong universality $\implies$ $h(x)$ uniform in $[m]$

For any $A \subseteq U$, $\mathbb{E}_h[|S_{h,t}(A)|] = |A| \cdot \frac{t}{m}$.

Thus we have an unbiased estimate $|A^\star| \approx \frac{m}{t} \cdot |S_{h,t}(A^\star)| = \frac{m}{t} \cdot |A^\star \cap \bigcup_i S_i|$.

How good is this estimate, i.e. what can we say about the *relative error* $:= \left|\frac{\text{estimated value} - \text{actual value}}{\text{actual value}}\right| = \left|\frac{\text{estimated value}}{\text{actual value}} - 1\right|$?

$$\begin{aligned}
\mathop{\mathbb{E}}_h\big[|S_{h,t}(A)|\big] &= \mathop{\mathbb{E}}_h\left[\sum_{x \in A} [h(x) < t]\right] \\
&= \sum_{x \in A} \mathop{\mathbb{E}}_h\big[[h(x) < t]\big] \\
&= \sum_{x \in A} \Pr_h[h(x) < t] \\
&= \sum_{x \in A} \tfrac{t}{m} \\
&= |A| \cdot \tfrac{t}{m}
\end{aligned}$$

Concentration bound

Lemma

Let $X = \sum_{a \in A} X_a$ where the X_a are pairwise independent 0–1 variables. Let $\mu = \mathbb{E}[X]$. Then $\mathrm{Var}[X] \leq \mu$, and for any $q > 0$,

$$\Pr[|X - \mu| \geq q\sqrt{\mu}] \leq \tfrac{1}{q^2}$$

Proof (not curriculum).

For $a \in A$ let $p_a = \Pr[X_a = 1]$. Then $p_a = \mathbb{E}[X_a]$ and

$$\begin{aligned}\mathrm{Var}[X_a] &= \mathbb{E}[(X_a - p_a)^2] = (1 - p_a)(0 - p_a)^2 + p_a(1 - p_a)^2 \\ &= (p_a^2 + p_a(1 - p_a))(1 - p_a) = p_a(1 - p_a) \leq p_a \\ \mathrm{Var}[X] &= \mathrm{Var}\Big[\sum_{a \in A} X_a\Big] = \sum_{a \in A} \mathrm{Var}[X_a] \leq \sum_{a \in A} p_a = \mu\end{aligned}$$

Finally, since $\sigma_X = \sqrt{\mathrm{Var}[X]} \leq \sqrt{\mu}$ we get:

$$\begin{aligned}\Pr[|X - \mu| \geq q\sqrt{\mu}] &\leq \Pr[|X - \mu| \geq q\sigma_X] \\ &\leq \tfrac{1}{q^2} \qquad \text{(Chebyshev's ineq.)}\ \square\end{aligned}$$

Application: Coordinated sampling

Let's apply this lemma to the estimate $|A^\star| \approx \frac{m}{t}|S_{h,t}(A^\star)|$ from our coordinated sampling.

Let $X = |S_{h,t}(A^\star)|$ and for $a \in A^\star$ let $X_a = [h(a) < t]$. Then $X = \sum_{a \in A^\star} X_a$ and for any $a, b \in A^\star$, X_a and X_b are independent. Also, let $\mu = \mathbb{E}_h[X] = \frac{t}{m}|A^\star|$.

Then for any $q > 0$,

$$\begin{aligned}
&\Pr_h\left[\left|\frac{\frac{m}{t}|S_{h,t}(A^\star)|}{|A^\star|} - 1\right| \geq q \cdot \frac{1}{\sqrt{\frac{t}{m}|A^\star|}}\right] \\
&\qquad = \Pr_h\left[\left||S_{h,t}(A^\star)| - \tfrac{t}{m}|A^\star|\right| \geq q \cdot \sqrt{\tfrac{t}{m}|A^\star|}\right] \\
&\qquad = \Pr_h\left[|X - \mu| \geq q \cdot \sqrt{\mu}\right] \\
&\qquad \leq \tfrac{1}{q^2}
\end{aligned}$$

We needed strong universality in two places for this to work. Where? h must be uniform to get unbiased estimate, and pairwise independent for the lemma.

Summary

Todays topic was hashing, and we have covered

- What is a random hash function, and what properties do we want.
- Two applications of universal hashing — unordered sets and signatures.
- Some concrete universal or strongly universal hash functions.
- An application of strongly universal hashing — coordinated sampling.
- Next time: Computational complexity, P, NP, and NP-completeness with Jakob Nordström.
- Later: An ordered set data structure that is not comparison based, and an application of hash tables.