Design and Analysis of Algorithms I

Data Structures

Universal Hash Functions: Performance Guarantees (Chaining)

Overview of Universal Hashing

Next: details on randomized solution (in 3 parts).

Part I: proposed definition of a "good random hash function". ("universal family of hash functions")

Part II: Concrete example of simple + practical such functions

Part III: justification of definition: "good functions" lead to "good performance".

Universal Hash Functions

Definition: Let $\mathcal{H}$ be a set of hash functions from U to $\{0,1,2,\dots,n-1\}$.

$\mathcal{H}$ is universal if and only if:

for all $x,y \in U$ (with $x \neq y$)

$$\Pr_{h \in \mathcal{H}}\left[\begin{array}{c} x,y \text{ collide} \\ (h(x)=h(y)) \end{array}\right] \leq \frac{1}{n}$$

(n = # of buckets)

when h is chosen uniformly at random from $\mathcal{H}$.

(i.e., collision probability as small as with "gold standard" of perfectly random hashing)

Tim Roughgarden

Chaining: Constant-Time Guarantee

Scenario: hash table implemented with chaining. Hash function h chosen uniformly at random from universal family $\mathcal{H}$.

Theorem: [Carter-Wegman 1979] All operations run in $O(1)$ time. (for every data set S)

Caveats: ① in expectation over the random choice of the hash function h.
② assumes $|S| = O(n)$ [i.e., load $\alpha = \frac{|S|}{n} = O(1)$] ($n$ = # of buckets)
③ assumes takes $O(1)$ time to evaluate hash function

Proof (Part I)

Will analyze an unsuccessful Lookup (other operations only faster).

# of buckets

So: let S = data set with $|S| = O(n)$.

(arbitrary data set S)

Consider Lookup for $x \notin S$.

Running Time: $O(1) + O(\text{list length in } A[h(x)])$

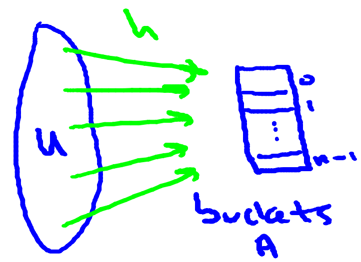
compute $h(x)$

traverse list

(a random variable, depends on hash function h)

Tim Roughgarden

A General Decomposition Principle

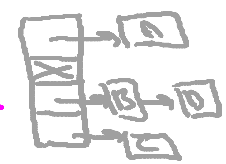


Collision: distinct $x, y \in U$ such that $h(x) = h(y)$.

Solution #1: (separate) chaining.

- keep linked list in each bucket
- given a key/object x, perform Insert/Delete/Lookup in the list in $A[h(x)]$

bucket for x — linked list for x



Solution #2: open addressing. (only one object per bucket)

- hash function now specifies probe sequence $h_1(x), h_2(x), \ldots$ (keep trying til find open slot)
- examples: linear probing (look consecutively), double hashing (use 2 hash functions)

Tim Roughgarden

Proof (Part II)

Let L = list length in $A[h(x)]$.

For $y \in S$ (so $y \neq x$), define $Z_y = \begin{cases} 1 & \text{if } h(y) = h(x) \\ 0 & \text{otherwise} \end{cases}$

<u>Note</u>: $L = \sum_{y \in S} Z_y$.

<u>So</u>: $E[L] \underset{\text{EXP}}{\overset{\text{LIN}}{=}} \sum_{y \in S} E[Z_y]$

What does $\mathrm{E}[Z_y]$ evaluate to?

- $\Pr[h(y) = 0]$
- $\Pr[h(y) \neq x]$
- $\Pr[h(y) = h(x)]$
- $\Pr[h(y) \neq h(x)]$

<u>Recall</u>

$$Z_y = \begin{cases} 1 & \text{if } h(y) = h(x) \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{E}[Z_y] = \cancel{0 \cdot \Pr[Z_y = 0]}^{\,0} + 1 \cdot \Pr[Z_y = 1]$$

$$= \Pr[h(y) = h(x)]$$

Proof (Part II)

Let L = list length in $A[h(x)]$.

For $y \in S$ (so $y \neq x$), define $Z_y = \begin{cases} 1 & \text{if } h(y) = h(x) \\ 0 & \text{otherwise} \end{cases}$

<u>Note:</u> $L = \sum_{y \in S} Z_y$.

<u>So:</u> $E[L] \overset{\text{LIN EXP}}{=} \sum_{y \in S} E[Z_y] = \sum_{y \in S} \Pr[h(y) = h(x)]$

Which of the following is the smallest valid upper bound on $\Pr[h(y) = h(x)]$?

- $1/n^2$
- $1/n$
- $1/2$
- $1 - 1/n$

by defintion of a universal family of hash functions

Proof (Part II)

Let L = list length in $A[h(x)]$.

For $y \in S$ (so $y \neq x$), define $Z_y = \begin{cases} 1 & \text{if } h(y)=h(x) \\ 0 & \text{otherwise} \end{cases}$

Note: $L = \sum_{y \in S} Z_y$.

So: $E[L] \overset{\text{LIN EXP}}{=} \sum_{y \in S} E[Z_y] = \sum_{y \in S} \underbrace{Pr[h(y)=h(x)]}_{\leq \frac{1}{n}}$

since $\mathcal{H}$ universal $\longrightarrow$ $\leq \sum_{y \in S} \frac{1}{n}$ provided $|S| = O(n)$

$= \frac{|S|}{n}$ = load α $= O(1)$.

QED!