- How to create good hash functions

Choosing our Hash Function

What do we want from our hash function?

- distribute keys evenly into our table slots

- should not depend on our keys being random

✗ dont want patterns in data to cause patterns in hash function

Division Method

Let $h(k) = k \bmod m$. // $0 \leq h(k) \leq m-1$ (even distribution)

Thus we'll hash a key k into a table with m slots using the slot given by the remainder of k divided by m.

Suppose we had 12 slots and a key of 100:

=> $h(100) = 100 \bmod 12 = 4$ // 100 goes to slot 4 of table

- Advantage: fast
- Disadvantage: value of m is critical (Table size matters a lot)

How are adjacent keys mapped?

=> $h(101) = 101 \bmod 12 = 5$

$h(102) = 102 \bmod 12 = 6$

$h(103) = \ldots = 7$

What if m is a power of 10 (say 1000)?

=> 419,200 / 57,200, 1,256,200] These all go into slot 200

=> Only last 3 digits really matter, and this isnt a good thing as everything will fall into the same bucket

✗ Dont choose # that is power of 10, or power of 2 for similar reasons [Only pick prime #s]

Pick m = prime number not too close to power of 2 (or 10)

The Multiplication Method // designer decides A value

For a constant A where $0 < A < 1$: → A is not a whole #

$h(k) = \lfloor m(kA \bmod 1) \rfloor$

Same as $h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$ → $0 \leq h(k) < m-1$

- Advantage: Value of m is not critical
- Disadvantage: relatively slower
  → Also, must pick A

Since the value of m is not critical, we can just pick $m = 2^p$ for some p. (i.e., the number of slots in our table is a power of 2)

How do we choose A?

- Any legal value will create a hash function but some are better than others.

What if we choose $A = 0.00001$? $m = 12$

$h(100) = \lfloor 12(100 \cdot 0.00001 - \lfloor 100 \cdot 0.00001 \rfloor) \rfloor$

$= \lfloor 12(0.001 - \lfloor 0.001 \rfloor) \rfloor$ ($\lfloor 0.001 \rfloor = 0$)

$= \lfloor 12(0.001) \rfloor = \lfloor 0.012 \rfloor = 0$

∴ key 100 goes into slot 0

$h(200) = \lfloor 12(0.002) \rfloor = \lfloor 0.024 \rfloor = 0$

∴ key 200 goes into slot 0

=> all of these go to slot 0: 101, 102, 103, ..., h(1000)

- Choose A not too close to 0 and 1.
- Good choice for $A = (\sqrt{5} - 1)/2$:
  → solution to the [golden ratio]
  - Appears in packing problems (like sunflower)

=> Sunflowers use this golden ratio formula to pack as many seeds in its head

The Multiplication Method - Implementation

- Choose $m = 2^p$ for some integer p.
- Let the word size of the machine be w bits. Assume k fits into a single word.

  Let s be a number such that $0 < s < 2^w$.
- We restrict A to be a fraction of the form $s/2^w$. Thus, $s = A2^w$

  Let $ks = r_1 2^w + r_0$.

  r_1 is the integer part of kA, that is $\lfloor kA \rfloor$, and is the fractional part of kA, that is $kA - \lfloor kA \rfloor$.

  So just use r_0

// done on hw #3

Hash Functions

— What if an adversary knows what our hash function? (depends)

- could map keys all into the same slot, thus anytime a search of keys operation is performed we would create $\Theta(n)$

=> (an be a denial of service attack
↳ causes system to run very slow (DDOS)

Universal Hashing (groups of hash functions)

- Idea: Randomize the algorithm
- Specifically, we can pick hash function randomly (which still performs well)
  - Pick hash function when hash table when hash table algorithm first starts
  - This hash function should give good performance on average
  - So need a family of good hash functions to choose from

Let H be a (finite) collection of hash functions:

- Map from a given universe U ↴
- Map into a given range $\{0, 1, \ldots, m-1\}$

H is universal if:

- For each pair of distinct keys $x, y \in U$ the number of hash functions $h \in H$ for which $h(x) = h(y)$ is at most $\frac{|H|}{m}$ (mapped to same location)
- In other words, with a random hash function from H the chance of a collision between $x, y \in U$ (where $x \neq y$) is at most $\frac{1}{m}$ ↳ random chance

- Choose hash function h from H
- Hash n keys into a table of m slots where $n \leq m$
- How many collisions do we expect to get?

$x \; != \; y$

For each pair of keys, x and y, let the RV $C_{xy} = 1$ if x and y collide. Otherwise, let $C_{xy} = 0$.

$$E(C_{xy}) = p(C_{xy} = 1) \cdot 1 + p(C_{xy} = 0) \cdot 0$$

$= 0$

$$= \frac{1}{m} \cdot 1 = \frac{1}{m}$$

$\frac{1}{m}$ ← from previous page

Let the RV C_x be the total number of collisions involving key x.

for every key $!= x$ see if they collide

$$\Rightarrow E(C_x) = E\left(\sum_{\text{for all } y \neq x} C_{xy}\right) = \sum_{\forall y \neq x} E(C_{xy})$$

$\Rightarrow$ for every key $\neq x$ see if they collide

$$= \sum_{\forall y \neq x} \frac{1}{m} = \frac{n-1}{m} \in \Theta(1)$$

$n-1$ ← # keys $\neq x$

$\therefore \; n \leq m$

- Since $n \leq m$ we have $E(C_x) \leq 1$ (i.e., $E(C_x) \in \Theta(1)$)
- Thus, our expected runtime for insert, delete, and search are $\Theta(1)$

So universal hash functions are great. How can we create one?

A Universal Hash Function

range of keys

- Choose a prime number p that is larger than all possible keys.
- Choose a table size $m \geq n$ (n is the number of keys)
- Randomly choose two integers a, b such that $1 \leq a \leq p-1$ and $1 \leq b \leq p-1$
- Our hash function is $h_{a,b} = ((ak + b) \bmod p) \bmod m$
- Example: $p = 17$, $m = 6$, $a = 3$, $b = 4$
- Compute $h_{3,4}(8)$ (i.e., 8 is a key)

$\Rightarrow h_{3,4}(8) = ((3 \cdot 8 + 4) \bmod 17) \bmod 6$

$= (28 \bmod 17) \bmod 6$

$= 11 \bmod 6 = 5$ = key 8 goes into slot 8

Previous hash function will be universal

Proof sketch:

m slots: 0 1 2 3 4 5 ...

- For any distinct keys, x and y, given hash function $h_{a,b}$ ($x \neq y$)
- Let $r = (ax + b) \bmod p$ and $s = (ay + b) \bmod p$ // everything prior to mod by m
- Can be shown that $r \neq s$ and different a, b results in a different (r, s) pair
- Thus, x and y collide only when $r \bmod m = s \bmod m$ ($0 \leq s \leq p-1$)
- For a given r, the number of s values such that $r \bmod m = s \bmod m$ is at most $\frac{p-1}{m}$.

$\frac{p-1}{m} =$ # values s can take − one value taken by r / # slots to map to (m)

- For a given r, what is the chance a random s gives us one of these values? => # bad values / total # values available $= \frac{\frac{p-1}{m}}{p-1} = \frac{1}{m}$

(values that cause collisions)

=> This is chance x & y collide $= \frac{1}{m}$