Hashing (1)

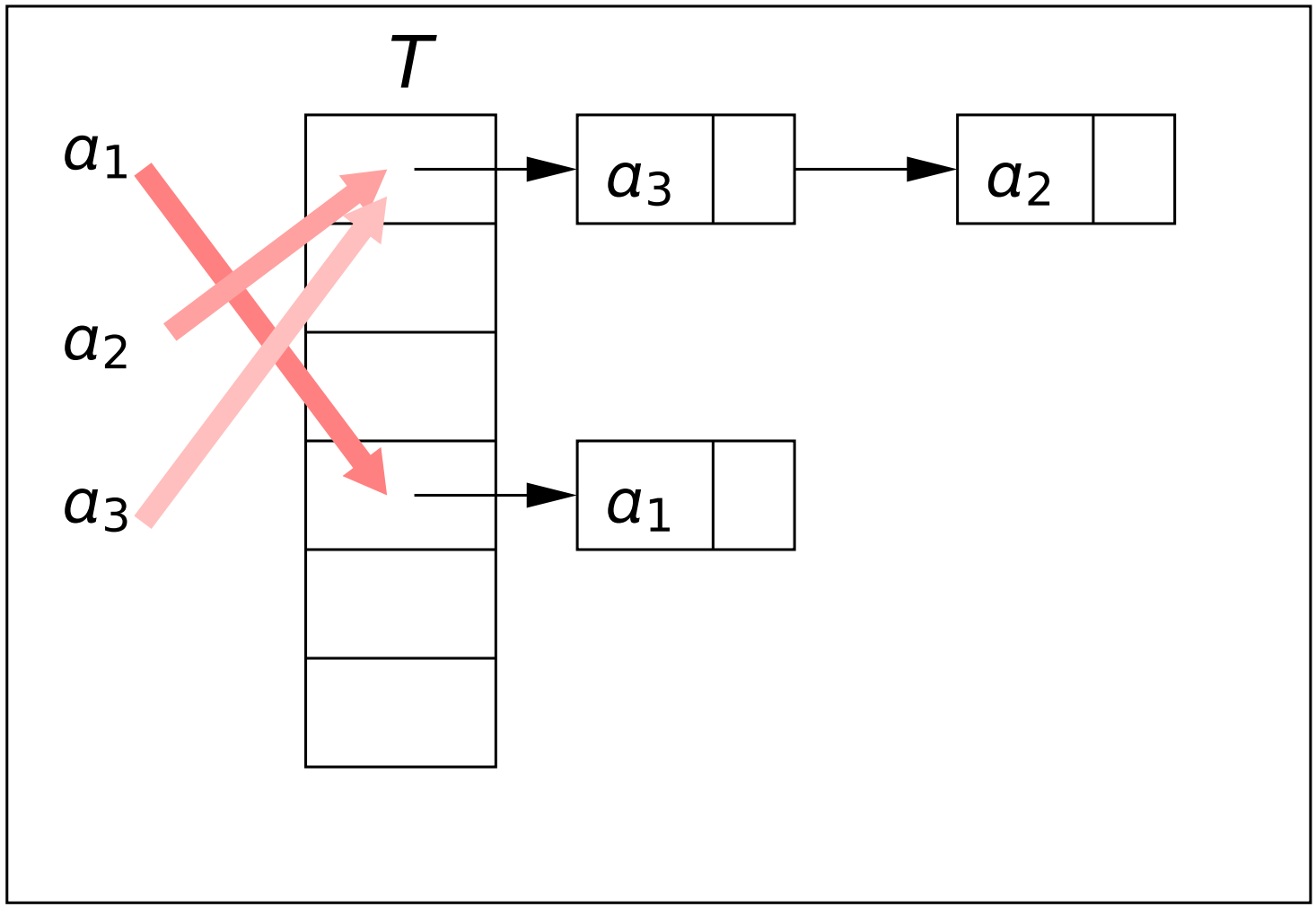
The general setup:

- $\mathcal{U}$ – (large, finite) universe of possible **keys**
- $\mathcal{V}$ – (small) set of **slots** of size m
  typically $\mathcal{V} = [0\,..\,m)$
- $h : \mathcal{U} \to \mathcal{V}$ – a "hash function" from $\mathcal{U}$ to $\mathcal{V}$
  maps keys to slots
- $T[\mathcal{V}]$ – a "hash table" for storing keys, indexed by $\mathcal{V}$

Implementing a dictionary:

- A key $a \in \mathcal{U}$ is stored in the hash table T at slot $s = h(a)$
- As long as no two keys hash to the same slot (a "collision"), we can perform all dictionary operations (*insert*, *search*, *delete*) in **constant time**

Resolving collisions by chaining



Dictionary Operations:

- *search*(a): search for a in $T[h(a)]$
- *insert*(a): search for a in $T[h(a)]$ and insert if not present
- *delete*(a): search for a in $T[h(a)]$ and delete if present

Running times: $O(n)$ (worst case)

Worst case occurs when all keys hash to the same slot

Example. Suppose $\mathcal{U}$ (the universe of possible keys) is the set of all ASCII character strings of fixed length t

So each key is a tuple $a = (a_1, \ldots, a_t)$, with each $a_i \in [0\,..\,256)$

Set $m := 256$, $\mathcal{V} = [0\,..\,256)$, and define $h(a) := \sum_i a_i \bmod m \in \mathcal{V}$

- For example, if $t = 4$, the string "abcd" hashes to the slot $97 + 98 + 99 + 100 = 394 \mod 256 = 138$

This is *not* a very good hash function

- It's easy to get lots of keys that hash to the same slot
- For example, the strings "abcd", "dcba", "badc", . . . all hash to slot 138

Better to use a more "complicated" hash function

But . . . for any *fixed* hash function, a "bad guy" can always cook up a large set of keys that *all* hash to the same slot

- For example, if $m = 256$, the "bad guy" can just hash 256,000 keys: one slot must contain at least 1000 keys, so the "bad guy" chooses these 1000 keys

Solution: randomization! The slot is computed as $s \leftarrow h(\lambda, a)$, where λ is a "seed"

We will choose the seed at random *once* when we initialize the hash table, and then hash all keys *using the same seed*

With luck, this will prevent pile ups from occurring, no matter how cleverly or maliciously the "bad guy" chooses the keys . . . as long as the "bad guy" does not know the seed

Assume λ is chosen from some seed space Λ

Notation: $h_\lambda(a) = h(\lambda, a)$

$\{h_\lambda\}_{\lambda \in \Lambda}$ is a **family** of hash functions, each mapping $a \in \mathcal{U}$ to $s = h_\lambda(a) = h(\lambda, a) \in \mathcal{V}$

Each seed $\lambda \in \Lambda$ indexes (or specifies) a particular hash function h_λ in the family

Universal Hashing [Carter & Wegman, 1975]

- Λ – a finite, non-empty set of **hash function seeds**
- $\mathcal{H} = \{h_\lambda\}_{\lambda \in \Lambda}$ – a **family** of hash functions from $\mathcal{U}$ to $\mathcal{V}$, indexed by seeds $\lambda \in \Lambda$
- $m := |\mathcal{V}|$

Def'n: $\mathcal{H}$ is called **universal** if for all $a, b \in \mathcal{U}$ with $a \neq b$,

$$\left|\{\lambda \in \Lambda : h_\lambda(a) = h_\lambda(b)\}\right| \leq \frac{|\Lambda|}{m}$$

Probabilistic interpretation: if R is a random variable, uniformly distributed over Λ, then

$$\Pr[h_R(a) = h_R(b)] \leq \frac{1}{m}$$

We will show (later): Suppose we implement a hash table using such a family and a randomly chosen seed. Then:

- if we perform any sequence of hash table operations
- and the number of keys in the table is always $O(m)$,
- then the *expected* cost per hash table operation is $O(1)$

A universal family: inner product hash

Let m be a prime, and t a positive integer

Define $\mathcal{U} := \mathbb{Z}_m^t$, $\Lambda := \mathbb{Z}_m^t$, $\mathcal{V} := \mathbb{Z}_m$

For $\lambda = (\lambda_1, \ldots, \lambda_t) \in \Lambda$, $a = (a_1, \ldots, a_t) \in \mathcal{U}$, define

$$h_\lambda(a) := \sum_{i=1}^{t} a_i \lambda_i$$

Define

$$\mathcal{H} := \{h_\lambda\}_{\lambda \in \Lambda}$$

Theorem: $\mathcal{H}$ is universal

Proof of theorem.

Suppose $(a_1, \ldots, a_t) \neq (b_1, \ldots, b_t)$

We want to count the number N of solutions $(\lambda_1, \ldots, \lambda_t)$ to the equation

$$\sum_i a_i \lambda_i = \sum_i b_i \lambda_i$$

Re-write this as

$$\sum_i c_i \lambda_i = 0$$

where $c_i := a_i - b_i$

By assumption, not all c_i's are zero

Want to show: $N \leq |\Lambda|/m = m^{t-1}$

Proof (cont'd).

Let N be the number of solutions $(\lambda_1, \ldots, \lambda_t)$ to $\sum_i c_i \lambda_i = 0$, where some $c_j \neq 0$

Want to show: $N \leq |\Lambda|/m = m^{t-1}$

Without loss of generality, assume $c_1 \neq 0$

For every choice of $\lambda_2, \ldots, \lambda_t$, there is a unique λ_1 such that $\sum_i c_i \lambda_i = 0$, namely, $\lambda_1 = -c_1^{-1} \sum_{i=2}^{t} c_i \lambda_i$

There are m^{t-1} ways of choosing $\lambda_2, \ldots, \lambda_t$, and each yields one solution

So $N = m^{t-1}$

QED

Practical considerations

Key space:

- Suppose keys are ASCII character strings of fixed length t
- So each key is a tuple $(a_1, \ldots, a_t)$, with each $a_i \in [0\,..\,256)$
- Choose a prime $m > 256$, so $a_i \bmod m = a_i$
- Variable length keys (padding)

Practical considerations (cont'd)

Slot space: table size must be a prime m

If there are n keys in the table, we want to choose m so that the ratio n/m is not too large

Dynamically growing the table: when n/m gets too large, choose a new $m' \approx 2m$, and rehash everything

Bertrand's Postulate: There is always a prime between x and $2x$ for all integers $x \geq 1$

Chebyshev said it
So I'll say it again
There's always a prime between N and 2N

ϵ-universal Hashing

General problem: large seed space

Solution: weaker (but still useful) hashing requirements

Let $\mathcal{H} = \{h_\lambda\}_{\lambda \in \Lambda}$ be a family of hash functions from $\mathcal{U}$ to $\mathcal{V}$

Def'n: Let $0 \leq \epsilon \leq 1$. $\mathcal{H}$ is called **ϵ-universal** if for all $a, b \in \mathcal{U}$ with $a \neq b$,

$$\left|\{\lambda \in \Lambda : h_\lambda(a) = h_\lambda(b)\}\right| \leq \epsilon \cdot |\Lambda|$$

Probabilistic interpretation: if R is a random variable, uniformly distributed over Λ, then

$$\Pr[h_R(a) = h_R(b)] \leq \epsilon$$

Note: universal $= (1/m)$-universal, where $m := |\mathcal{V}|$

An ϵ-universal family: polynomial evaluation hash

Let m be a prime, and t a positive integer

Define $\mathcal{U} := \mathbb{Z}_m^t$, $\Lambda := \mathbb{Z}_m$, $\mathcal{V} := \mathbb{Z}_m$

For $\lambda \in \Lambda$, $a = (a_0, a_1, \ldots, a_{t-1}) \in \mathcal{U}$, define

$$h_\lambda(a) := \sum_{i=0}^{t-1} a_i \lambda^i$$

Define

$$\mathcal{H} := \{h_\lambda\}_{\lambda \in \Lambda}$$

Theorem: $\mathcal{H}$ is ϵ-universal for $\epsilon = (t-1)/m$

Proof. Suppose $(a_0, \ldots, a_{t-1}) \neq (b_0, \ldots, b_{t-1})$

Count the number N of solutions λ to the equation

$$\sum_i a_i \lambda^i = \sum_i b_i \lambda^i$$

Re-write this as

$$\sum_i c_i \lambda^i = 0$$

where $c_i := a_i - b_i$

By assumption, not all c_i's are zero

Want to show: $N \leq \epsilon \cdot |\Lambda| = (t-1)/m \cdot m = t-1$

$N = \#$ roots of $\sum_i c_i \mathbf{x}^i$, which is a non-zero polynomial of degree at most $t-1$

$\therefore N \leq t-1$ QED