Hash Functions

Adapted from the CLRS book slides

HASH FUNCTIONS

What makes a good hash function?

- Ideally, the hash function satisfies the assumption of independent uniform hashing: each key is equally likely to hash to any of the m slots, independent of any other key.
- In practice, it's not possible to satisfy this assumption, since you don't know in advance the probability distribution that keys are drawn from, and the keys may not be drawn independently.
- If you know the distribution of keys, you can take advantage of it.

Example: If keys are random real numbers independently and uniformly distributed in the half-open interval $[0, 1)$, then can use $h(k) = \lfloor km \rfloor$.

KEYS ARE INTEGERS, VECTORS, OR STRINGS

In practice, hash functions assume that the keys are either

- A short nonnegative integer that fits in a machine word (typically 32 or 64 bits), or
- A short vector of nonnegative integers, each of bounded size, e.g., a string of bytes.

For now, assume that keys are short nonnegative integers.

STATIC HASHING

A single, fixed hash function. Randomization comes only from the hoped-for distribution of the keys.

Division method

$h(k) = k \bmod m$.

Example: $m = 20$ and $k = 91 \Rightarrow h(k) = 11$.

Advantage: Fast, since requires just one division operation.

Good choice for m: A prime not too close to an exact power of 2.

MULTIPLICATION METHOD

1. Choose constant A in the range $0 < A < 1$.
2. Multiply key k by A.
3. Extract the fractional part of kA.
4. Multiply the fractional part by m.
5. Take the floor of the result.

Put another way, $h(k) = \lfloor m\,(kA \bmod 1) \rfloor$, where $kA \bmod 1 = kA - \lfloor kA \rfloor =$ fractional part of kA.

Disadvantage: Slower than division method.

Advantage: Value of m is not critical. Can choose it independently of A.

RANDOM HASHING

Suppose that a malicious adversary, who gets to choose the keys to be hashed, has seen your hashing program and knows the hash function in advance. Then they could choose keys that all hash to the same slot, giving worst-case behavior. Any static hash function is vulnerable to this type of attack.

One way to defeat the adversary is to choose a hash function randomly independent of the keys. We describe a special case, ***universal hashing***, which can yield provably good performance average when collisions are resolved by chaining, no matter the keys.

UNIVERSAL HASHING

Consider a finite collection $\mathcal{H}$ of hash functions that map a universe U of keys into the range $\{0, 1, \ldots, m-1\}$. $\mathcal{H}$ is ***universal*** if for each pair of keys $k_1, k_2 \in U$, the number of hash functions $h \in \mathcal{H}$ for which $h(k_1) = h(k_2)$ is $\leq |\mathcal{H}| / m$.

In other words, $\mathcal{H}$ is universal if, with a hash function h chosen randomly from $\mathcal{H}$, the probability of a collision between two different keys is no more than the $1/m$ chance of just choosing two slots randomly and independently.

ACHIEVABLE PROPERTIES OF RANDOM HASHING

Families of hash functions may exhibit any of several properties. Consider a family $\mathcal{H}$ of hash functions over domain U and with range $\{0, 1, \ldots, m-1\}$, keys in U, slot numbers in $\{0, 1, \ldots, m-1\}$, and a hash function h picked randomly from $\mathcal{H}$. The following properties may pertain to $\mathcal{H}$:

Uniform: The probability over picks of h that $h(k) = q$ is $1/m$.

Universal: For any distinct keys k_1, k_2, the probability that $h(k_1) = h(k_2)$ is at most $1/m$.

ϵ-universal: For any distinct keys k_1, k_2 the probability that $h(k_1) = h(k_2)$ is at most ϵ (so that universal means $1/m$-universal).

***d*-independent:** For any distinct keys $k_1, k_2, \ldots, k_d$ and any slots $q_1, q_2, \ldots, q_d$, not necessarily distinct, the probability that $h(k_i) = q_i$ is $1/m^d$.

BASED ON NUMBER THEORY

- Choose a prime number p large enough so that every possible key is in the set $\{0, 1, \ldots, p-1\}$. Assume that $p > m$ (otherwise, just use direct addressing). m need not be prime.
- Denote $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$.
- Given any $a \in \mathbb{Z}_p^*$ and any $b \in \mathbb{Z}_p$, define

$$h_{ab}(k) = ((ak + b) \bmod p) \bmod m \ .$$

Example: $p = 17, m = 6, a = 3, b = 4 \Rightarrow$

$$\begin{aligned} h_{3,4}(8) &= ((3 \cdot 8 + 4) \bmod 17) \bmod 6 \\ &= (28 \bmod 17) \bmod 6 \\ &= 11 \bmod 6 \\ &= 5 \ . \end{aligned}$$

BASED ON NUMBER THEORY (continued)

- Given p and m, the family of hash functions is

  $\mathcal{H}_{pm} = \{h_{ab} : a \in \mathbb{Z}_p^* \text{ and } b \in \mathbb{Z}_p\}$.

  Each maps $\mathbb{Z}_p$ to $\mathbb{Z}_m$.
- This family of hash functions contains $p(p-1)$ functions, one for each combination of a and b.

Theorem

The family $\mathcal{H}_{pm}$ of hash functions defined above is universal.