COMPSCI 330 Lecture 18 Hashing

Wednesday, November 2, 2016 2:57 PM

- Data structure: Map/Dictionary
- Hashing
- Hash Functions

- data structure for map / dictionary
  - maintain a set of (key, value) pairs
  - given a key, want to find its value quickly
  - operations
    ① insert(key, value) add (key, value) pair to the dictionary
    ② find(key) return value for this key / report key does not exist.
- goal : 1. fast insert and find (O(1) time)
  2. save space
- for simplicity : key is a number in $\{0, 1, 2, \cdots, m-1\}$ $(m = 2^{32})$

  if we only want to achieve 1

  can use a large array $a[0 \ldots m-1]$

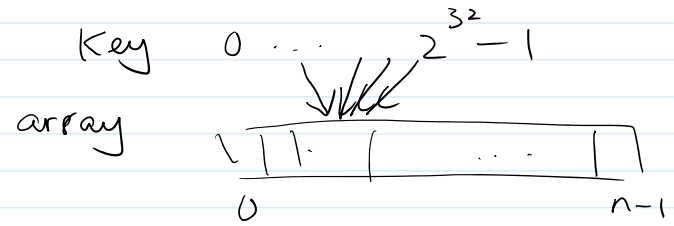
  insert : $a[key] = value$

  find : return $a[key]$

  if we only want to achieve 2

  can store (key, value) pairs in a list.

- Hashing

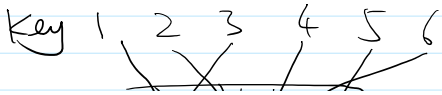


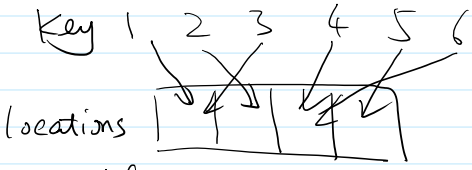
idea: map multiple keys to the same location.

- hash function: $\{0, \cdots, m-1\} \longrightarrow \{0, \cdots, n-1\}$

  insert : $a[h(key)] = value$

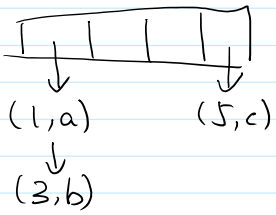
  find : return $a[h(key)]$





- collisions: if $x \neq y$, but $h(x) = h(y)$, we say x, y is a collision.
- working with collisions
  - instead of holding only 1 value at each location use a linked list (array list) to store all key value pairs

(1, a) (3, b) (5, c)

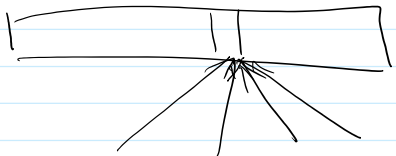


  - insert(key, value) : insert (key, value) into the list at h(key)
  - find(key) : find the key in the list at h(key)

- Question: how to choose a hash function?
  - goal: minimize # of collisions.

observation: cannot use a deterministic hash function (✗)

for any hash function, by pidgeon hole principle



many values map to same location.

very slow if all these key values appear.

solution: use a randomized hash function.

$h_1, h_2, \cdots, h_t$ (t can be very large)

randomly choose $i \in \{1, 2, \cdots, t\}$, use h_i as hash function.

want: for any $x \neq y$, $\Pr[h(x) = h(y)] = \frac{1}{n}$ (1)

Probability for any two key values to have a collision is $\frac{1}{n}$.

Probability is over the choice of hash function

- Lemma: if $h(x)$ maps all key values to independent random locations then h satisfies (1).

Proof: if $x \neq y$, $h(x)$ random in $0 \ldots n-1$

$h(y)$ independent, random in $0, \ldots, n-1$

$$\Pr[h(x) = h(y)] = \frac{1}{n}$$ □

- Claim: expected running time of find/insert operation is $O(\frac{k}{n}+1)$ where k is the number of (key, value) pairs in the dictionary.

Proof: For find(key), running time = length of the list at $h(key)$

Suppose the pairs in the dictionary is $(key_1, v_1) \ldots (key_k, v_k)$

$$\mathbb{E}[\text{length of the list}] = \sum_{i=1}^{k} \Pr[h(key) = h(key_i)]$$

$$\leq \frac{k}{n} + 1$$ □

- Problem: if we select a uniformly random hash function, can not represent/store the function efficiently.
  - total # of functions from $\{0 \ldots m-1\}$ to $\{0 \ldots n-1\}$ n^m, cannot sample/store such functions.
  - need: function that can be represented succintly.
  - recall: modular arithmetic

can do $+, -, \times \mod p$

$p = 5$ $\quad 3+4 = 2 \pmod 5$ $\quad 3 \times 4 = 2 \pmod 5$

$3 - 4 = 4 \pmod 5$

if p is a prime

for any $x \bmod p \neq 0$, there is a unique $y \in \{0, \ldots p-1\}$ s.t. $xy \bmod p = 1$ $(y = x^{-1})$

example: $x = 3$ $p = 5 \Rightarrow y = 2$

$3 \times 2 \bmod 5 = 1$

There is an algorithm (extended Euclid) that can compute y given x, p.

- Design a hash function: p is prime, $n = p$

Key: represent the key in p-ary representation

$$\text{key} = \sum_{i=1}^{t} x_i p^{k-1} \qquad x_i \in \{0, 1, \cdots, p-1\}$$

hash function: randomly choose $a_1, a_2, \cdots, a_t \in \{0, 1, \cdots, p-1\}$ (randomly)

$$h_{a_1, a_2, \cdots, a_t}(\text{key}) = \left(\sum_{i=1}^{t} a_i x_i \right) \bmod p$$

$p = 11 \quad t = 2 \quad 0 \le \text{key} < 11^2 = 121$

$a_1 = 7 \quad a_2 = 4$

- $h_{a_1, a_2}(37)$

$37 = 4 + 3 \times 11 \Rightarrow x_1 = 4 \quad x_2 = 3$

$h_{a_1, a_2}(37) = (a_1 x_1 + a_2 x_2) \bmod 11 = (7 \times 4 + 4 \times 3) \bmod 11$

$= 7$

- Claim: For this hash function, for any $x \neq y$, $\Pr[h(x) = h(y)] = \frac{1}{p}$

Proof: $x \neq y \Rightarrow$ there is one location where $x_q \neq y_q$

first fix $a_1, a_2, \cdots, a_{q-1}, a_{q+1}, \cdots, a_t$

$$h(x) = \sum_{i=1}^{t} a_i x_i$$

$$h(y) = \sum_{i=1}^{t} a_i y_i$$

$$h(x) = h(y) \Rightarrow a_q x_q + \sum_{i \neq q} a_i x_i = a_q y_q + \sum_{i \neq q} a_i y_i \bmod p$$

$$a_q (x_q - y_q) = \sum_{i \neq q} (a_i y_i - a_i x_i) \bmod p$$

because $x_q \neq y_q$

$$a_q = \underbrace{(x_q - y_q)^{-1} \sum_{i \neq q} (a_i y_i - a_i x_i)}_{b} \bmod p$$

$$\Pr[h(x) = h(y)] = \Pr[a_q = b \bmod p]$$

$$= \frac{1}{p}$$

□