7 Mar 2025

Hashing - II

Hash function $h(k)$ for key k:

- Assumption of simple uniform hashing is difficult to guarantee
- Desired properties:
  1. Should distribute keys uniformly into the slots of the table
  2. Regularity in the key distribution should not affect this uniformity.
     → ⇒ No patterns in hashing.

- Division Method:

$$h(k) = k \bmod m \quad ; m \text{ slots}$$

Don't pick m with a small divisor d

e.g. if $d=2$ (i.e, m is even)

then if all keys are even then $h(k)$ will be even

i.e, (even) mod (even) = (even)

⇒ all odd slots will be wasted

e.g. if $m = 2^r$ then the hash doesn't depend on all bits of the key k

So, if $k = 10111000111\underline{011010}_2$; $r = 6$, $m = 2^6$

$h(k) = 011010$

- A good heuristic is to pick m to be a prime not too close to any power of 2 or 10.

Multiplication method

Assumptions:
- All keys are integers
- $m = 2^r$
- Computer has w-bit words

Define:

$$h(k) = (A \cdot k \bmod 2^w) \text{ rsh } (w - r)$$

(w-bit, w-bit, right shift)

A : odd integer in range $2^{w-1} < A < 2^w$

e.g. $m = 8 = 2^3$, $w = 7$ (7-bit word)

$A = 1011001_2$

$k = 1101011_2$

$A \cdot k = 100101\underline{001}\underline{1001}1_2$

rsh (w-r) ≡ Get r higher order bits of the w-bit part.

mod 2^w

$h(k) = 011_2$

Open addressing to resolve collisions

→ No storage for links required

→ Insertion systematically probes the table until an empty slot is found

a) Probing should explore all slots

b) A probed slot should not be probed again.

→ Hash function takes two arguments:
key, and a probe number

Sequence of hash functions:

$$h: U \times \{0, 1, \ldots, m-1\} \longrightarrow \{0, 1, \ldots, m-1\}$$

U: Universe of keys; $\{0, 1, \ldots, m-1\}$: probe number; $\{0, 1, \ldots, m-1\}$: slot

- for a key k, we get a probe sequence:

$$\langle h(k,0), h(k,1), \ldots, h(k, m-1) \rangle$$

- The table may fill up so $n \leq m$.

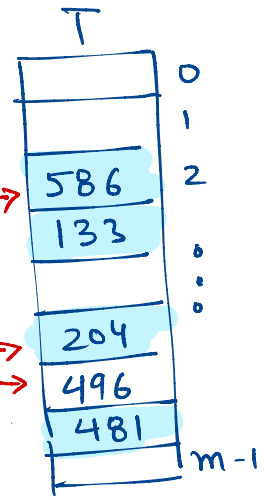
Example of insertion with open addressing.

Insert key $k = 496$

Probe 0: $h(496, 0)$

Probe 1: $h(496, 1)$

Probe 2: $h(496, 2)$



- While searching, go through the same set of Probe sequence
  - ↳ Terminate when the key is found (successful)
    or, when an empty slot is encountered (unsuccessful)

Hash-Insert(T, k)

```
i = 0
repeat
    q = h(k, i)
    if T[q] == NIL
        T[q] = k
        return q
    else i = i + 1
until i == m
Error "Hash table overflow"
```

Hash-Search(T, k)

```
i = 0
repeat
    q = h(k, i)
    if T[q] == k
        return q
    else i = i + 1
until T[q] == NIL or i == m
return NIL
```

Probing Strategies

1. Linear probing

$$h(k, i) = (h'(k) + i) \bmod m$$

↳ ordinary hash function

$$\Rightarrow h(k, 0) = h'(k) \bmod m$$

↳ Suffers from primary clustering

↳ long runs of occupied slots build up thereby increasing search time

↳ long runs of occupied slots tend to get longer

2. Quadratic probing.

$$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$$

when c_1, c_2 are constants

↳ Suffers from a milder form of clustering called Secondary clustering

↳ occurs when $h(k_1,0) = h(k_2,0)$

3. Double hashing

Given two ordinary hash functions $h_1(k)$, $h_2(k)$ we define double hashing as:

$$h(k,i) = (h_1(k) + i\, h_2(k)) \bmod m$$

↳ $h_2(k)$ must be relatively prime to m

Example cases: 1. Let $m = 2^r$ and $h_2(k)$ produces odd number

2. Let m be prime & $h_2(k)$ returns a positive integer less than m.

Complexity of open addressing: (Note $\alpha \leq 1$ for open addressing)

a) # of probes in an unsuccessful search is at most $\frac{1}{(1-\alpha)}$

if $\alpha = 0.5$ then # = 2

if $\alpha = 0.9$ then # = 10 | if $\alpha = 1$, then # = m

b) # of probes in a successful search is at most $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$