

Hash Tables

XXX-XXX-XXX store SIN's & do following operations:

- Insert
- Delete
- Search

Current knowledge, using:

an array

a linked-list

Using an array:

0		Insert, delete, search: $O(1)$ time
1		Drawback:
2	000000001	i.e: Canada has 35,000,000
3	000000003	But 1 billion entries array
⋮	⋮	⇒ More than 97% array space goes wasted! space
⋮	⋮	
999 999 999		

Using a linked list:

→ □ → □ ... → □ → □ → NULL

Memory: $O(n)$, as many elements as we store

Time for operations: $O(n)$

⇒ Trade off of time and memory.

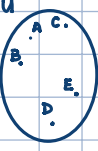
⇒ Hash tables provide "nearly" $O(1)$ time for the operations and $\approx O(n)$ space

U = universe of n -keys, m = size of hash table

we want $m \geq n$ (or they cannot fit)

$\alpha = \frac{n}{m} \leq 1$ is load factor

Hashing function $h(\text{key})$: $h: U \rightarrow \mu$ (μ is an integer $[0 \dots m-1]$) ⇒ a good hash function: to avoid collision as much as possible

U 	0	B	→ B	$h(A)=1$ → functions of calculation
	1	A	→ A	$h(B)=0$
	2			$h(C)=3$
	3	C	→ C ⇒ E ⇒ D	$h(E)=3$ ⇒ Collision
	⋮	⋮		To solve collision: resolution
	$m-1$			- Resolution by Chaining: create a linked list at that location - Resolution by Open addressing

h : division method

$$h(\text{key}) = k \bmod m$$

Bad value: m = powers of 2

Good value: m = prime #s (质数)

H	$m=5$
0	$7 \bmod 5 = 2$
1	→ 11 → 21 $11 \bmod 5 = 1$
2	→ 7 → 12 $12 \bmod 5 = 2$
3	→ 33 $21 \bmod 5 = 1$
4	$33 \bmod 5 = 3$

h : Multiplication method

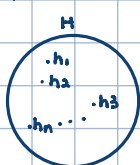
$$h(\text{key}) = m \cdot [(key \cdot A) \bmod 1], \text{ where } 0 < A < 1$$

Good value: $m = \frac{16-1}{2}$ m = powers of 2

Compute $h(\text{key})$ by using bitwise AND, OR etc. CPU operations (Reference at CLRS)

h : Universal hash (Not include for this class)

Introduce randomization (like Quicksort) to perturb (扰动) (introducing noise) in the input



pick a random function for every set of input numbers

To build H family of hash function? **Number theory**

Resolution by Open Addressing

Upon a collision on probe i_0 we examine other cells i_1, i_2, i_3 etc.

We do not build linked lists like in chaining but we only use the hash table entries.

Linear probing:

$$i_0 = h(k)$$

$$i_{j+1} = (i_j + c) \bmod m$$

Example: $m=9$ $c=1$

0		Insert 28
1	19	$\leftarrow 28+28$ Insert 17
2	20	$\leftarrow 28+28$ Delete 21
3	21	$\leftarrow 28+28$ Search 28
4	28	$\leftarrow 28+28$ \downarrow return don't have it
5		\Rightarrow when delete in open addressing, place a tombstone
6		
7		
8	17	

Problem: clustering

Double hashing:

Don't use a predetermined step "c" but a secondary hash function as a "step"

$$i_0 = h_1(k)$$

$$i_{j+1} = (i_j + h_2(k)) \bmod m$$

where h_1 is primary hashing function

h_2 is secondary hashing function

Run time

For successful search ($\alpha = \frac{n}{m} \leq 1$)

Expected # collisions

Chaining: $1 + \frac{\alpha}{2}$

Linear Probe: $\frac{1}{2} (1 + \frac{1}{1-\alpha})$

Double hashing: $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$

