7) Lecture 13-14 Dictionary Problem (Open Addressing)

21 August 2024 07:45

- In open addressing, all elements occupy the hash table itself. — Hash table := array
- That is, each table entry contains either an element of the dynamic set or NIL.
- When searching for an element, we systematically examine table slots until either we find the desired element or we have ascertained that the element is not in the table.
- No lists, ~~Hash Tables~~ no elements are stored outside the table, unlike in chaining.

Linear Probing

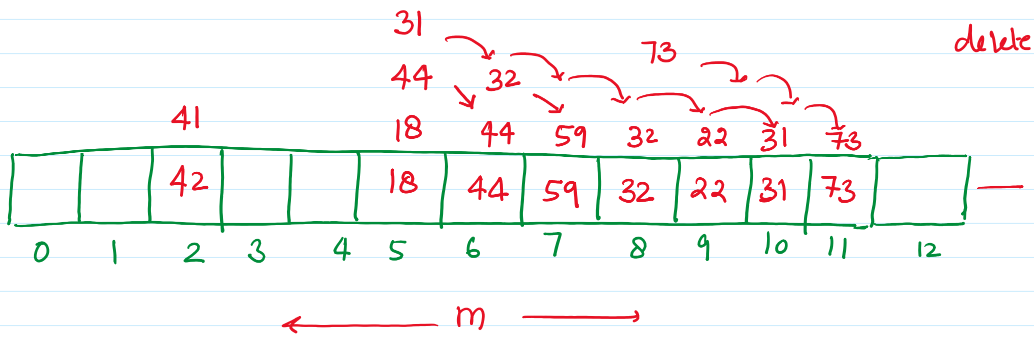
Data := (key, value) , $h(k) = k \bmod 13$

Insert keys 18, 41, 22, 44, 59, 32, 31, 73

Search (key)

Insert (key, value)

delete (key)



Insert: Linear Probing Insert (key)

if (table is Full)

"error"

probe = h (key)

while (table [probe] is occupied)

probe = h(Key)

while (table[probe] is occupied)

[probe = (probe+1) mod m

table[probe] ← Key

Search In linear Probing

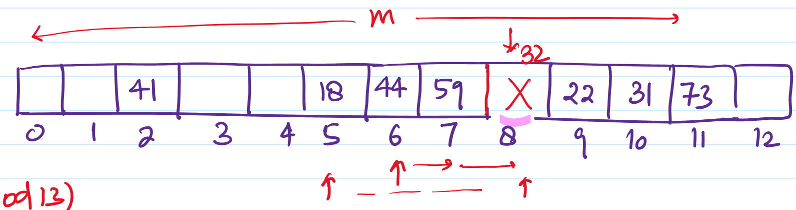
- To search for a key K, we go to (K mod 13), and continue looking at sucessive location until we find K, or encounter with empty location

- Sucessful Search: To search for 31, compute (31 mod 13) = 5 and continue location 6, 7, -- till we find 31 at location 10.

- Unsucessful Search To search 33, compute 33 mod 13 = 7, continue till we encounter the empty location

Deletion in Linear Probing

: Suppose we need to delete element 32.



(32 mod 13)

(31 mod 13) = 5

\# of elements n

\# of slots m

Assume $n <= m$

- Instead of setting location 8 to NULL, we place a marker X.
- During Search when we see X, ignore it and continue to next location

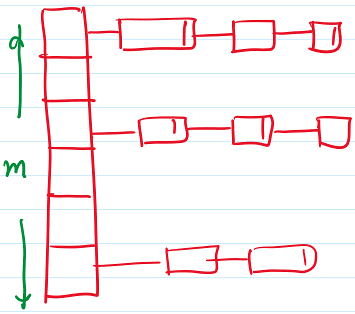
- During insert if X came across, we put that element on the location & remove X.

- Too many X degrades search performace

- Rehash if there are too many 'X'

Hash table based techniques for dictionary Problem

Collision By chaining

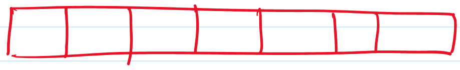


expected search time $O(1+\frac{n}{m}) = O(1+\alpha)$

$n > m$

$\alpha > 1$

Linear Probing

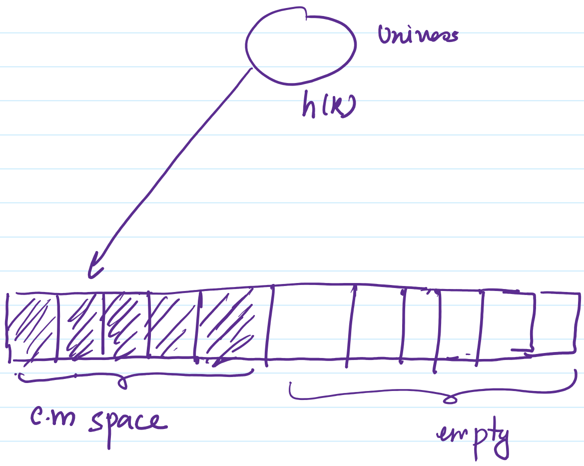


$n < m$

$\alpha < 1$

linear probing use less space than chaining as it doesn't require storing ptr

linear probing is slower than chaining, as we might have to walk along the table for long time.



Double Hashing

- Use two hash function h_1 & h_2
- $h_1(K)$ gives the position of key K in hash table
- $h_2(K)$ gives step count for key K.
- In linear probing $h_2(K) = 1$

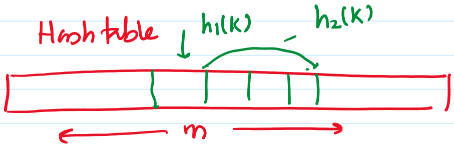
n # of key

m # slots in Hash table

$n < m$

$h_1(K) \in \{0 .. m-1\}$

$h_2(K) \in \{0 .. m-1\}$



Double Hashing Insert (K)

if (table is full) error

probe ← $h_1(K)$, offset ← $h_2(K)$

while table[probe] is occupied

probe ← (probe + offset) mod m

table[probe] ← K

$h_2(K)$ must be relatively prime to m

⇒ every location of hash table is accessible

load factor :- $\alpha = n$

table[probe] ← K

load factor $:= \alpha = \frac{n}{m} < 1$

n: # of keys

m: # of slots in hash table

$n < m$

[Search, Delete is same as linear probing + using $h_2(K)$ fn as offset]

h_1, h_2 are universal Hash fn

- ✓Assume that final effect of $h_1 - h_2$ gives a random location in hash table.

☆ Once all keys are hashed $(m-n)$ cell are empty ⇒ $\left(\frac{m-n}{m}\right)$ fraction of table is empty

⇒ $(1-\alpha)$ fraction of table is empty

☆ Expected # of probes required to find an empty location is $\frac{1}{(1-\alpha)}$

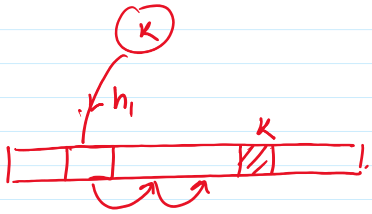
⇓

Expected # of probes required to insert a new key in Hash table $O\left(\frac{1}{1-\alpha}\right)$

[Unsuccessful Search := event when given key is not present in hash table]

Successful Search := " " " is present in hash table.

☆ Expected no. of probe for successful search = Expected no. of probe required to insert element



m = 100

1 – 49   50th key ≤ 2

51 – 74   75th key

To insert a key we need to find empty location

Inserting	Expected no. of probes	Total no. of probes
First m/2	≤ 2	≤ m
Next m/4	≤ 4	≤ m
Next m/8	≤ 8	≤ m

Expected

# of probes requires to insert

$$\underbrace{\frac{m}{2} + \frac{m}{4} + \frac{m}{8} - - + \frac{m}{2^i}}_{\text{After inserting these keys}}$$ element = Expected # of probes is $\leq mi$

$= -m \log(1-\alpha)$

$\frac{m}{2^i}$ cells are empty

$$\frac{1}{2^i} = 1-\alpha$$

$$\Rightarrow i = -\log(1-\alpha)$$

Ans No. of probes required to leave $(1-\alpha)$ fraction of table empty $= -m\log(1-\alpha)$

" " insert n element is $= -\frac{m}{n}\log(1-\alpha)$

" Expected # of probes for succesful search $= \left(\frac{1}{\alpha}\right)\log\left(\frac{1}{1-\alpha}\right)$

$$O\left[\frac{1}{\alpha}\log\left(\frac{1}{1-\alpha}\right)\right]$$

	Successful Search	Unsuccessful Search	Deletion	Insertion
Expected no. of probes	$O\left(\frac{1}{\alpha}\right)\log\left(\frac{1}{1-\alpha}\right)$	$O\left(\frac{1}{1-\alpha}\right)$	$O\left(\frac{1}{\alpha}\log\frac{1}{1-\alpha}\right)$	$O\left(\frac{1}{1-\alpha}\right)$

$- l_i$ # of keys hashed in i^{th} slot

$$\sum_{i=0}^{m-1} l_i^2 > cn$$

$O(n)$

n # of keys $5n$