HASHING

T.C. with Hashing : O(1) : avg. case
O(n) : worst case

Lec3L - slide 8

$h(K) = a$ h → hash f^n
K → key
a → hash value of the key

While choosing hash f^n, consider :
- easy to compute
- generate address with min.^m collision

Techniques for making hash f^n :-

1) Truncation Method :-

↳ Simplest
- We take a part of the key as address.
- Chances of collision are more.

Eg. The keys are 6239457 2, 8713555 65, 9345727 1, 4539322 5

So, the address will be 72, 65, 71, 25 resp.

2) Mid-Square Method :-

The key is squared and some digits from the middle of this square are taken as address.

3) Folding Method :-

The key is divided into different parts where the length of each part is same as that of the req. address, except possibly the last part

Eg :- Let key is 123945234 and table size is 1000.

123945234 → 123 945 234

add = 1302

Ignore the final carry 1, the address is 302

4) Division Method

The key is divided by the table size and the remainder is taken as the address of the hash table

Eg. The keys are 123, 945, 234 and table size is 11,
the address will be :-

123 % 11 = 2
946 % 11 = 10
235 % 11 = 4

COLLISION HANDLING

- Chaining / Seperate Chaining / Open Hashing
- Open Addressing / Closed Hashing

1) SEPERATE CHAINING

Make each cell of hash table point to a linked list of records that have same hash f^n value

eg. hash f^n → key mod 7
keys → 50, 700, 76, 85, 92, 73, 101

50 % 7 = 1	0	700
700 % 7 = 0	1	50 → 85 → 92
76 % 7 = 6	2	
85 % 7 = 1	3	73 → 101
92 % 7 = 1	4	
73 % 7 = 3	5	
101 % 7 = 3	6	76

Initial Empty Table

Lec 31 - slide 21 (Advantages & Disadvantages)

Performance

m = No. of slots in hash table
n = No. of keys to be inserted in hash table
Load factor $\alpha = n/m$
Expected time to search = $O(1+\alpha)$
" " " insert/delete = $O(1+\alpha)$

T.C. of all operations is $O(1)$ if α is $O(1)$.

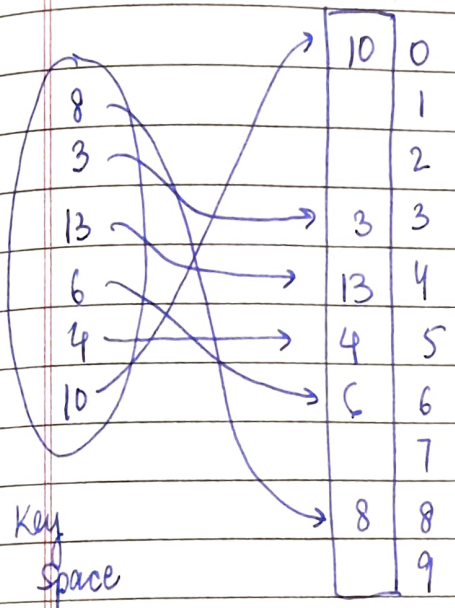
2) OPEN ADDRESSING

1→ Linear Probing

$h(x) = x \% size$ $\quad size = 10$

$h'(x) = [h(x) + f(i)] \% size$ $\quad f(i) = i$

where $i = 0, 1, 2 - - -$



$h'(8) = [h(8) + f(0)] \% 10$
$= 8$

$h'(3) = [h(3) + f(0)] \% 10$
$= 3$

$h'(13) = [h(13) + f(0)] \% 10$
$= 3$ (collision)

$h'(13) = [h(13) + f(1)] \% 10$
$= 4$

$h'(6) = 6$

$h'(4) = [h(4) + f(0)] \% 10 = 4$ (collision)

$= [h(4) + f(1)] \% 10 = 5$

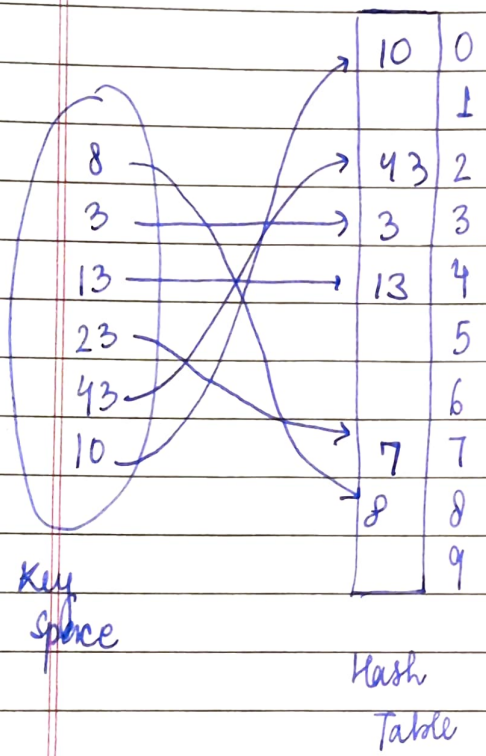
For searching any key element, use hash fn, go to that particular index and start searching for element from that index until the element is found or empty space is reached

2 → Quadratic Probing

$h(x) = x \% \text{ size}$

$h'(x) = [h(x) + f(i)] \% \text{ size}$ $\quad f(i) = i^2$

where $i = 0, 1, 2 \ldots$



$h'(43) = (3 + 3^2) \% 10$

3 → Double hashing

$h'(x) = [h_1(x) + i\, h_2(x)] \% \text{ size}$ where $i = 0, 1, 2 \ldots$

h_1 & h_2 are 2 hash functions.