2024|08|20 – Data Structures – Week 04

Perfectly random mapping → space complexity = $O(|U| \log m)$

Dot product hash family

- assume m is prime
- assume $|U| = m^r$ for $r \in \mathbb{Z}^+$

base m representation

bits

$a = \langle a_0, \ldots, a_{r-1} \rangle$

$k = \langle k_0, \ldots, k_{r-1} \rangle$

- $\mathcal{H}_a = \{ h_a \mid a \in \{0, \ldots, |U| - 1\} \}$

o add example

$$h_a(k) = \left(\sum_{i=0}^{r-1} a_i k_i \right) \bmod m$$

* Space required to store $h_a(\ )$ = space required to store a

= $O(\log|U|)$

a will be stored in binary

Theorem: Dot product hash family is universal.

Proof: Let $k \neq k'$. We need to prove that

$$Pr_a\left[h_a(k) = h_a(k')\right] \leq \frac{1}{m}$$

$|U| = m^r$

maximum r digits

Base-m representation of $k = \langle k_0, k_1, \ldots, k_{r-1} \rangle$

$k' = \langle k_0', k_1, \ldots, k_{r-1}' \rangle$

Since $k \neq k'$, their base-m representations differ in at least one digit, say d^{th} digit.

$$k_d \neq k'_d$$

$$Pr_a\left[h_a(k) = h_a(k')\right] = Pr_a\left[\left(\sum_{i=0}^{r-1} a_i k_i\right) \bmod m = \left(\sum_{i=0}^{r-1} a_i k'_i\right) \bmod m\right]$$

$$= Pr_a\left[\left\{a_d (k_d - k'_d) + \sum_{i \neq d} a_i (k_i - k'_i)\right\} \bmod m = 0 \bmod m\right]$$

$$= Pr_a\left[a_d (k_d - k'_d) \equiv -\sum_{i \neq d} a_i (k_i - k'_i) \pmod m\right]$$

$$= Pr_a\left[a_d \equiv -(k_d - k_d')^{-1} \sum_{i \neq d} a_i (k_i - k_i') \pmod m \right]$$

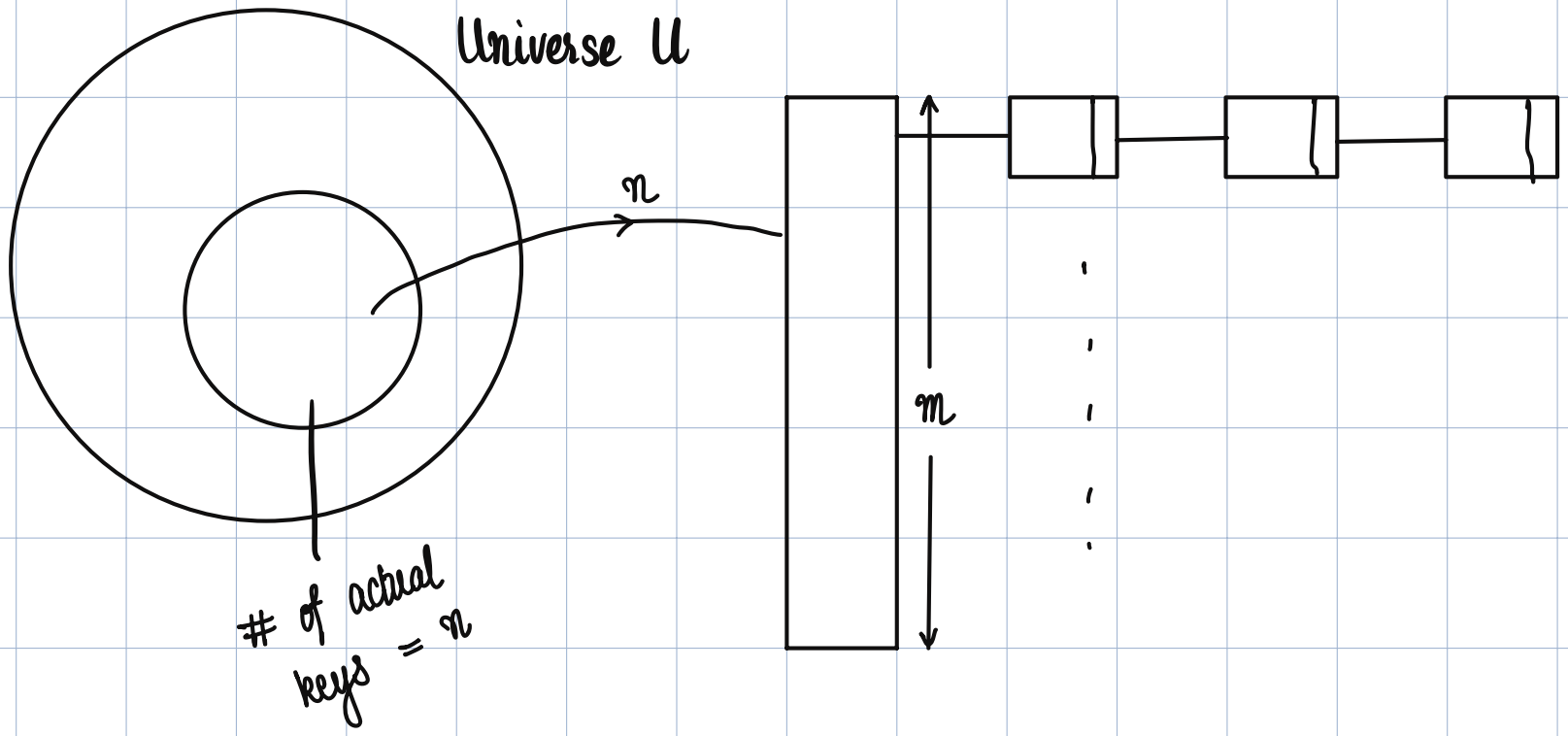
$(k_d - k_d')^{-1}$ → multiplicative inverse of $(k_d - k_d') \bmod m$

$$f(a_0, \ldots, a_{d-1}, a_{d+1}, \ldots, a_{r-1})$$

$(r-1)$ digits

$$= Pr_a\left[a_d = f(a_0, \ldots, a_{d-1}, a_{d+1}, \ldots, a_{r-1}) \right]$$

$$= \frac{\#\text{ favourable choices}}{\text{total number of choices}} \leq \frac{m^{r-1}}{m^r} \leq \frac{1}{m}$$



search (key) —— expected n/m

Insert (key, val) —— $O(1)$

delete (key) —— expected n/m (due to search)

Static Dictionary Problem

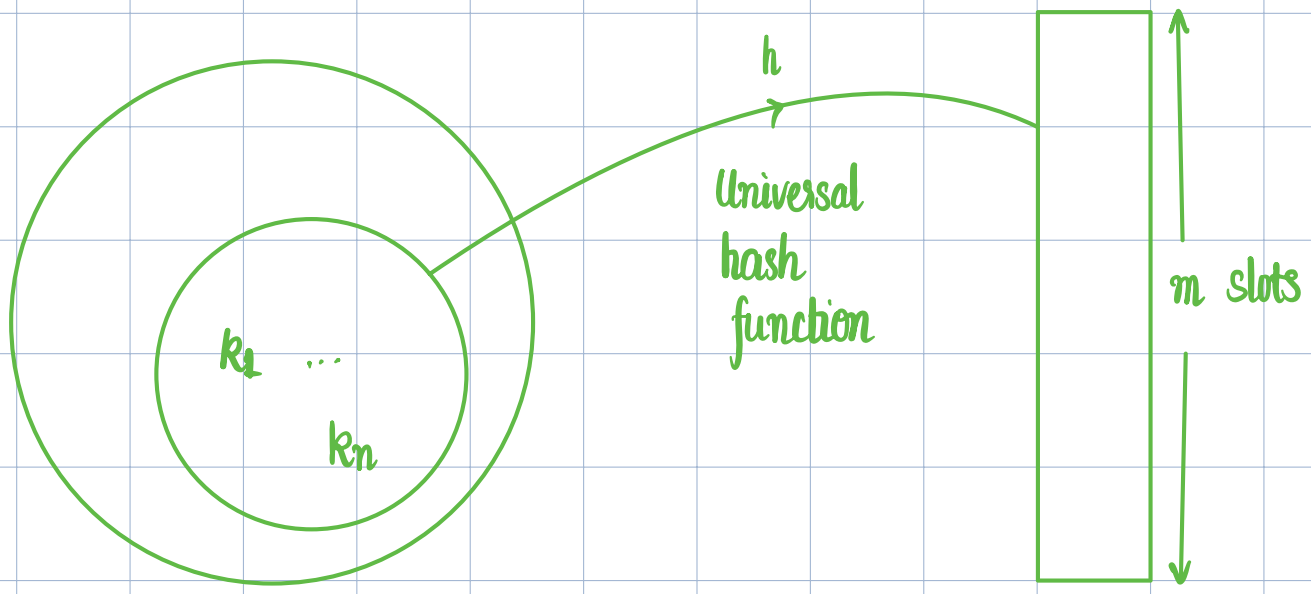
Given n keys upfront — prepare a data structure that facilitates fast search operations.

(No further insertion / deletion needed)

WLOG assume that there is an ordering for the keys.

Naive solutions:	Space Complexity	Time Complexity (search)
① store keys in sorted order	$O(n)$	$O(\log n)$
② Universal hashing	$O(n^2)$	$O(1)$

Simple Solution using Hashing



$$I_{ij} = \begin{cases} 1 & \text{if } h(k_i) = h(k_j) \\ 0 & \text{otherwise} \end{cases}$$

$$\#\ \text{of collisions} = \left(\sum_{i,j} I_{ij}\right) \cdot \frac{1}{2} \qquad i, j \in [n]$$

$$E\left[\sum_{i,j} I_{ij}\right] = \sum_{i,j} E[I_{ij}] \leq \sum_{i,j} \frac{1}{m} < \frac{n^2}{m}$$

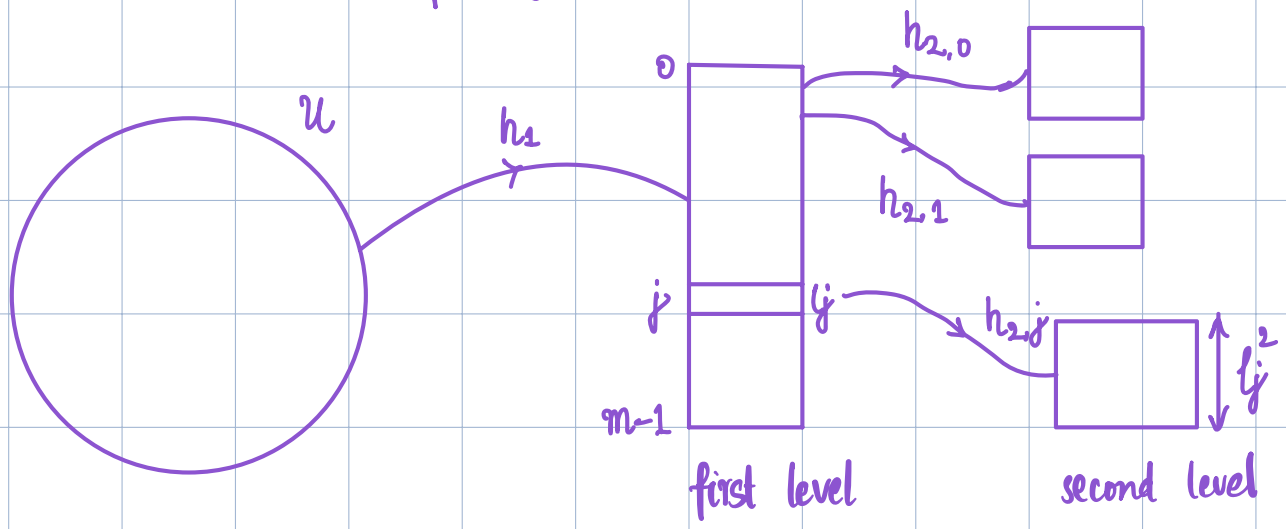
Choose $m = 10\, n^2$

$$E\left[\sum_{i,j} I_{i,j}\right] < \frac{1}{10}$$

$$\Pr\left[\#\text{ of collisions} > 1\right] < \frac{E\left[\sum_{i,j} I_{ij}\right]}{1} < \frac{1}{10}$$

With probability $\frac{9}{10}$, there won't be any collision.

Question: Coming up with data structure for static dictionary problem with space complexity $O(n)$.



* Pick $h_1 : \{0, \ldots, |U| - 1\} \longrightarrow \{0, \ldots, m-1\}$ from universal hash family.

  if $\sum l_j^2 > c \cdot n \longrightarrow$ rehash

* For each slot in first hash table $j \in \{0, \ldots, m-1\}$

  $l_j :=$ no. of keys mapped in j^{th} slot

Sample $h_{2,j} : \{0, \ldots, |U| - 1\} \longrightarrow \{0, \ldots, l_j^2 - 1\}$

Rehash if there is a collision at 2^{nd} level

* Do the following for all $j \in [m]$

while $\left(h_{2,j}(k_i) == h_{2,j}(k_i') \right)$ for $k_i \neq k_i'$ with $h_1(k_i) = h_1(k_i')$

Repick $h_{2,j}$

2024|08|21

Claim: Space complexity is linear in n

$$m = O(n)$$

Proof:

$$\sum_{j=0}^{m-1} \left(l_j^2 \cdot 10\right)$$

$l_j \rightarrow$ random variable

$$E\left[\underbrace{10 \cdot \sum_{j=0}^{m-1} l_j^2}\right] = 10 \cdot E\left[\sum_{j=0}^{m-1} l_j^2\right]$$

space required for second-level hash table

$$= 10 \cdot E\left[\sum_{j=1}^{n}\sum_{i=1}^{n} I_{ij}\right]$$

$l_j :=$ no. of keys mapped in j^{th} slot

$l_j^2 \rightarrow$ pairs of keys mapped in j^{th} slot

$$I_{ij} = \begin{cases} 1, & h(k_i) = h(k_j) \\ 0 \end{cases}$$

$$= 10 \sum_{i=1}^{n} \sum_{j=1}^{n} E[I_{ij}]$$

$$\leq 10 \frac{n^2}{m}$$

$m = \Theta(n)$

$$\Rightarrow E\left[10 \cdot \sum_{j=0}^{m-1} l_j^2\right] \leq c \cdot n \quad \text{— (II)}$$

multiple levels of hashing $\longrightarrow$?

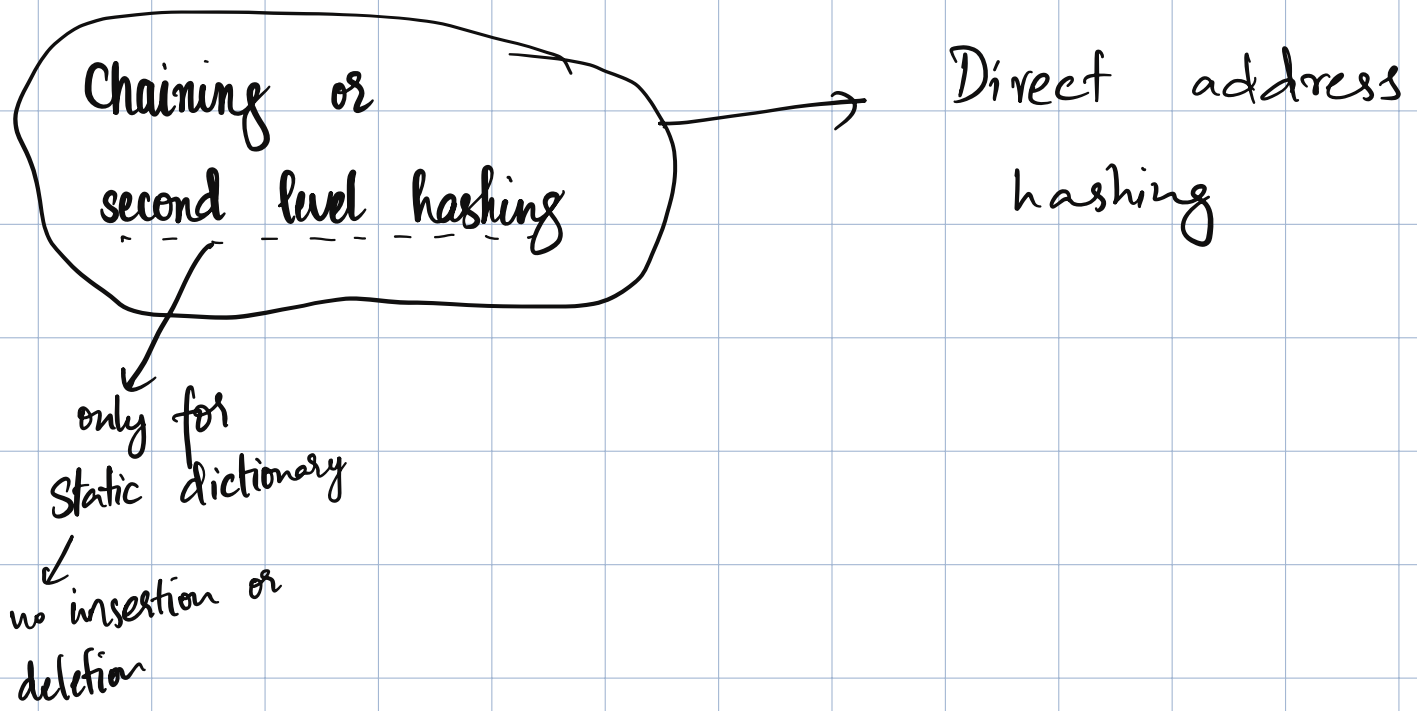
Apply markov on (II)

$$\Pr_{h_1}\left[\underbrace{10\sum_{j=0}^{m-1} l_j^2 > c'n}\right] \leq \frac{E\left[10\sum_{j=0}^{n} l_j^2\right]}{cn} \leq \frac{cn}{c'n}$$

if this happens repeat h_1

choose c such that

$$\frac{c}{c'} < \frac{1}{10}$$


Chaining or
second level hashing
Direct address
hashing
only for
Static dictionary
no insertion or
deletion

Open addressing

Linear Probing

data := (key, value) pair

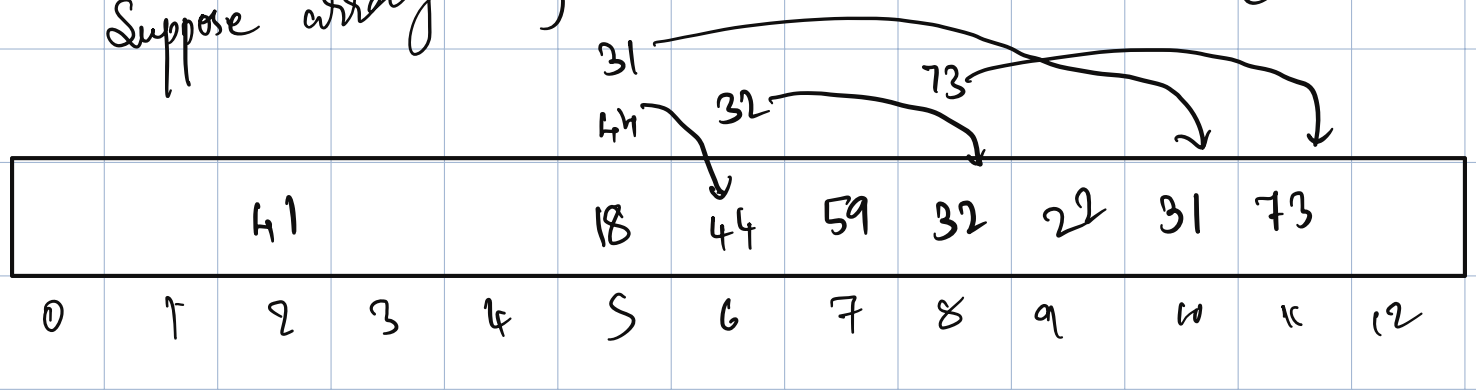
hash table = array

Want:

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

Suppose array size = 13

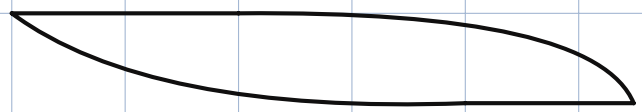
$h(k) = k \bmod 13$



search (key)

insert (key, value)

delete (key)



Insert : Linear Probing Insert (key)

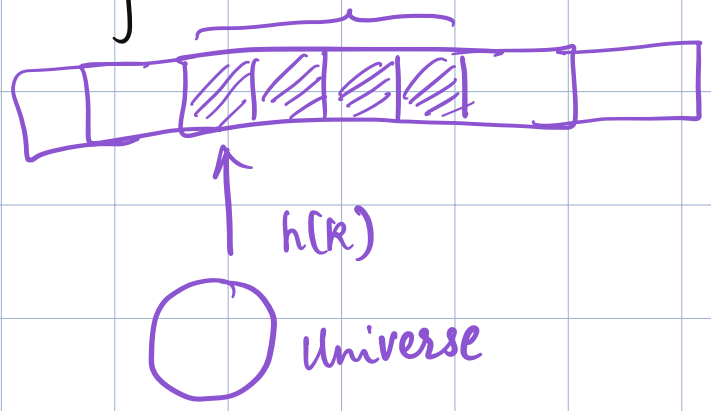
```
if (table is full)
        "error"
Probe = h(key)
while (table[probe] is occupied)
        probe++
table[probe] = key
```

2024|08|22

Linear Probing → space efficient (we don't need to store pointers)

↓ time inefficient for search

Consider the case where all numbers have the same hash fn output → one big cluster



Optimization

① block

↓ Use 2 hash fns

↓ multiple blocks per hash value → Space wastage: empty space but cannot be filled

Double Hashing

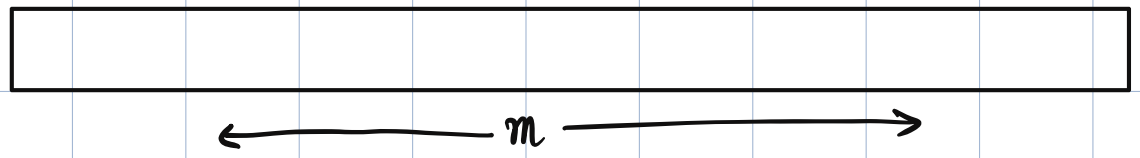
- Use 2 hash functions h_1 & h_2
- $h_1(k)$ gives the position of key k in hash table
- $h_2(k)$ gives step count for key k.
- In linear probing $h_2(k) = 1$

Double Hashing insert (k)

```
if (table is full) error
    probe ← h1(k), offset ← h2(k)
    while table[probe] is occupied
        probe ← (probe + offset) mod m
    table[probe] ← k
```

Search and delete is same as linear probing

hash table



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

$\Rightarrow$ every location of hash table is accessible.

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

n : no. of keys

m : no. of slots in hash table

Collision by chaining → $n > m$, $\alpha > 1$

Search : $O\left(1 + \frac{n}{m}\right)$

$O(1 + \alpha)$

Linear Probing → $n < m$, $\alpha < 1$

Search : $O(1 + \alpha)$

Assumption :-

key $\xrightarrow{\text{final effect of } h_1 + h_2}$ random location in the hash table

h_1, h_2 — universal hash functions

- Once all keys are hashed, $m - n$ cells are empty

$\Rightarrow \left(\frac{m-n}{n}\right)$ fraction of table empty

$\Rightarrow (1 - \alpha)$ fraction of table empty

Ex: $\alpha = 1/2$

$\frac{1}{2}$ empty $\frac{1}{2}$ full

$(h_1 + h_2) \longrightarrow$ randomised effect

$$P(\text{empty slot for new key}) = \frac{1}{2}$$

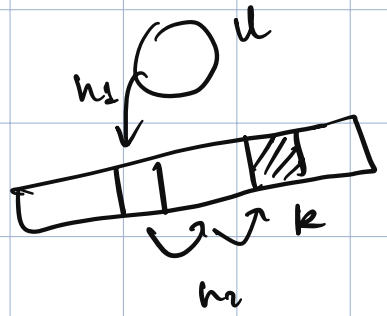
$$\therefore E\left[\begin{array}{c}\#\text{ of probes required to fill an empty location}\end{array}\right] = \underbrace{\frac{1}{1-\alpha}}_{\begin{array}{c}\text{expected no. of trials for unsuccessful search}\end{array}}$$

Unsuccessful search

↳ even when given key is not present in the hash table

Successful search → event when given key is present in the hash table

Expected No. of probes while searching = E. No. of probes while inserting



To insert an element, we need to find empty location

Total Expected no. of probes for inserting first $m/2$ keys $= 2 \cdot \frac{m}{2} = m$

Total Expected no. of probes for inserting next $m/4$ keys $\leq m$

Next $m/8$

Expected no. of probes ≤ 8

Next $m/2^i$

Total no. of probes $\leq m$