6) Lecture 8-12 Dictionary Problem

12 August 2024 15:29

Motivating example:

- Bank account scenario:
  - Each account is identified by an account number
  - Each account stores additional information such as
    - Balance,
    - Name & address of account holder
    - History of transactions
- Any application that wish to operate on account need to provide the account no as key

The dictionary:

- Dictionaries is a data structure that store elements so that they can be located quickly using keys
- It provides and abstract model of database such that
  - Stores (key, element) pair
  - The main operations supported by dictionary are
    - FindElement (key),
    - insertItem(key, Element),
    - deleteItem(Key),

Following data structure can be used to implement dictionary:

- Array & linked list (inefficient)
- Hash tables
- Binary tree
- Red black tree
- AVL tree
- B tree

Searching problem:

Input: Sequence of numbers (database)
A single number (query)

Output: Index of the number, if found or NIL

Standard Techniques:

Data = { (key, value) }

- Insert (key, value)
- Search (key)
- Delete (key)

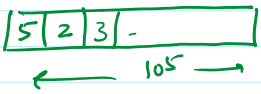
```
Struct node
{
   int  key
   int  value
}
```

Certain ordering can be defined on key.

(i) Create an Unordered Sequence 5, 2, 3, 10 –

Use array - append new element at the end of array

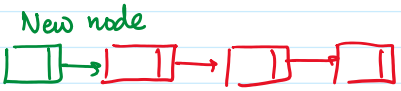
- Insertion O(1) — Advantage
- search Θ(n) (linear search - list is unordered)
- deletion Θ(n) (due to search + shift)
- Space complexity is high



disadvantage

Use linked list – append new element at beginning of linked list

- Insertion O(1) Advantage
- Search / deletion Θ(n) (linear search)
- Lesser Space complexity



Useful to applications where frequent insertion & rare search / deletion.

(2) Create an ordered Sequence

- Create an array of struct - insertion sort
- Search O(log n) (Binary Search)
- Insertion O(n)
- Deletion O(n)

> Mainly due to shift

4

- Deletion O(n) — mainly due to shift

Useful in application where frequent search and rare insertion/deletion

Input {(key, value)} = Data

Search (key)

Insert (key, value)

Delete (key)

Direct address table

Key: Account No. 15 digits

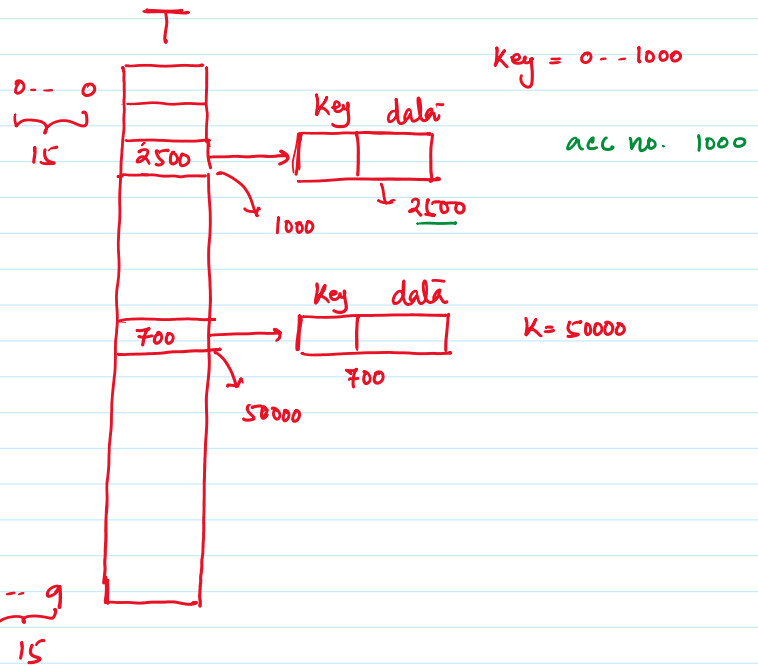
0..0

.

.

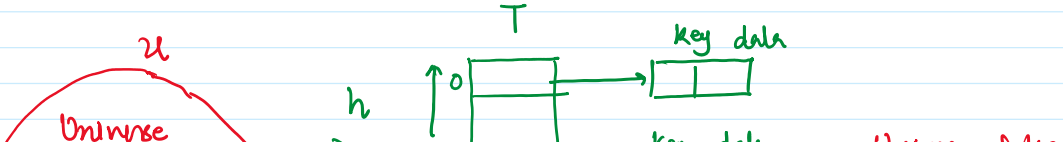
9..9

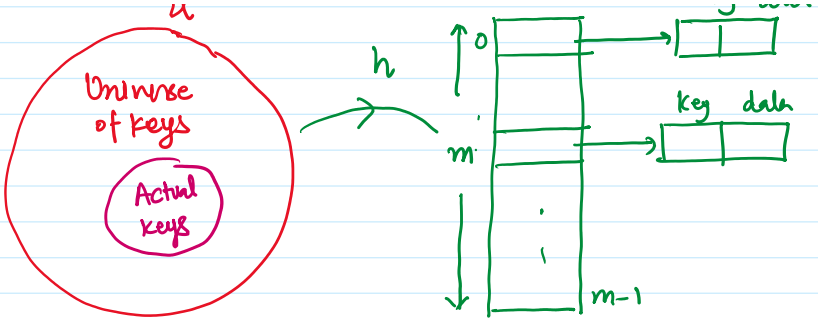
One location for each possible acc. no.



- Search O(1)
- Insertion O(1)
- Deletion O(1)

Disadvantage: Space complexity is high





Universe of keys := set of all possible 15 digit no.

$|U|$:= # of element in universe

n := set of actual key

m := size of array T

$\sim \frac{n}{m}$

We used $m = |U| \longrightarrow$ higher space complexity

we want $m << |U|$

Hash fn. $[h: U \longrightarrow \{0 \cdots m-1\}]$

$m = 100$

Naive approach: $h(x) = x \bmod m$ $x :=$ acc. no

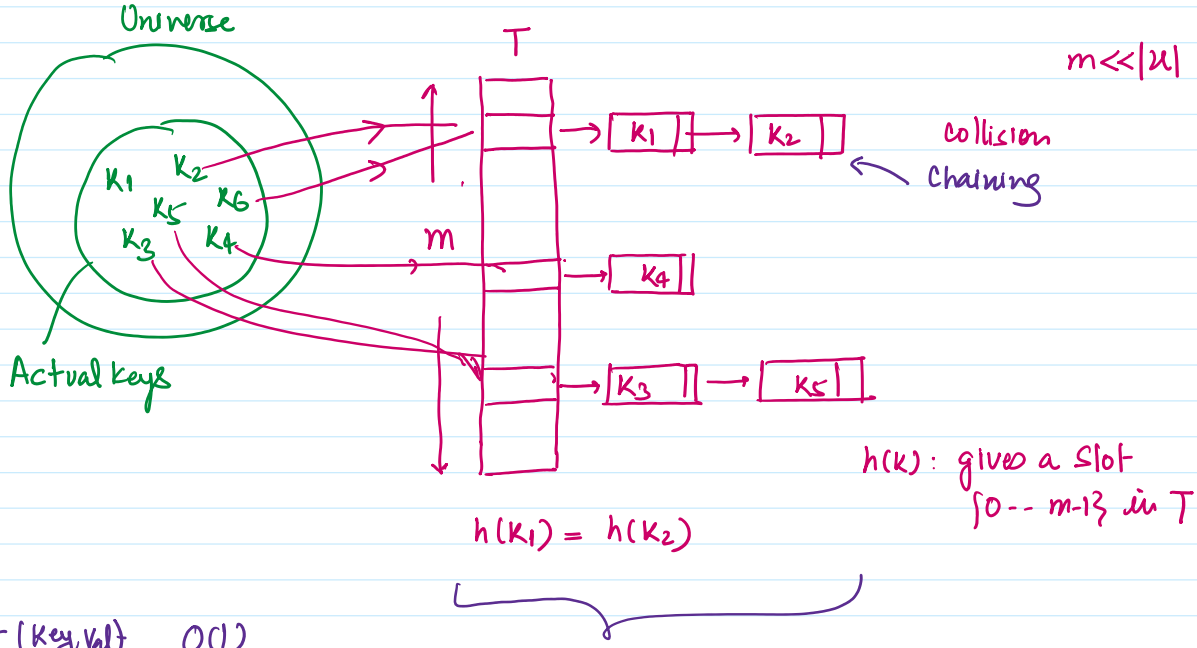
$h: U \longrightarrow \{0 \cdots m-1\}$:= hash function

Key $\in \mathbb{Z}_n^+$

$m = 100$

h: acc no. (15 digit) $\longrightarrow \{0 \cdots 99\}$

acc. no. mod 100



$m << |U|$

h(K): gives a slot $\{0 \cdots m-1\}$ in T

$h(K_1) = h(K_2)$

Hash Table

Insert (Key, Val) O(1)

Search (key) O(n)

Delete (key) O(n)

: n keys $\approx \frac{n}{m}$ in each slot

Delete (key)

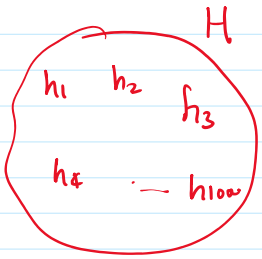
: n keys $\approx \frac{n}{m}$ in each slot

: m slots in T

if $m = \sqrt{n}$

Universal Hashing

Design a family of hash function H

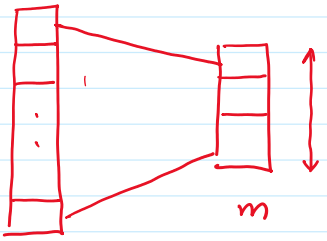


for all pair of keys $k \neq k' \in \mathcal{U}$,

if $\Pr_{h \in H}[h(k) = h(k')] \leq \frac{1}{m}$ — (1)

Then H is called as Universal Hash family.

h : Universal hash fn.



Perfectly random mapping from $|U| \to [m]$

Space required $O(|U| \log m)$

Not space efficient !!

Th: Let n (be) arbitrary distinct key $\in \mathcal{U}$, Let H be a Universal Hash family

$$E\left[\#\text{ keys colliding a partiaular slot } (0 \text{ -- } m-1)\right] \leq \frac{n}{m} + 1.$$

Proof: Let K_0 -- K_{n-1} (n- distinct keys)

Consider some arbitrary key K_i

$E[$ # of key colliding with $K_i]$

Let $I_{ij} = \begin{cases} 1 & \text{if } h(k_i) = h(k_j) \\ 0 & \text{o.w.} \end{cases}$ $\quad h \in_R H$

h is sampled u.a.r. from H

$$E[\#\text{ of key colliding with } k_i] = E\left[\sum_{i \neq j} I_{ij} + I_{ii}\right]$$

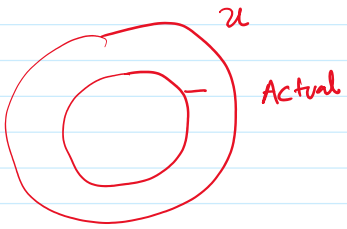
$$= E\left[\sum_{i \neq j} I_{ij}\right] + 1$$

$$= \sum_{i \neq j} E[I_{ij}] + 1$$

$$= \sum_{i \neq j} Pr[h(k_i) = h(k_j)] + 1$$

$$\leq \sum_{i \neq j} \frac{1}{m} + 1 \quad [\text{from Eqn } (1)]$$

$$= \frac{n}{m} + 1$$



Dot product Hash family:

- assume m is prime
- assume $|U| = m^r$ for +ve int r

$\vec{a} = \langle a_0, a_1, \ldots a_{r-1} \rangle$ > base m representation

$\vec{k} = \langle k_0, k_1 \ldots k_{r-1} \rangle$ > base m representation

$$H_a = \{h_a \mid a \in_R \{0 \ldots |U|-1\}\}$$

$$h_a(k) = \left(\sum_{i=0}^{r-1} a_i k_i\right) \bmod m = \langle \vec{a} \cdot \vec{k} \rangle \bmod m \quad \text{— (II)}$$

Space required to store $h_a(\,)$ = space required to store a = $O(\log|U|)$

$m = 2$ $\quad k = 5$ $\quad 101$

$a_1 = 4$ $\quad a_2 = 6$

$h_{a_1}(k) = \langle 100, 101 \rangle \bmod 2$

$= 1$

101
100
$1+0+0 = 1$

$h_{a_2}(k) = \langle 110, 101 \rangle \bmod 2$

Theorem Dot product hash family is universal.

$|\mathcal{U}| = m^r$

Proof: Let $k \neq k' \in \mathcal{U}$

* $\Pr_a [h_a(k) = h_a(k')] \leq \frac{1}{m}$ (Need to show)!!

$k = \langle k_0, k_1 \cdots k_d, \cdots k_{r-1} \rangle$

$k' = \langle k_0', k_1' \cdots k_d' \cdots k_{r-1}' \rangle$ Base m representation,

$a = \langle a_0 \cdots a_d \cdots a_{r-1} \rangle$

Since $k \neq k'$, their base m representation differ at say d^{th} digit

assume $k_d \neq k_d'$

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

By def

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

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

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

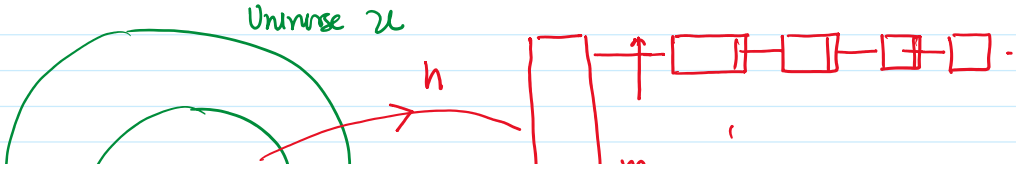
$(k_d - k_d')^{-1}$ → multiplicative inverse of $(k_d - k_d') \bmod m$

!!

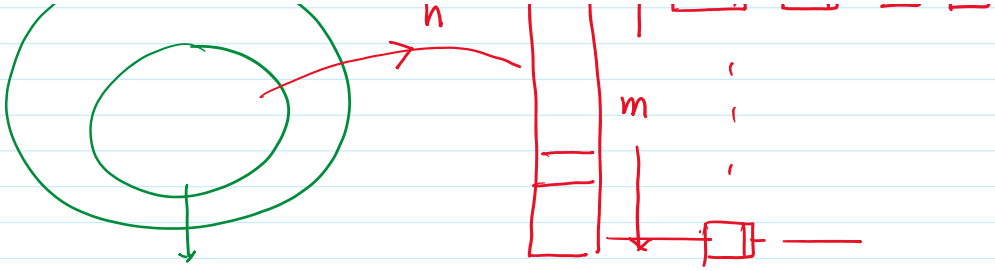
$f(a_0 \cdots a_{d-1}, a_{d+1} \cdots a_{r-1})$

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

$= \frac{\#\text{ favourable choices}}{\text{total no of choices}} \leq \frac{m^{r-1}}{m^r} \leq \frac{1}{m}$



Each slot



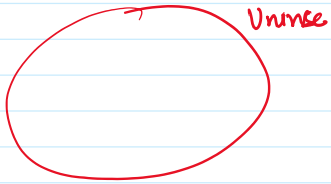
Each slot get $\approx \frac{n}{m}$ key on expectation

Actual keys = n

Search (key) — on expectation n/m

Insert (key, val) — $O(1)$

delete (key) — on expectation n/m (due to search)



Static dictionary Problem

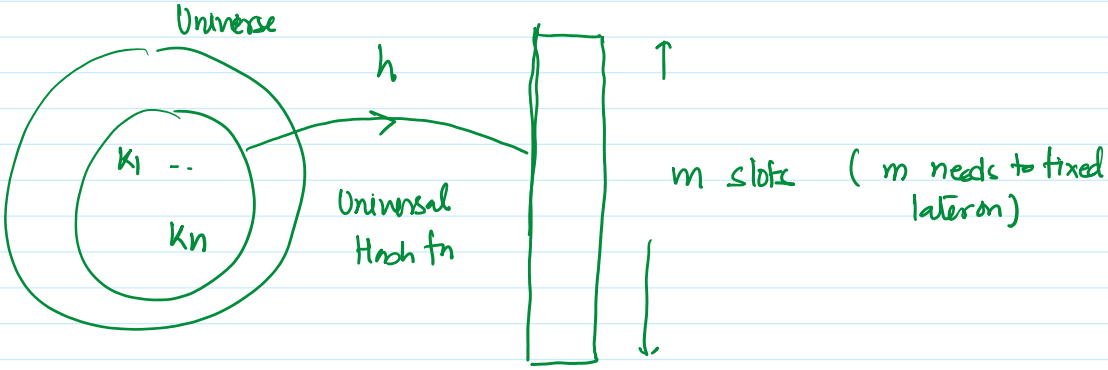
Given n keys upfront — prepare a data structure that facilitates fast search operation

(No further insertion/deletion allowed)

Naive Soln

	Space complexity of preparing data str	Time complexity (search)	
① Store key in sorted order	$O(n)$	$O(\log n)$	— deterministic
② Universal Hashing	$O(n^2)$	$O(1)$	Probabilistic

Simple Soln using Hashing



Lemma 1

$$I_{ij} = \begin{cases} 1 & \text{if } h(k_i) = h(k_j) \quad k_i \neq k_j \\ 0, & \text{o.w} \end{cases}$$

$$\#\text{ of collision} = \frac{1}{2}\underbrace{\sum_{i,j} I_{ij}} \qquad i, j \in [n]$$

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

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

Choose $m = 10n^2$

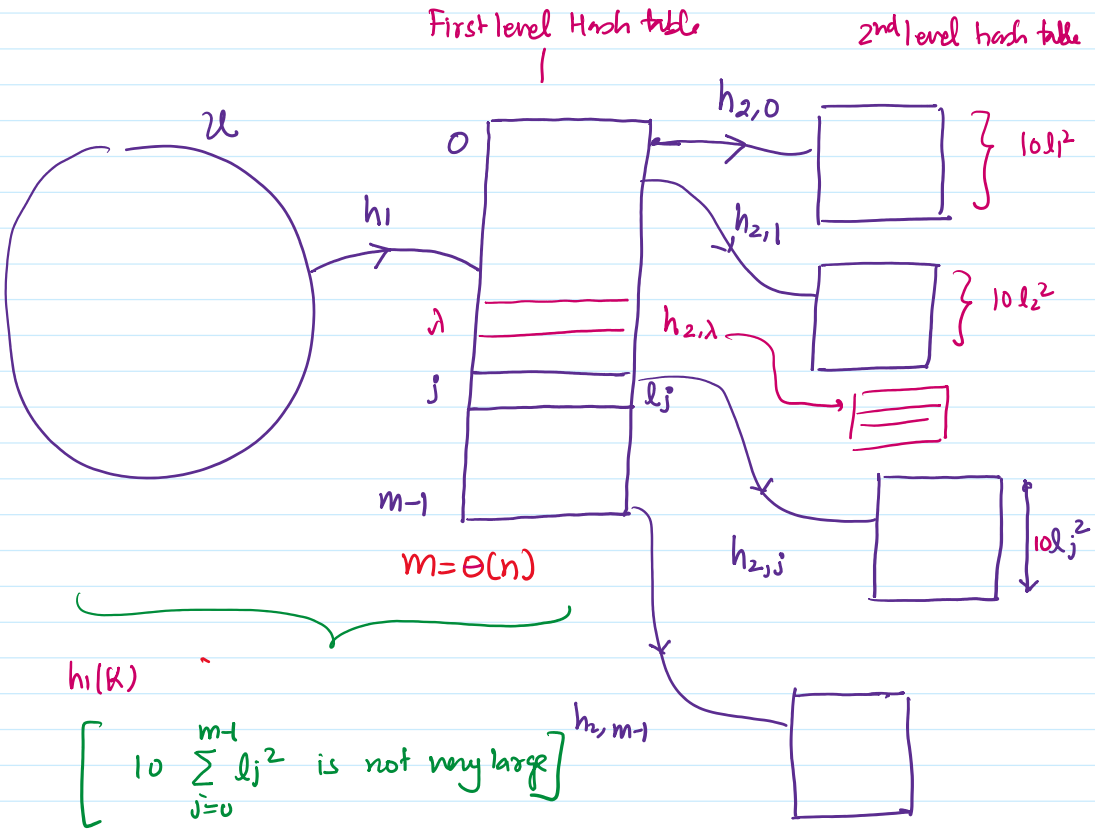
$$\Pr[\#\text{ of collision} > 1] < \frac{E[\sum_{i,j} I_{ij}]}{1} < \frac{1}{10}$$

$\Rightarrow$ w.p. at least $\frac{9}{10}$ there will not be any collision

Question: Coming up with data structure for static dictionary problem

with space complexity $O(n)$

Search " $O(1)$.



Step 1 Pick $h_1 : \{0 \cdots |U|-1\} \to \{0 \cdots m-1\}$ from universal hash family

Step 1.5 If $\sum_{j=0}^{m-1} l_j^2 > c'n$ then redo step 1

Step 2: For each slot in first hash table $j \in \{0 \text{ -- } m-1\}$

$l_j :=$ # of key mapped in j^{th} slot

Sample $h_{2,j} : \{0 \text{ -- } |2l|-1\} \longrightarrow \{0 \text{ -- } (10l_j^2 - 1)\}$

Rehash if there is collision at 2nd level

Step 2.5 Do the following for all $j \in [m]$

While $h_{2,j}(K_i) = h_{2,j}(K_{i'})$ for some j, $K_i \neq K_{i'}$ with $h_1(K_i) = h_1(K_{i'})$

Repick $h_{2,j}$

Similar to proof of lemma 1

$\Rightarrow$ Pr[# of collision > 1] $< 1/10$

$\Rightarrow$ w.p. $9/10$ there is no collision at 2nd level

Claim Total space complexity is linear in n

[1984 Fredman, Komlos, Szemeredi

FKS Algo

Perfect Hashing

Proof:

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

Space required for 2nd level of Hash table

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

$I_{ij} = \begin{cases} 1, & h_1(K_i) = h_1(K_j) \\ 0, & \text{o.w.} \end{cases}$

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

$$\leq \frac{10n^2}{m} \quad [\because m = \theta(n)]$$

$$\leq c \cdot n \quad \text{— (II)}$$

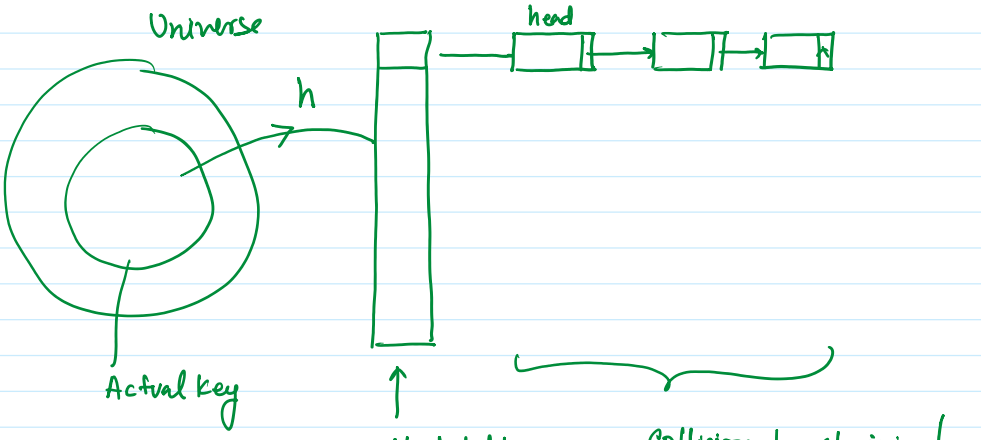
$\leq c \cdot n$ — (II)

Apply Markov on Eqn (II)

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

choosing c s.t

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



Direct level Hashing