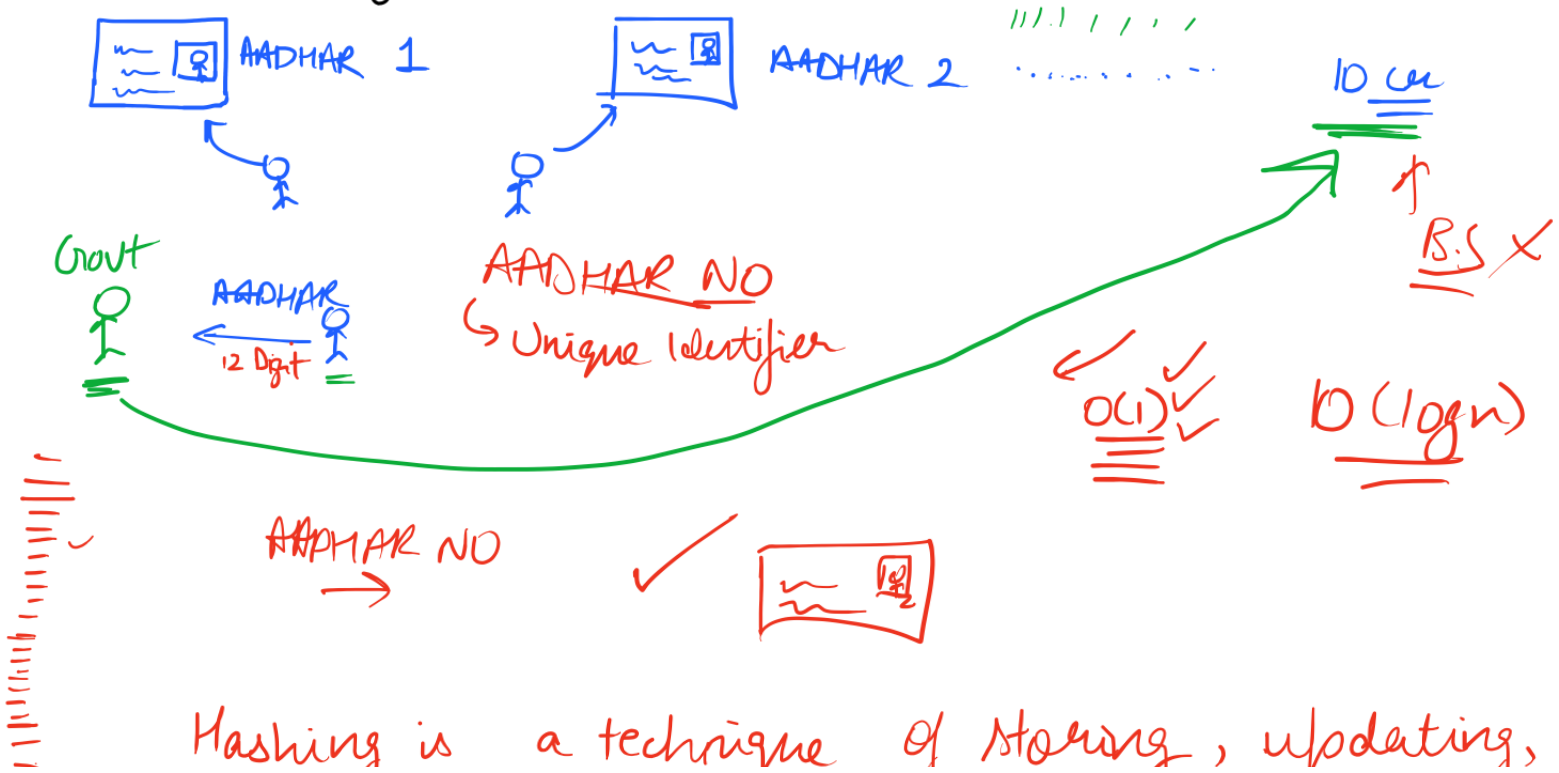
Friday, 12 July 2024
8:39PM

Hashing (4) → HashSet & HashMap

↳ What is hashing?

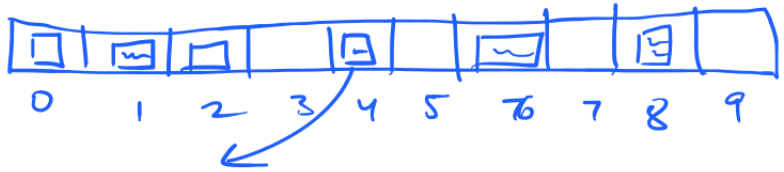


Hashing is a technique of storing, updating,

fetching info from a large chunk of data having a quick access and retrieval.

10 ✓

10 Aadhar (0-9)



→ 4

12

Arrays

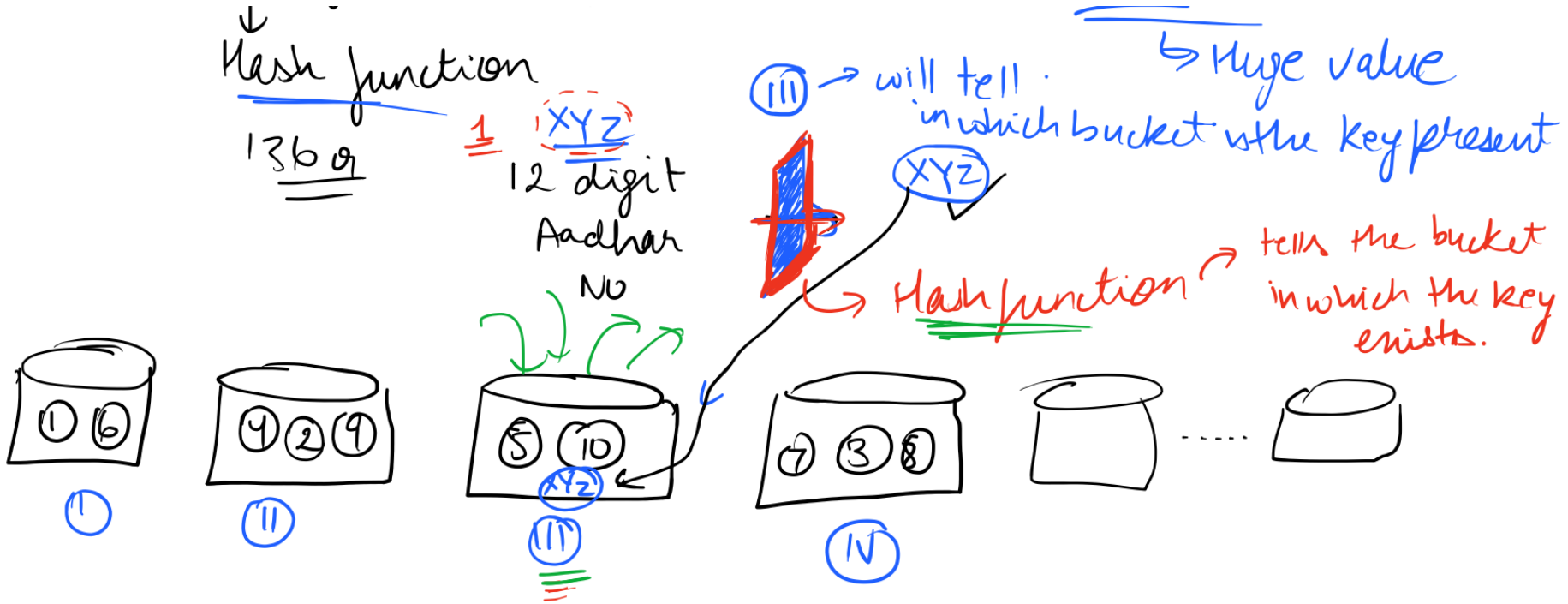
arr[i] = value

get O(1)
put O(1)
size → (0, n-1) ✓

0-99

3 → 0 - 999
12 → 0 - $(10^{12}-1)$

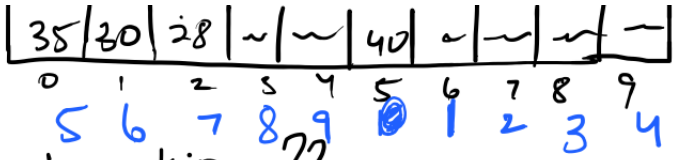
↳ Hashing → same type of data



Hash function ensures that a particular key is present in a specific bucket.

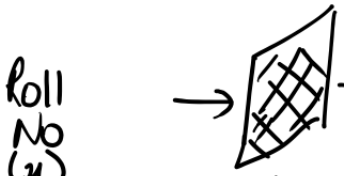
Why is hashing needed?
↳ optimise the searching in O(1)

→ Ideal Hashing



10

Hash function ??



$h(x) = x$ → Very fast + optimal

Info of student with roll no. x is present at a position.

$h(x) = 5$

↳ Bad hash fun

$h(x) = (x+5) \% 10$

$h(x) = (x \times 10^5 + 3x \times 6 * 10^8) \% 10$

x is the key

Any func^n which will tell me the bucket

⇒ We choose the hash function

↳ The performance of our hashing will depend on the hash function choosen.

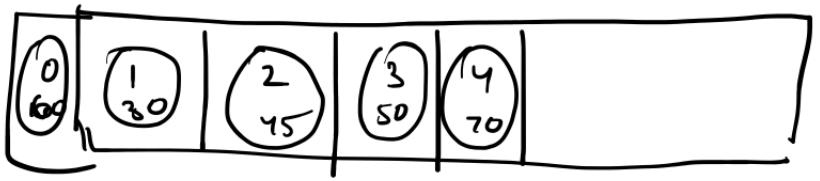
Student → Roll No, Marks

Array

→ O(n)

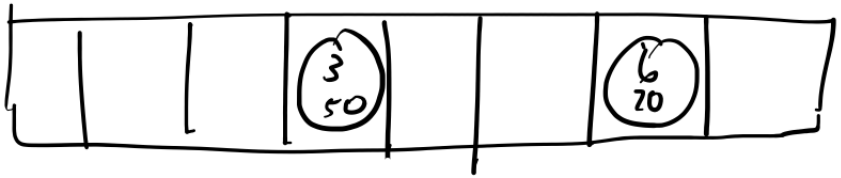
Ideal Hashing

1000



→ O(1)

↳ searching to O(1)



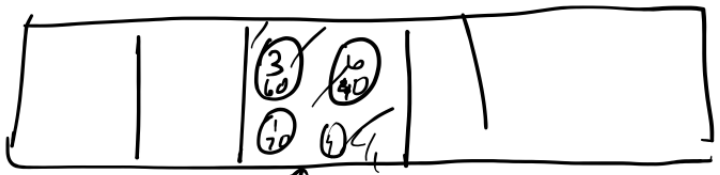
10 buckets

hash fun

↳ $h(x) = x$

Ideal hash function

Bad hash $h(x) = 2$



Searching $O(1)$??? ✗ No

2 ↳ searching is again $O(n)$

Limitation of Ideal Hashing

↳

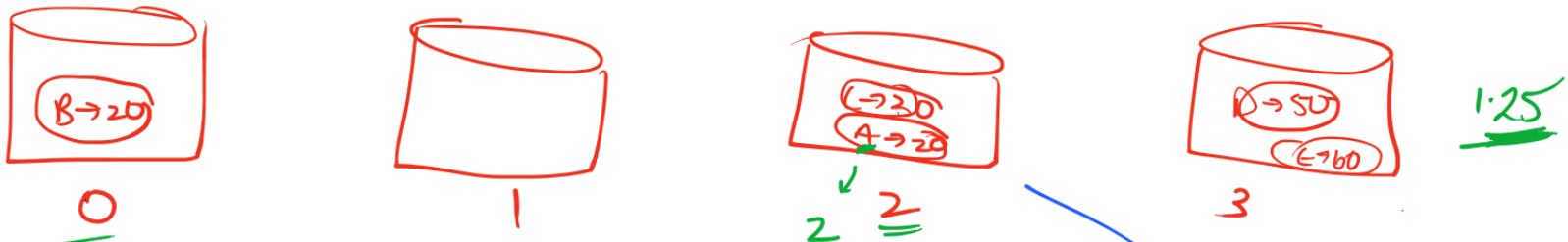
The size of table needed is very large if the range of keys is very large

$h(x) = x$

n pairs (A→10) B→20 C→30 D→50 E→60

$\frac{5}{4} = 1.25$

b buckets



B → 0

Key → → Bucket No

→ 2

More than 1 element is present in a single bucket

A

hash fn(x) → this should be in balanced way

↳ O(1) ✓

O(n) ✗

Key % buckets.size

100 % 4 = 0

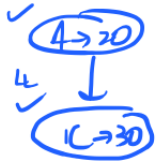
52 % 4 = 0

33 % 24 = 1

Collision

↓

Chaining to avoid collisions



every bucket will ha a chain (Linked Li

LF = Load Factor = Number of pairs / size of buckets

= n / b → Avg no of pairs in each bucket

n → No of pair

b → buckets

Put, get, Remove → We approximately visit the average size of pairs in a bucket.

↳ O(LF)

$LF = \frac{n}{b}$ → when n keeps on increasing, the avg pairs in each will increase.

1.25

4

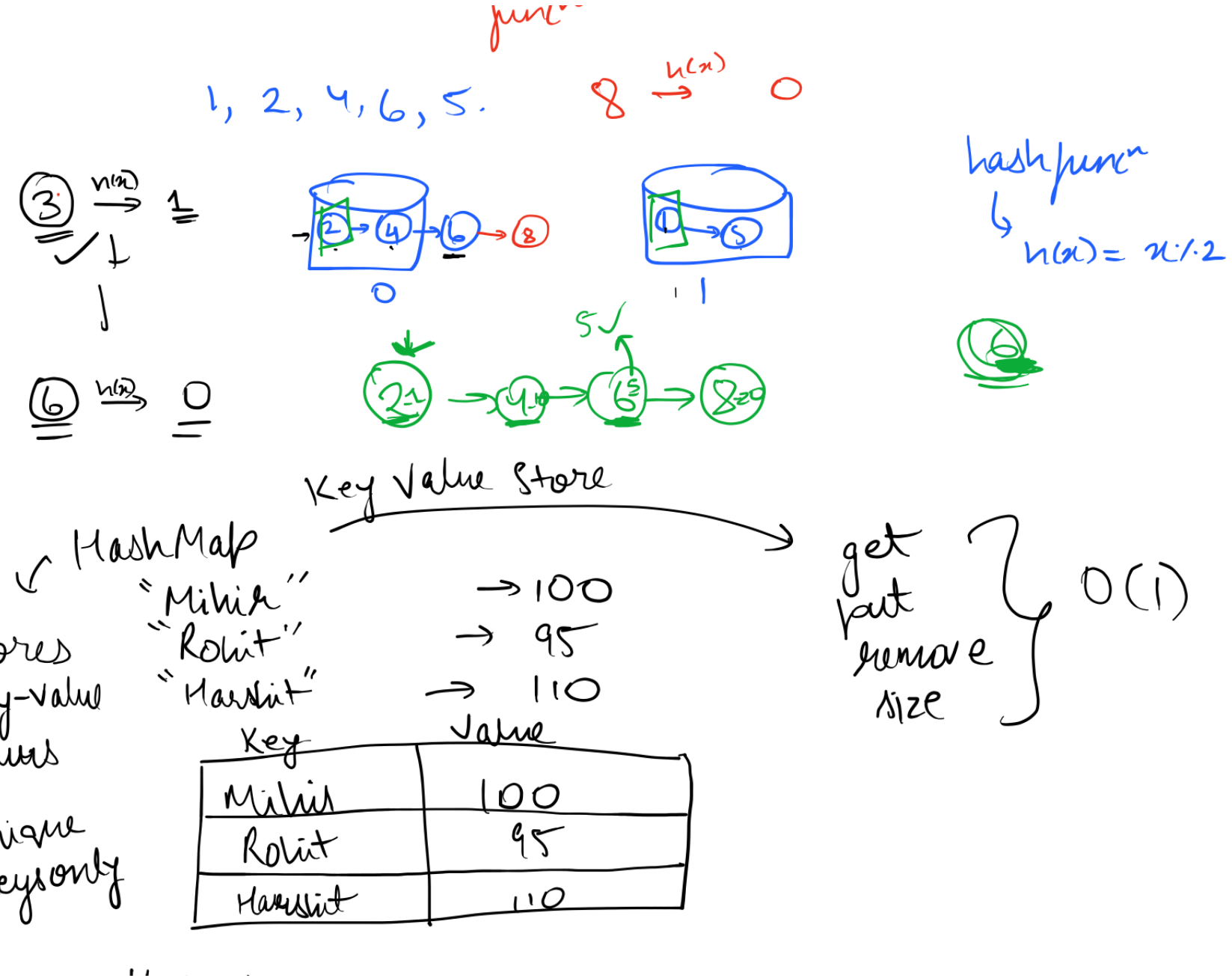
$O(LF) = O(4)$

Time complexity changed from $O(1.25) \rightarrow O(4)$

⇓

We can increase the buckets

$LF = \frac{16}{10} = 1.6$


1, 2, 4, 6, 5.
8 → h(x) 0
hash func^n
h(x) = x % 2
3 h(x) 1
6 h(x) 0
0
1
5
Key Value Store
HashMap
"Mihir" → 100
"Rohit" → 95
"Harshit" → 110
Stores Key-value pairs
→ Unique keys only
Key
Value
Mihir
100
Rohit
95
Harshit
110
get
put
remove
size
O(1)

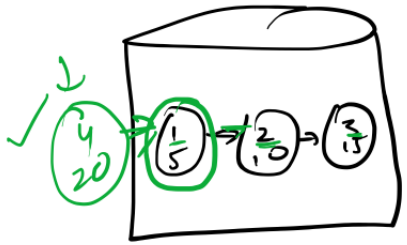
Hash set → Stores unique values

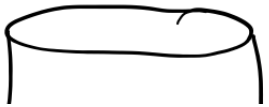
↳ contains, add, remove } O(1)

↳ is like Hashmap where key is anything and value is true or false

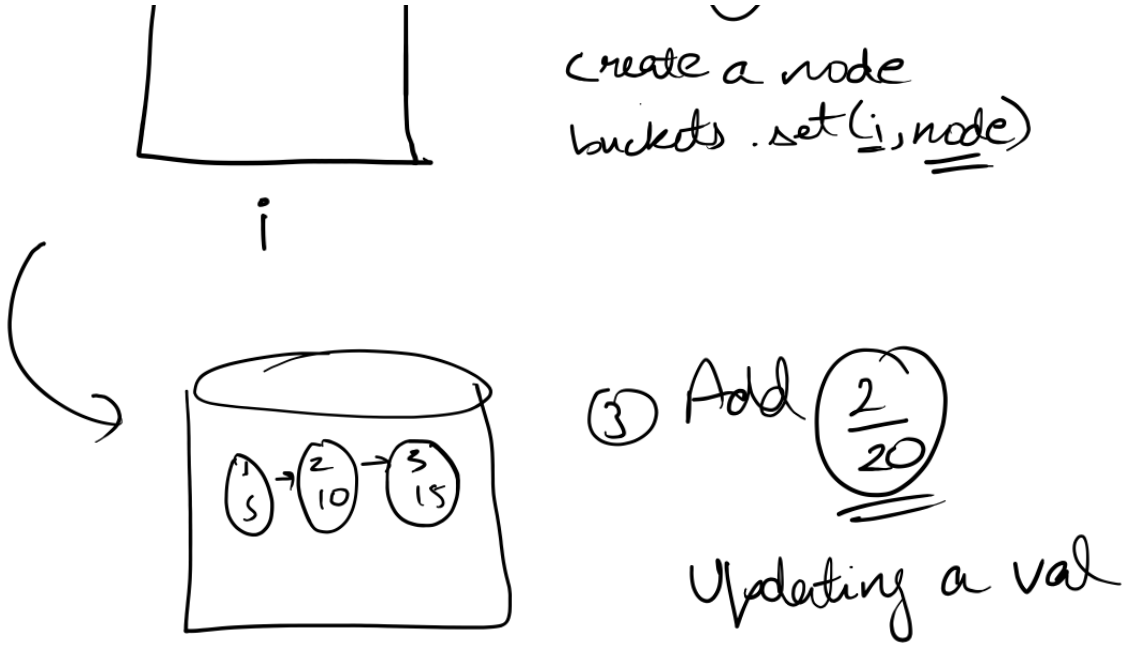
Implementation / Design a Hashmap

① → Add (4, 20)





② Add (4, 20)



Rehashing

↳ In case of put

↳ After adding many pairs the LF can change

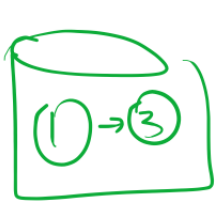
⇓

Rehash

Double the total no. of buckets to improve the performance of each bucket.

⇓

(LF) avg no. of elements in each bucket will decrease



✕ ⇓ hash function has changed