3.4 - Hashing

Sunday, July 13, 2025 11:29 AM

→ Linear - $O(n)$

→ Binary → $O(\log n)$

→ Technique which helps you perform search in an optimized manner ($O(1)$)

Functions

$y = f(x)$ → independent variable

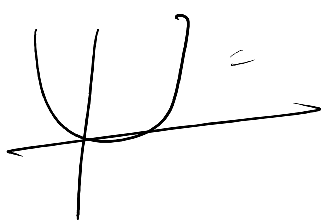
y ↓ dependent variable

$y = x^2$ parabolic func^n

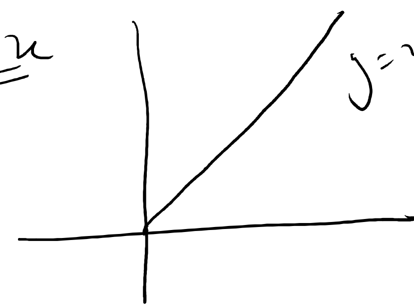
$x=1$, $y=1$

$x=2$, $y=4$

$y = f(x)$



$y = x$



$y = x$

$y = x \% 10$

⇒ $x = 13$, $y = 3$, $x = 21$, $y = 1$

$\boxed{y = x \% \varepsilon}$ ⇒ $[0, \varepsilon - 1]$

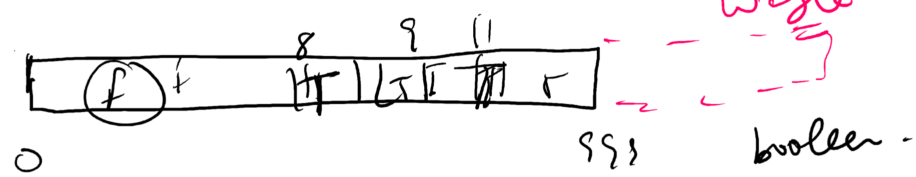
$6 \% 10$

$0 - 9$

Concept of Hashing

x

⇒ {8, 9, 11, 10, 5}



O(1) { if (arr[x] == true) present
else Not present.

5 18

⇒ Technique of Hashing

hash function

$x \rightarrow \boxed{f(x)} \rightarrow y$

Key

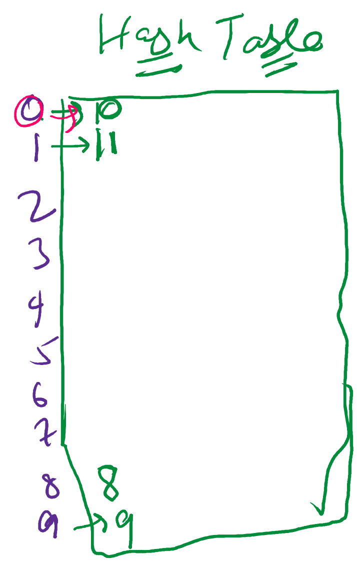
x%10

hash value

$\underbrace{\{8, 11, 10, 9\}}_{\text{keySet}} \rightarrow \{8, 1, 0, 9\}$

10 18 → [x%...]

O(1)



⇒ Size of Hash Table depends on hash function

$f(x) = x \% \Sigma \Rightarrow [0 - \Sigma-1], (\Sigma)$

$f(x) = x \% E \Rightarrow [0 - E-1] \rightarrow E$

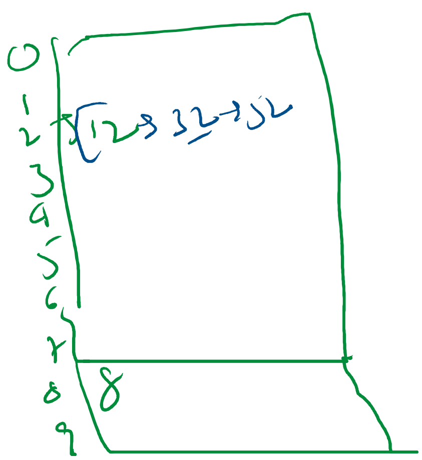
$\Rightarrow$ Collision

→ same values for diff multiple keys

$\{8, 12, 32, 9, 11\} \rightarrow \boxed{x \% 10}$

↓

$\{8, 2, 2, 9, 1\}$



[Handling collision]

Open hash — linked hashing

→ o → o → o → o

Closed hashing

→ Linear probing

→ Quadratic probing

If the no. of collisions are high, then the complexity in open hashing goes to $O(n)$.

Closed hashing

⇒ Linear probing ⇒

i

$h(x) + f(i) \Rightarrow$

$$\left[f(x) = [h(x) + i] \right]$$

$h(x) = x \% 10$

$f(x) = [x \% 10 + i] \Rightarrow i = 0, 1, 2 \ldots$

Ex: {8, 12, 7, 22, 14, 21} 104, 224

$f(x) [x \% 10 + 0]$

$f'(x) [x \% 10 + 1] =$

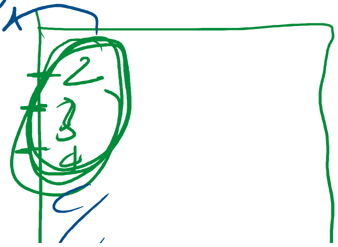
] +1

] +1

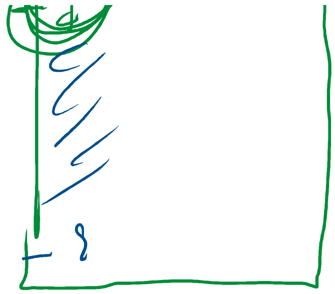
] +1

0	
1	21
2	12
3	22
4	14
5	104
6	
7	7
8	8
9	

{22, 32, 42, ...} clustering



Problems of linear (Clustering)



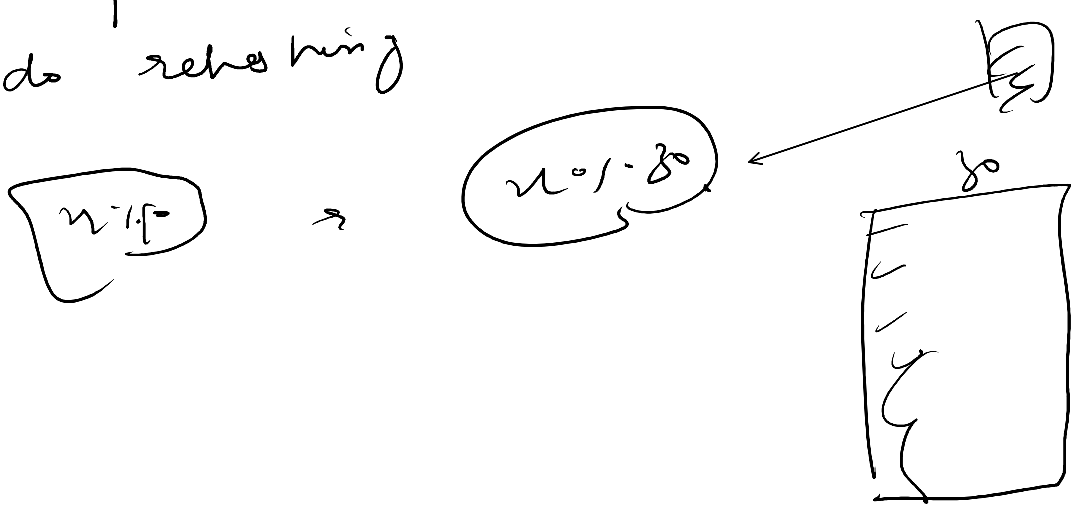
(Clustering)

Load Factor

⇒ No of entries / size of the hash table

⇒ a good hash func keeps the load factor below 0.75.

⇒ When load factor becomes large, we do rehashing



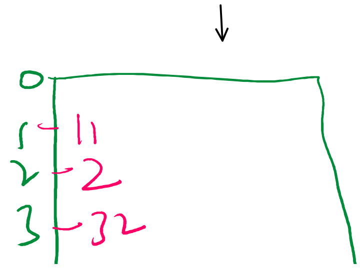
⇒ Quadratic Probing

$$f(x) = [\underbrace{h(x)}_{x\%10} + i^2]\%10 \quad ; i = 0, 1, 2 \ldots$$

{8, 12, 11, 32, 14, 102}

9 + 16 = 25%10

5



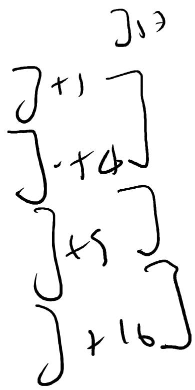
{8, 12, 11, 32, 14, 102}

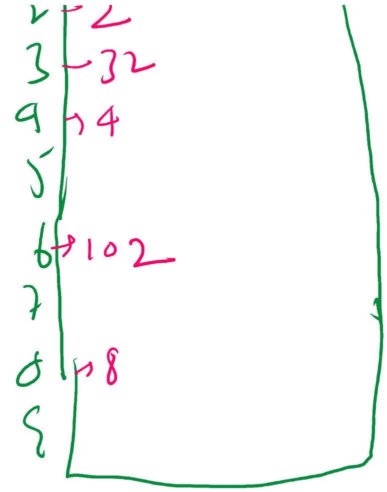
$2+0^2$

$2+1^2$

$2+2^2$

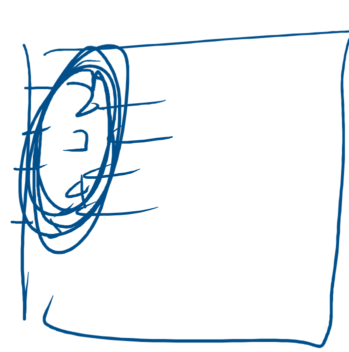
$2+1^2 =>$

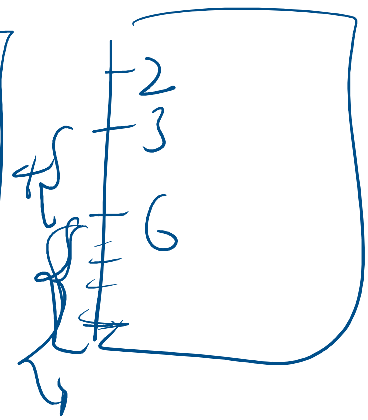




{22, 32, 42, 52}

L | Q





Data structures in Java based on the concept of Hashing

=> HashSet

=> HashMap

Based on the concept of Hashing

↓
Based on the concept of Hashing
HashTable

Hash Set

⇒ DS which only stores unique values.

{ 2, 3, 5, 2, 4, 3 }

⇓

{ 2, 3, 5, 4 }

⇒ Hash set stores values in random order

TC
insertion → O(1)
search → O(1)
Delete → O(1)

Hash Map
↳ collection of key-value pairs.

{ Rohit → 40 } key-value
Kohli → 60

{ Rohit → 40
Kohli → 60
KL Rahul → 0 } key-value

⇒ keys are always unique
but values can be duplicate

T.C insertion → O(1)
read → O(1)
deletion → O(1)

→ How to iterate on HashMap

⇒ mp.keySet()
⇒ mp.values()

→ for each loop
because there is no concept of indexing.

map.put(key, value)
map.get(key) $\xrightarrow{present}$ value
→ null

(*) While doing a get operation on a map, always first check whether the key is

(4) When ... always first check whether the key is present or not.

int x = map.get(key)

null pointer exception