Hashing → useful for searching

Linear Search – O(n) — random

A	8	5	12	6	15	9	4	3	7	10
	0	1	2	3	4	5	6	7	8	9

Binary Search – O(log n) — Sorted order

A	3	4	5	6	7	8	9	10	12	15
	0	1	2	3	4	5	6	7	8	9

O(1) ↓ requirement

Hashing.

Keys – 8, 3, 13, 6, 4, 10, 55

A				3	4		6		8		10			13
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

O(1)

Key 10

Key 12

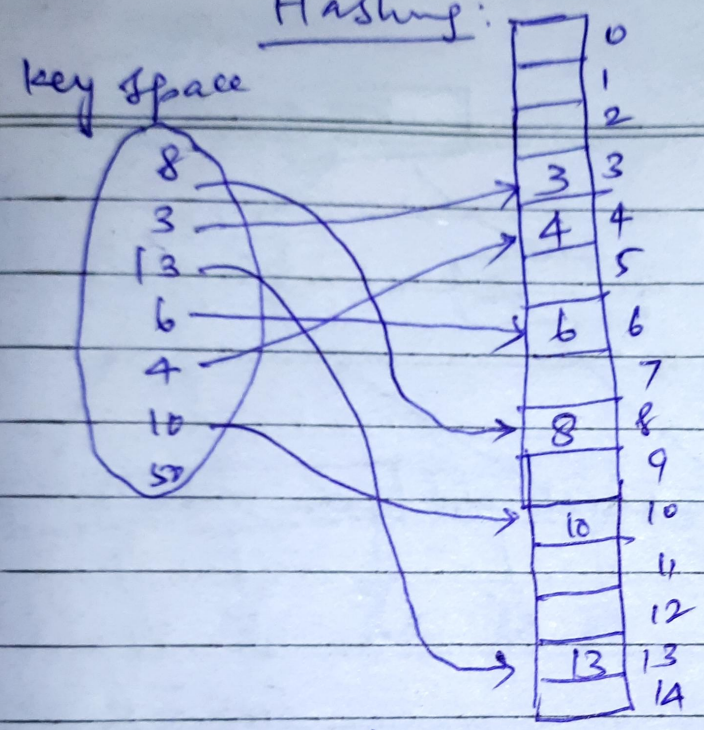
Disadvantages: → require more memory space.

→ lot of memory are empty

②

TNPL பக்கம் :
நாள் : / /

Hashing:



$h(x) = x$

one to one

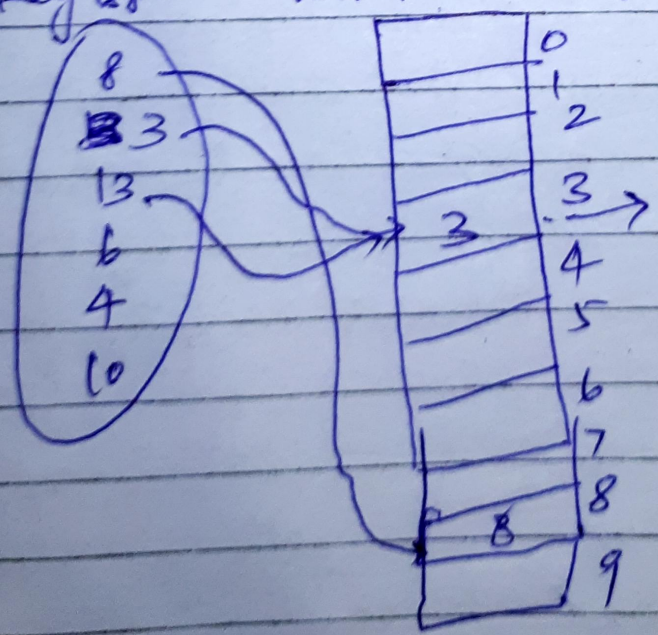
$h(x) = x$

many-one.

Hash function:

$h(x) = x \% 10$ → example

key space hash table size of hash table. size = 10



$h(x) = x \% size$

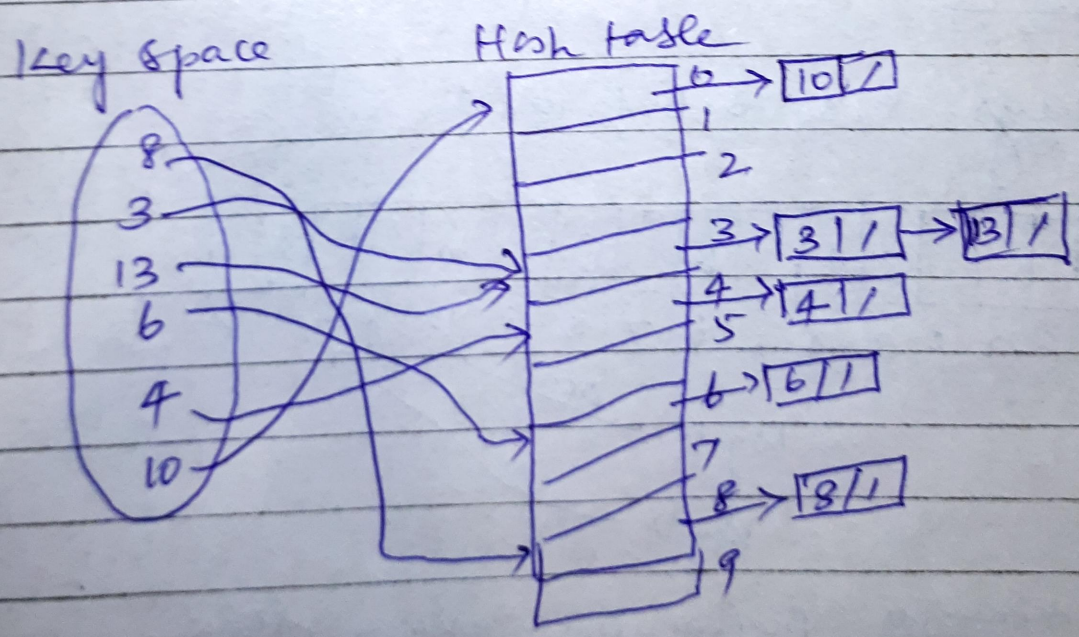
TNPL பக்கம் :
நாள் : / /

Collision Resolution

* Open hashing
  chaining
* Closed hashing
  open addressing
  * linear Probing
  * quadratic "

① **Chaining:** → linked list

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

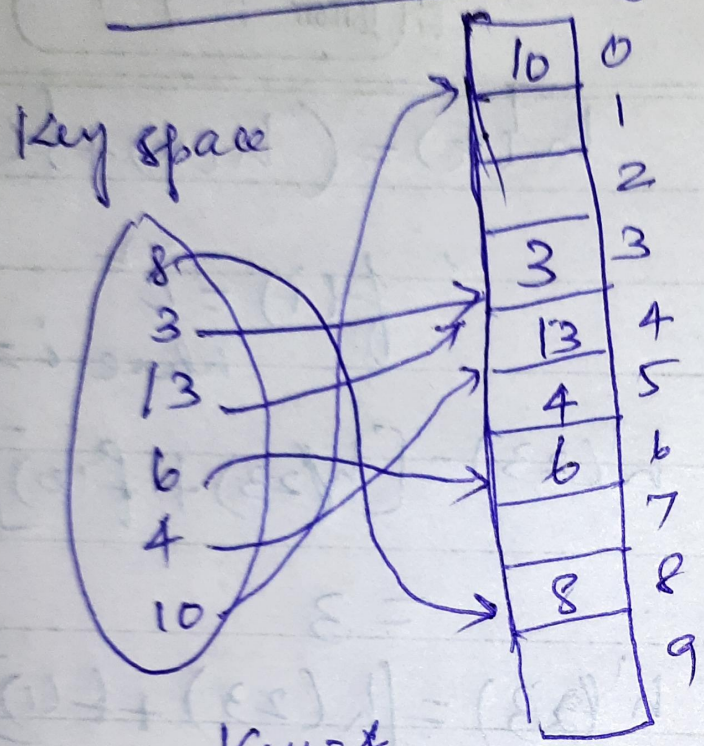


key = 13

$13 \% 10 = 3$

time complexis not O(1) but less than O(logn)

Linear Probing:



Key = 4

Key = 24

$h(x) = x \% size$

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

$f(i) = i$ where $i = 0, 1, 2 \ldots$

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

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

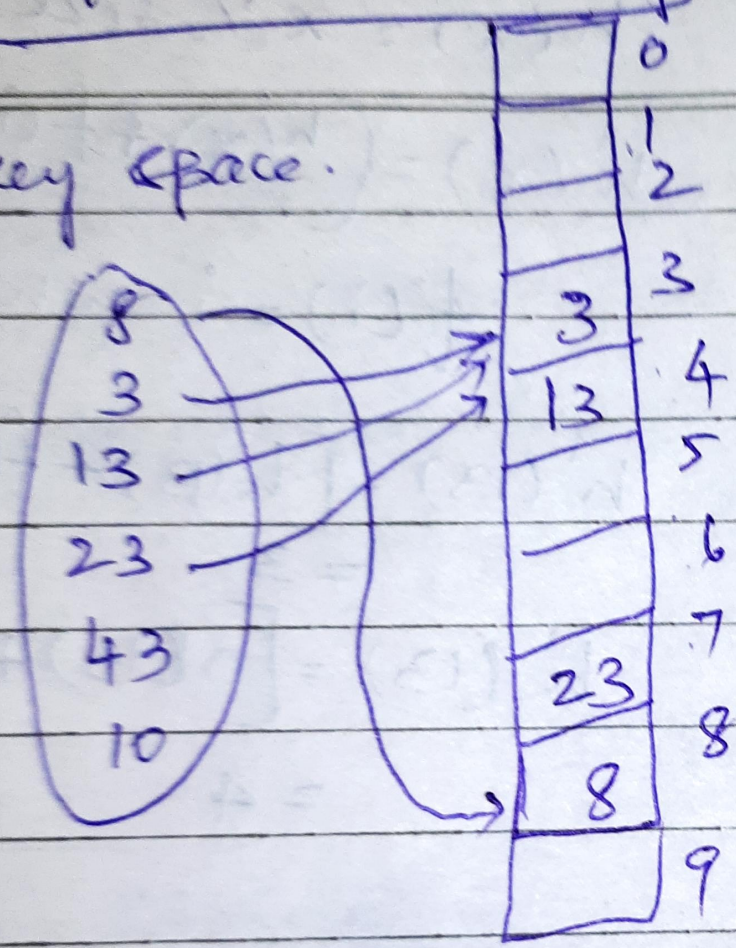
- To check the ~~to~~ next ~~space~~ index and if not found next index until it is empty.

Time complexity is not ~~or~~ $O(1)$ but less than $O(\log n)$.

TNPL பக்கம் :
நாள் : / /

Quadratic Probing

Key space.



$h'(x) = (h(x) + f(i)) \% 20$

$f(i) = i^2$ where $i = 0, 1, 2, \ldots$

$h'(23) = [h(23) + f(0)] \% 10$

$= 3$

$h'(23) = [h(23) + f(1)] \% 10$

$= (3+1) \% 10 = 4$

$h'(23) = [h(23) + (f(2))] \% 10$

$= 3 + 4 = 7$