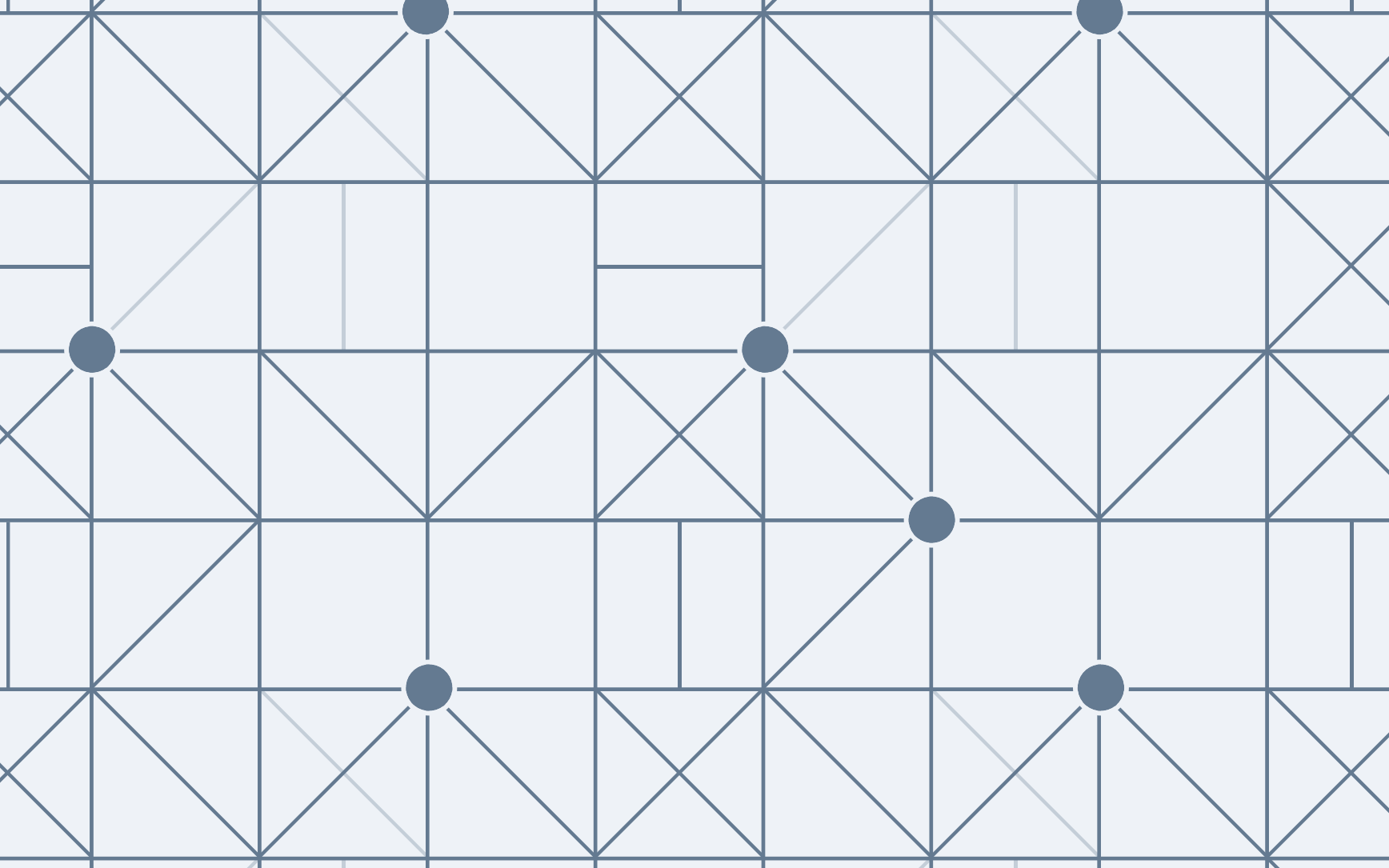
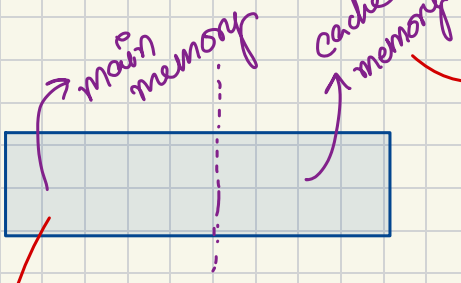


Imp

Agenda

1. LRU Cache
2. Snapshot array
3. Longest substring with equal freq of 0, 1 & 2

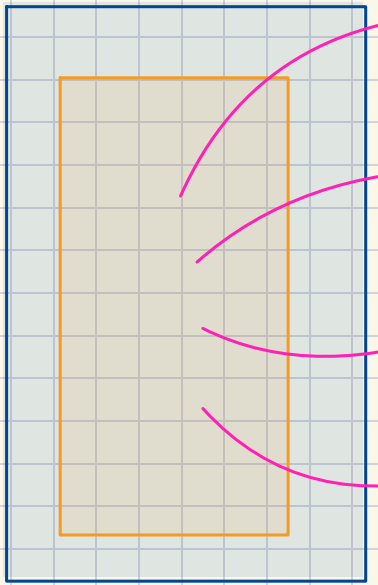
LRU Cache



applications running in background.

RAM → everything runs on RAM

applications running in foreground

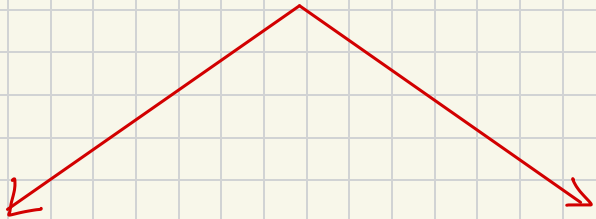


eventually this cache memory will get-filled, won't accomodate more ?

Cache Memory

App1, App2, App3,

Cache Memory Managment System

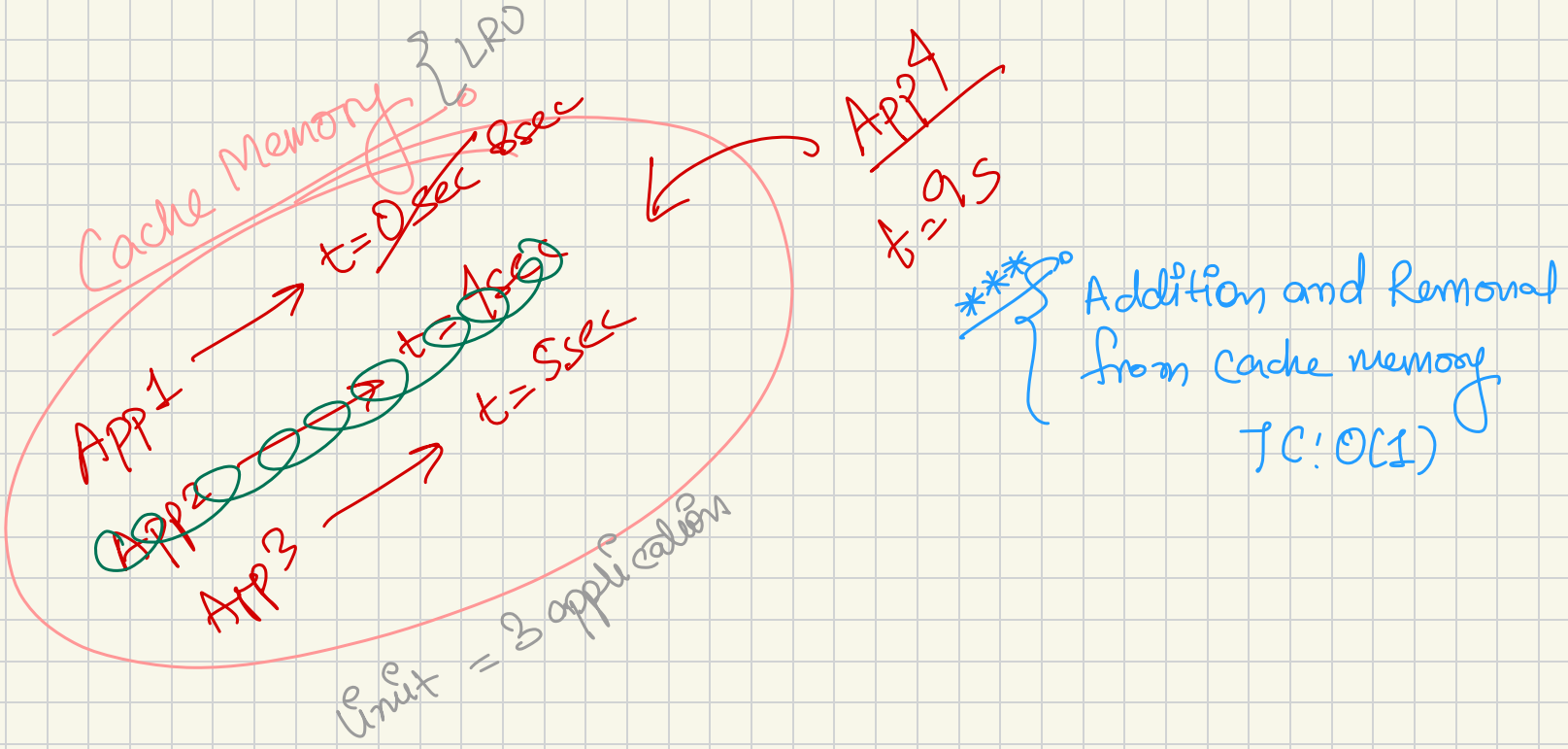


LRU

{ Least recently used }

LFU

{ Least frequently used }

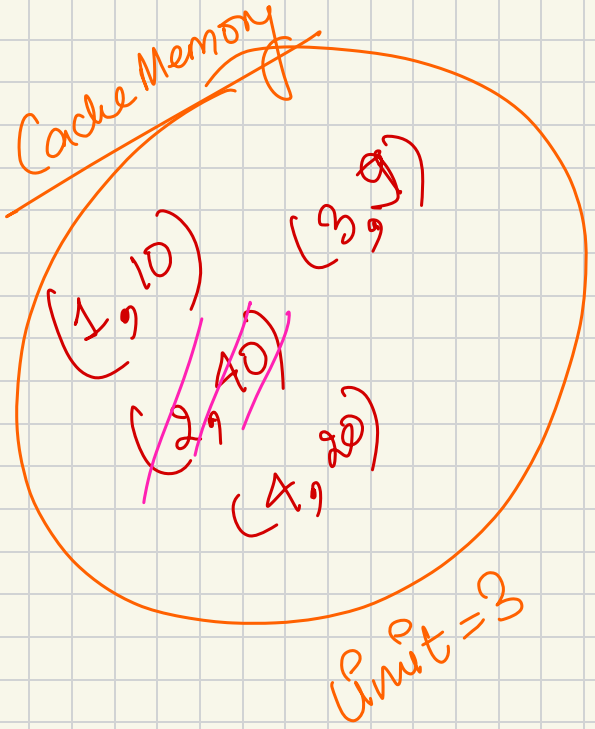

Cache Memory
LRU
APP 1
t=0sec
APP 2
t=4sec
APP 3
t=5sec
Limit = 3 applications
APP 4
t=9.5
***
Addition and Removal from cache memory
TC: O(1)

```
class LRUCache {
    // your code here
    public LRUCache(int capacity) {
        // your code here
    }

    public int get(int key) {
        // your code here
    }

    public void set(int key, int value) {
        // your code here
    }
```

defines max. capacity of Cache Memory

read, value against an application

opens/update an application in Cache memory



set (1, 10)

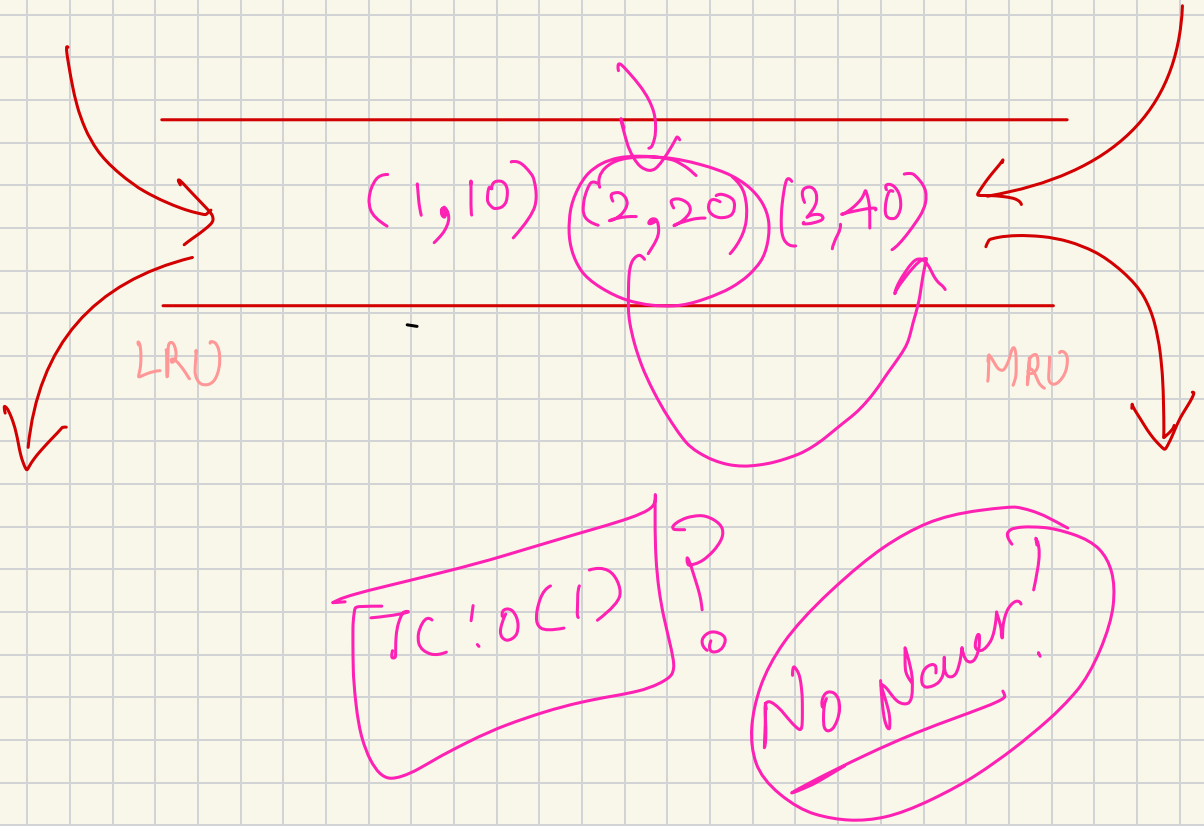
set (2, 40)

set (3, 7)

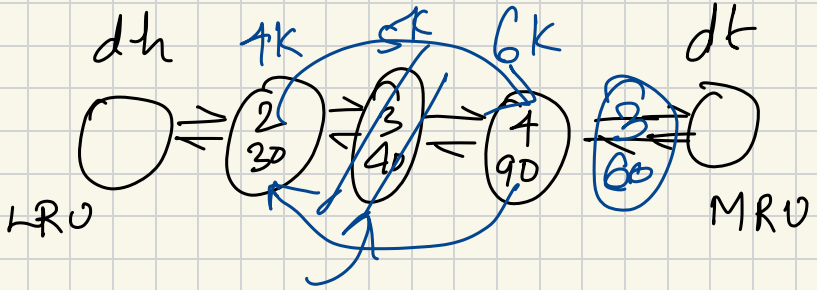
get (1) ⇝ 10

set (3, 9)

set (4, 20)


(1,10)
(2,20)
(3,40)
LRU
MRU
TC: O(1)
?
No Never!

Doubly Ended LinkedList :- Eg HashMap.



removeNode()
addLast()

set(1, 20)

set(2, 30)

set(3, 40)

set(4, 90)

set(3, 60)

HashMap

Key	Add
2	1K
3	5K
4	6K

Snapshot Array → Unit testing

int[] arr = { 0 1 2 3 4 5 6 7
0, 0, 9, 0, 2, 0, 0, 0 }

snap_id = ~~0~~ ~~1~~ 2

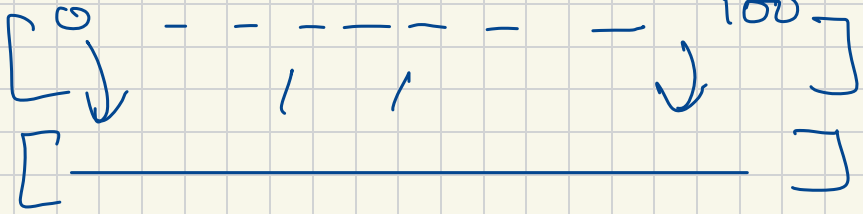
set(2, 8)
set(4, 2)
snap()
set(2, 9)
snap()
get(2, 1) ⇝ 9
get(4, 0) ⇝ 2

snap_id 0 = { 0 1 2 3 4 5 6 7
0, 0, 8, 0, 2, 0, 0, 0 }

snap_id 1 = { 0 1 2 3 4 5 6 7
0, 0, 9, 0, 2, 0, 0, 0 }

get(3, 1) ⇝ 0

Brute Force :



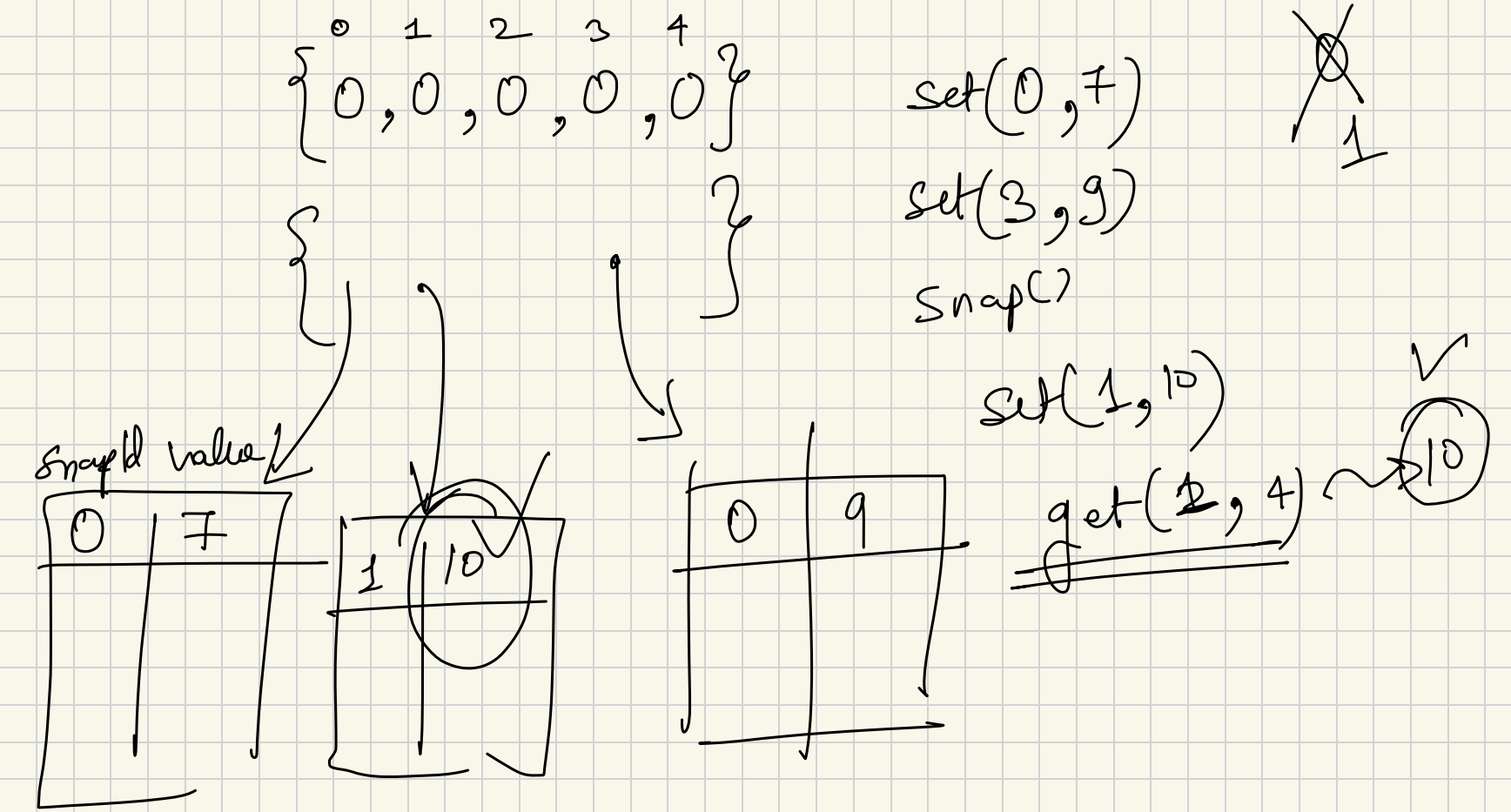
HashMap

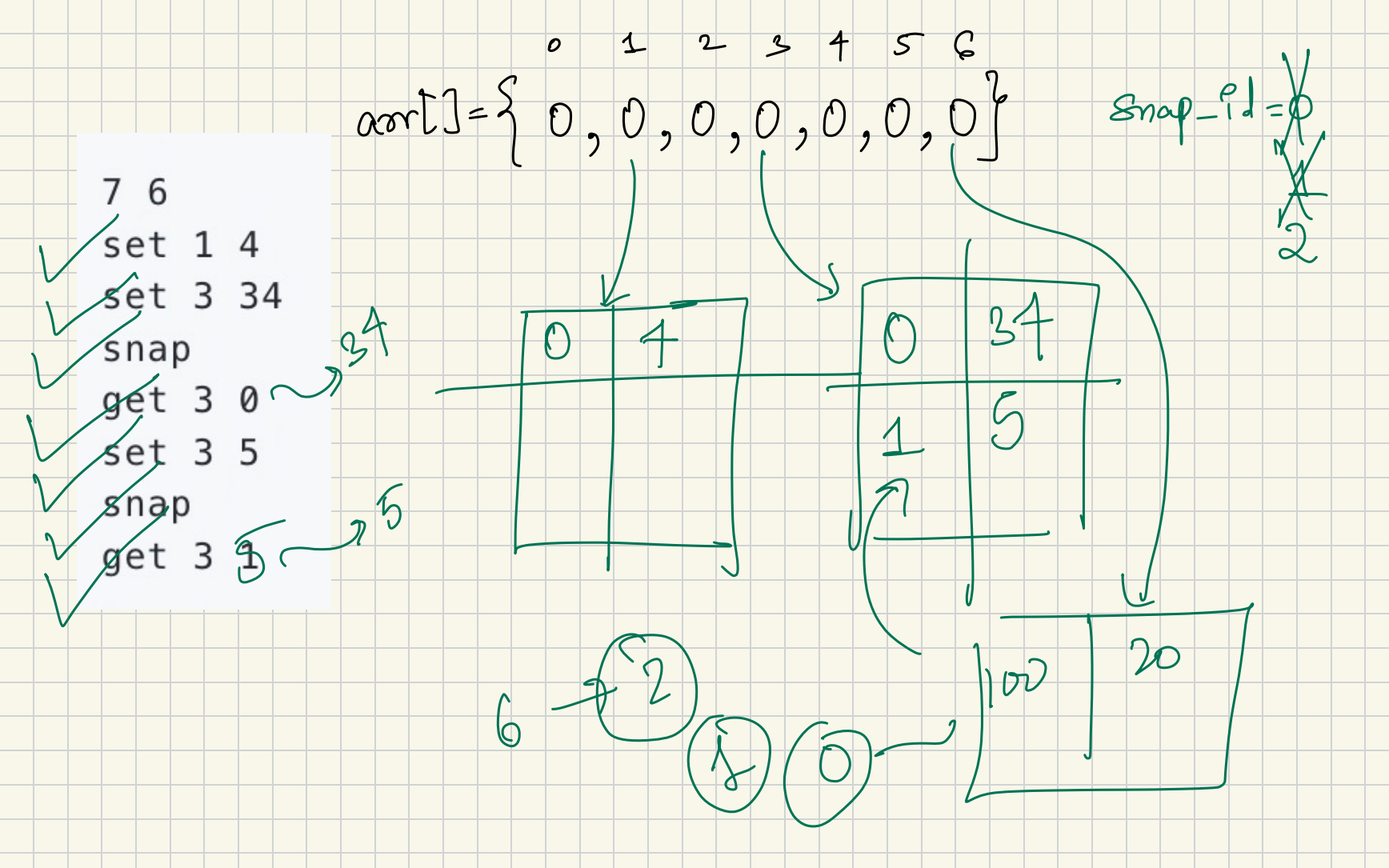
snap_id (Key)

array {Snapshot} (Value)

Drawback

{ Lot of Memory is used without any delta }


0 1 2 3 4
{0,0,0,0,0}
set(0,7)
set(3,9)
snap()
set(1,10)
get(2,4)
1
10
snapId value
0 7
1 10
0 9


7 6
set 1 4
set 3 34
snap
get 3 0
set 3 5
snap
get 3 1
0 1 2 3 4 5 6
arr[] = { 0, 0, 0, 0, 0, 0, 0 }
snap_id = 0
1
2
0 4
0 34
1 5
34
5
6
2
1
0
100 20