Longest subarray with equal freq of 0's, 1's and 2's.

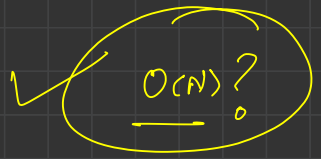
arr[] : {1, 1, 2, 0, 1, 0, 1, 2, 1, 2, 2, 0, 1}

Brute force

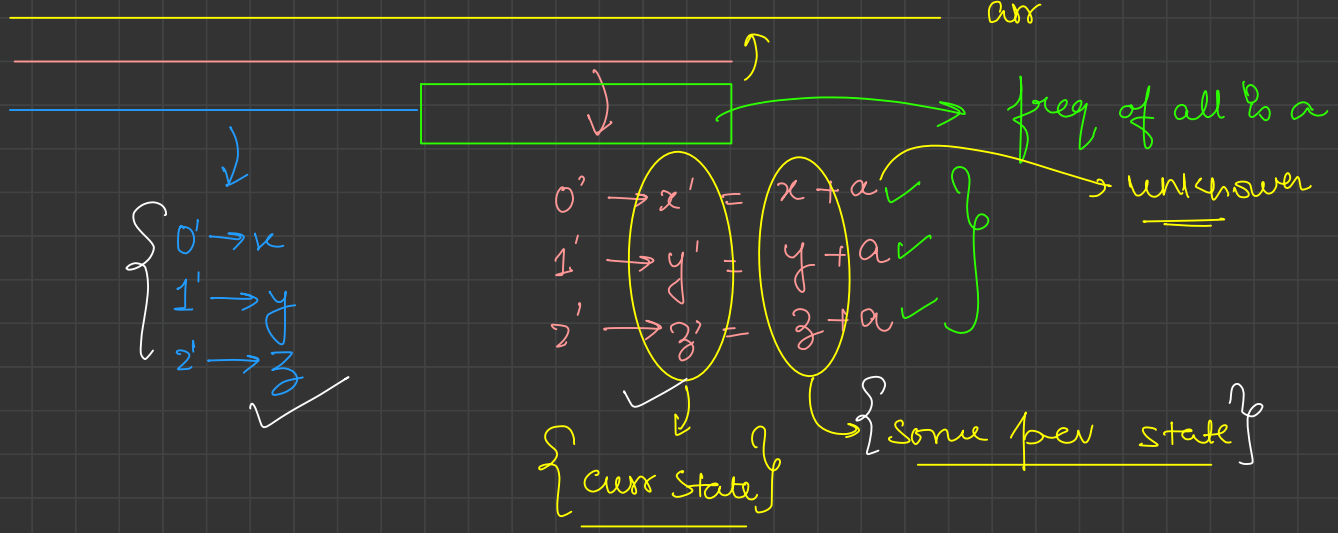
↳ find all subarray with freq 0's, 1's & 2's, Calc longest of them.

TC: $O(N^2)$	SC: $O(1)$

✓



arr[] = {1, 1, 2, 0, 1, 0, 1, 2, 1, 2, 2, 0, 1}



$x' - x = a$, $y' - y = a$, $z' - z = a$

arr[]: {1, 1, 2, 0, 1, 0, 1, 2, 1, 2, 2, 0, 1}

0, 0, 0

0, 1, 0

0, 2, 0

TC: $O(N^2)$

Curr State

$$x' = x + a \quad \text{---} (1)$$
$$y' = y + a \quad \text{---} (2)$$
$$z' = z + a \quad \text{---} (3)$$

(2) − (1)

$$y' - x' = y - x$$

(3) − (2)

$$z' - y' = z - y$$

String code

↳ $(y-x) \# (z-y)$

arr[] = {1, 1, 2, 0, 1, 0, 1, 2, 1, 2, 2, 0, 1}

x	0	0	0	0	1	1
y	0	1	2	2	2	3
z	0	0	0	1	1	1
y-x	0	1	2	2	1	2
z-y	0	-1	-2	-1	-1	-2

code 0#0 2#-2 1#-1 1#-1 2#-1 2#-2

0#0 ⟶ -1

1#-1 ⟶ 0

2#-2 ⟶ 1

2#-1 ⟶ 2

len = curr idx - first state

∴ 3 - 0 = 3

Rabbits in forest

arr[] = { 2, 2, 3, 1, 0, 2, 2, 3, 1 }

4/3 ~ 1+1

a ✓ ✓ ✓

b ✓ ✓

c ✓ ✓

d ✓

e ✓

4/2 = 0 +1

2/2

freq

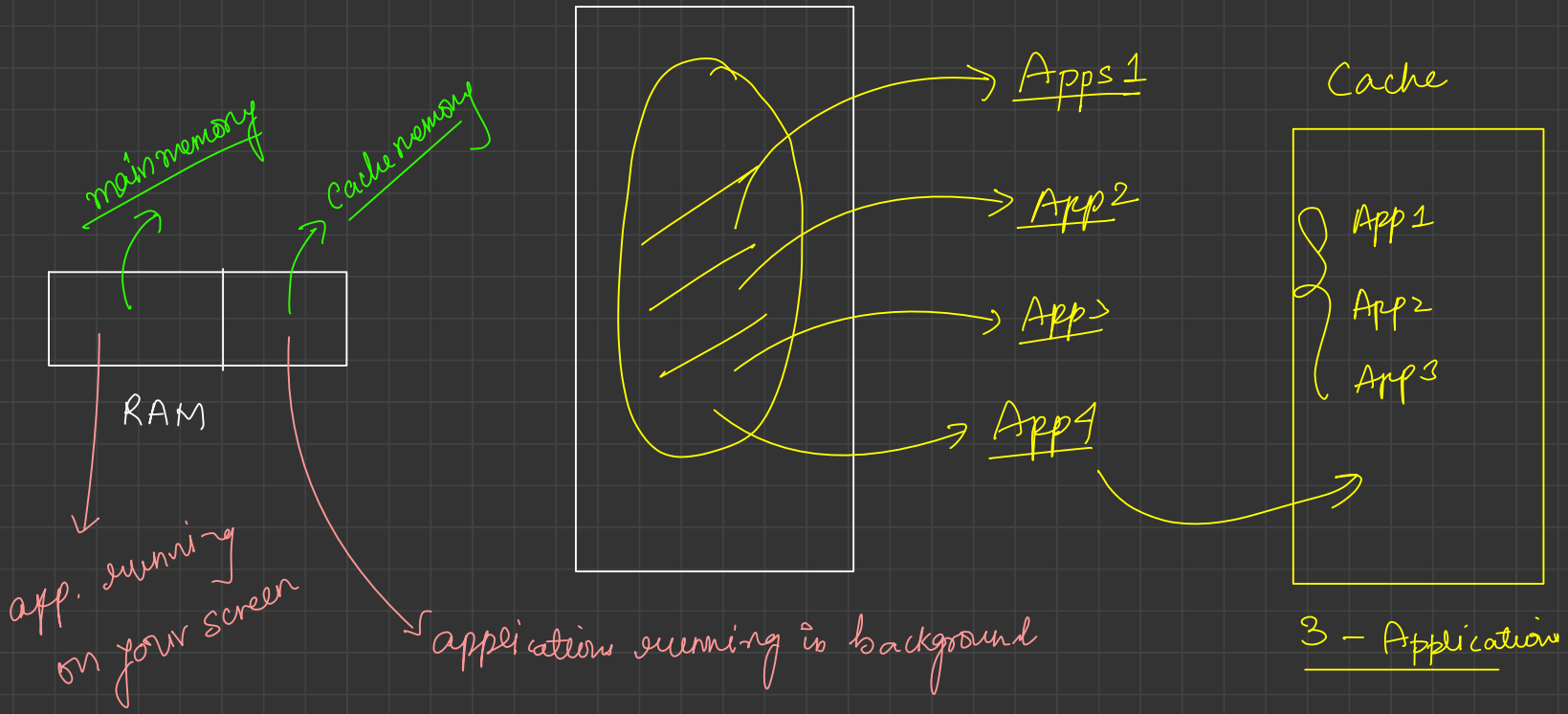
2 (3) → 4 → 2

3 (4) → 2 → 1

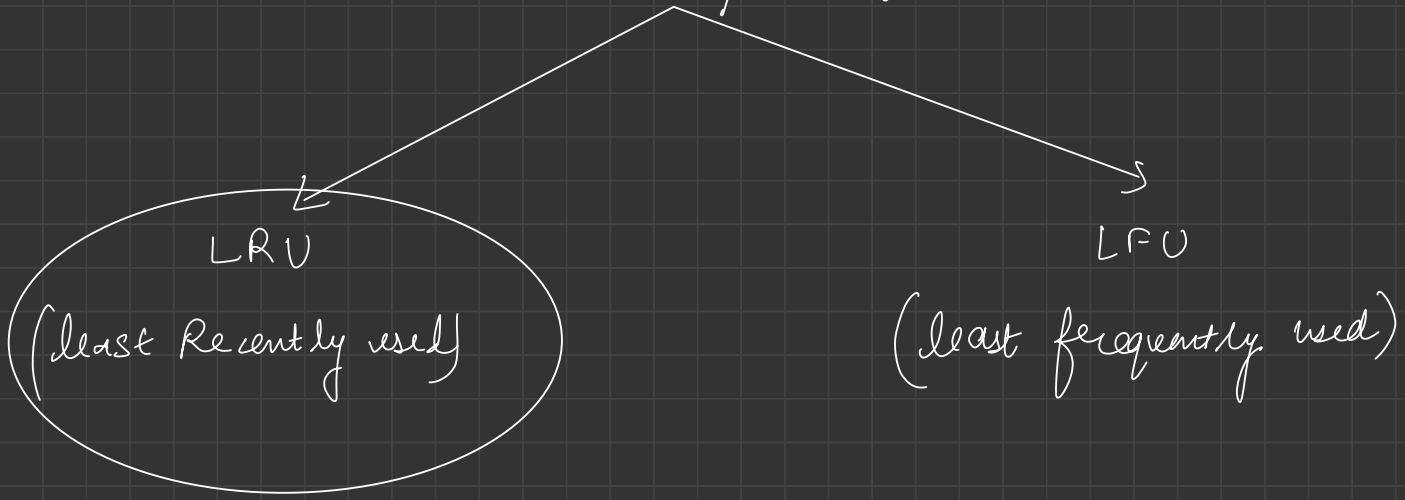
1 (2) → 2 → 1

0 (1) → 1 → 1

LRU Cache


main memory
Cache memory
RAM
app. running on your screen
application running in background
Apps 1
App 2
App 3
App 4
Cache
App 1
App 2
App 3
3 - Application

Cache Memory Managment



opened → moving to cache memory

0s App1 → opened

2s App2 → opened

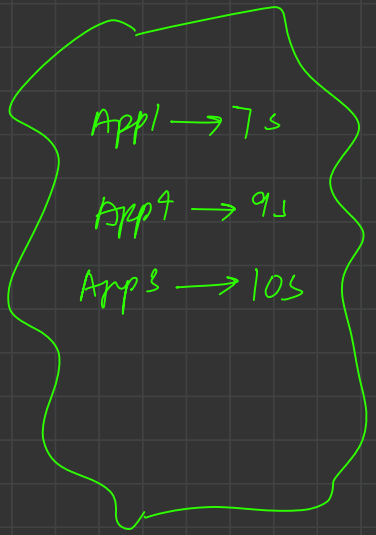
5s App3 → opened

7s App1 → re opened

9s App4 → opened

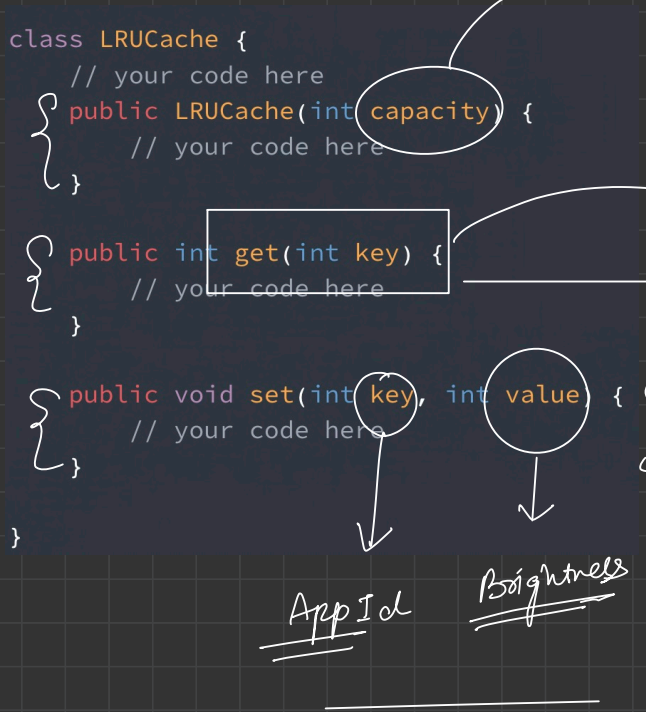
10s App3 → popup

(Mobile Phone)

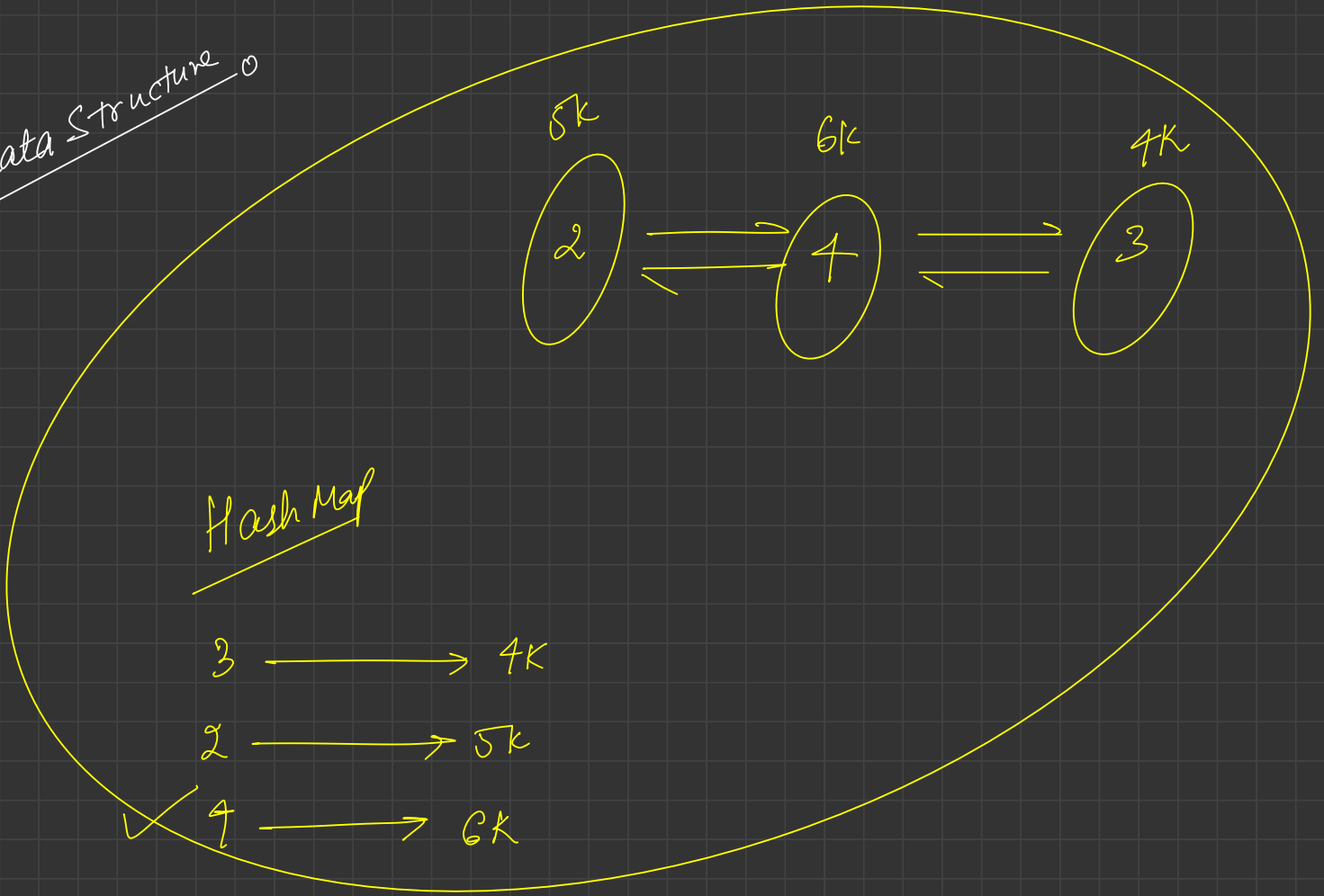


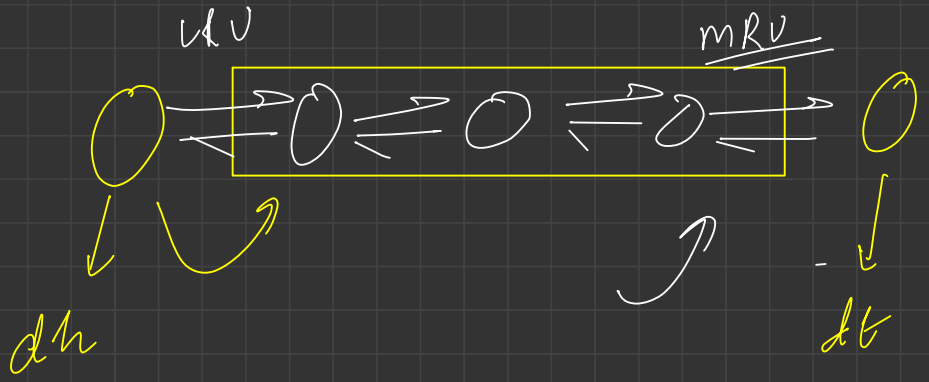
Cache (LRU)

(Limit = 3app)


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
}
}
max^m app. cache memory can hold
read O/P | pop up
move this app to most recently used
add new app. to cache memory, or update prev app. with new a value.
(Reopned)
AppId
Brightness

data Structure





LRU → head.next

addLast()

remove()

moveToLast()

1

2

Snapshot Array.

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

set (1, 3) ✓
set (4, 5) ✓

snap()

set (1, 5) ✓

snap()

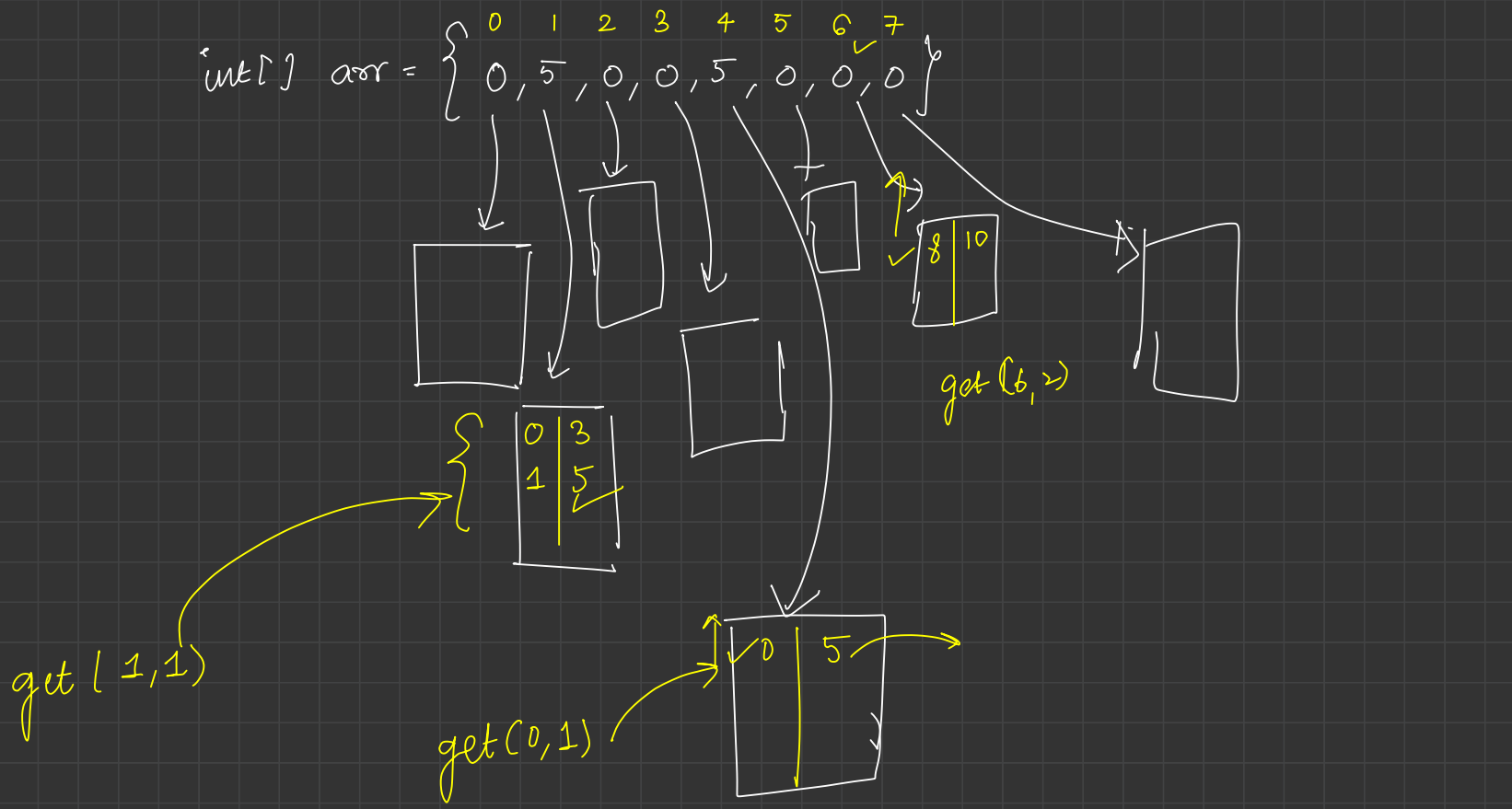
get (1, 1) → 5 ✓

get (1, 0) → 3 ✓

snap = 0 → { 0, 3, 0, 0, 5, 0, 0, 0 }

snap = 1 → { 0, 5, 0, 0, 5, 0, 0, 0 }

HashMap < snap_id, array > ✗ A lot of Space


int[] arr = { 0 1 2 3 4 5 6 7
0, 5, 0, 0, 5, 0, 0, 0 }
8 10
get(6,2)
0 3
1 5
get(1,1)
0 5
get(0,1)