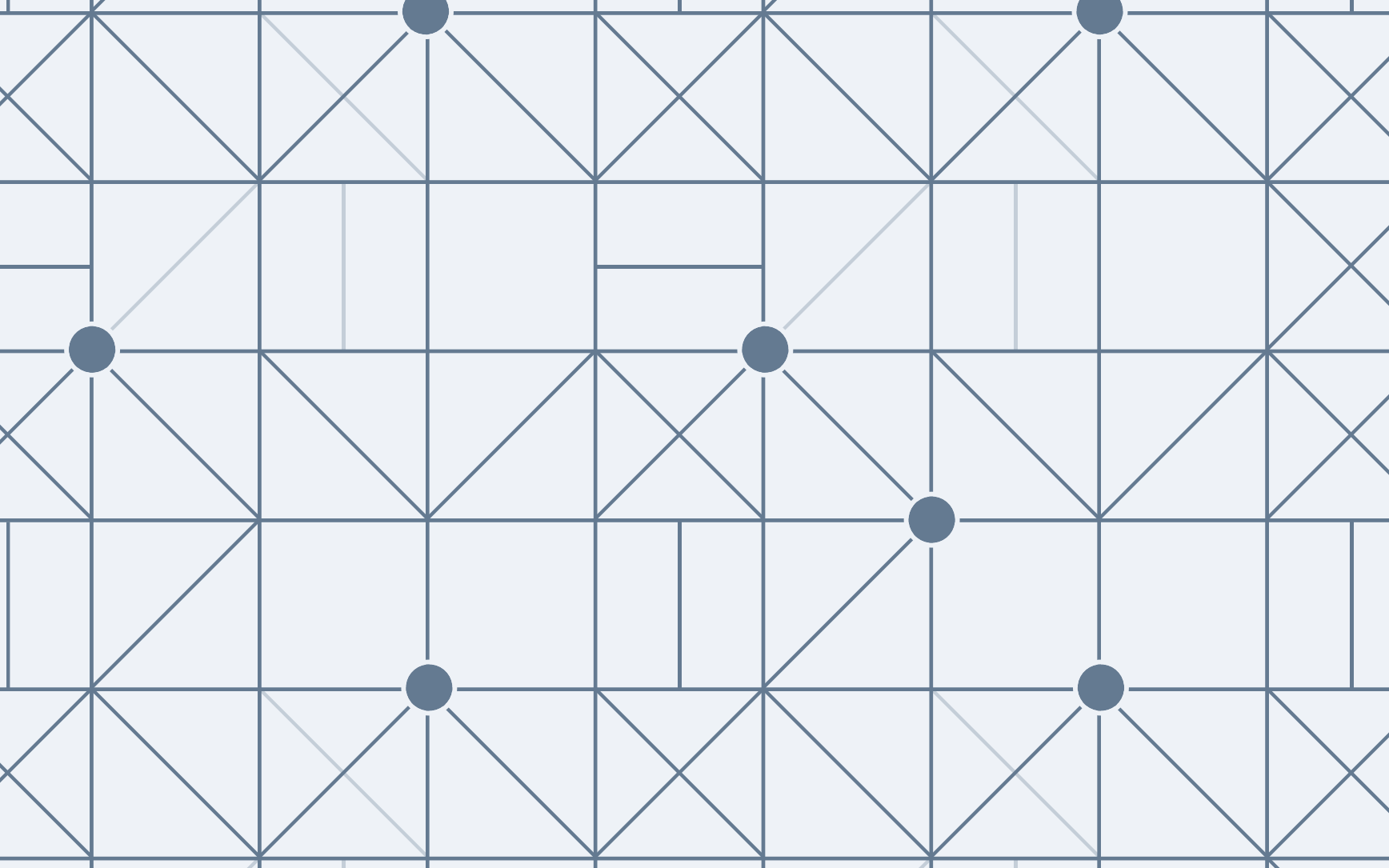


H.W

- Smallest substring with all characters. }
- Maximum ones after Modifications }

Agenda

- Longest subarray with equal freq. of 0's, 1's and 2's
- LRU Cache
- Snapshot Array

Longest Subarray with equal frequency of 0's, 1's and 2's.

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

freq 1 → 1
freq 2 → 1
freq 0 → 1

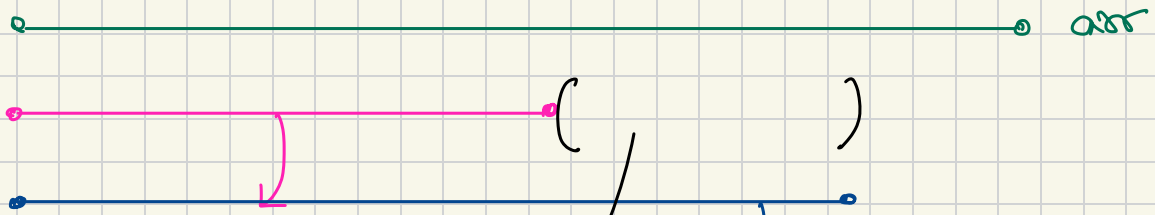
Brute force.

find all the subarrays, with equal freq 0's, 1's and 2's
store longest length, with equal freq. of all 3.

int cnt0
int cnt1
int cnt3

TC: O(N²) SC: O(1).

int[] arr = { }



0's → x
1's → y
2's → z
(prevState)

0's → x'
1's → y'
2's → z'
(currState)

0's → α
1's → α
2's → α

$x' = x + \alpha$
$y' = y + \alpha$
$z' = z + \alpha$

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

when this is satisfied, we found subarray with equal freq. of 0's, 1's and 2's

```
int[] arr = { 1, 1, 0, 2, 1, 0, 1, 2, 1, 2, 2, 0, 1 }
            0  1  2  3  4  5  6  7  8  9  10 11 12
```

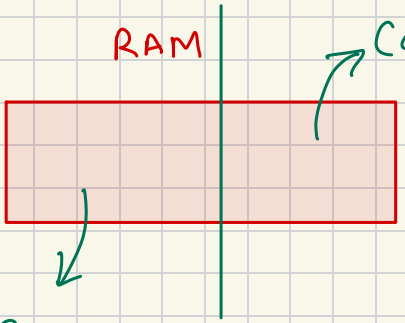
x	0	0	0	1	1	1	2	2	2
y	0	1	2	2	2	3	3	4	4
z	0	0	0	0	1	1	1	1	2
y-x	0	1	2	1	1	2	1	2	2
z-y	0	-1	-2	-2	-1	-2	-2	-3	-2
	0≠0	1≠-1	2≠-2						2≠-2

Longest subarray = ~~0~~ ~~1~~ 6

2 keys in a HashMap — String Key, int value

encoding = "y-z $ z-y"

LRU Cache

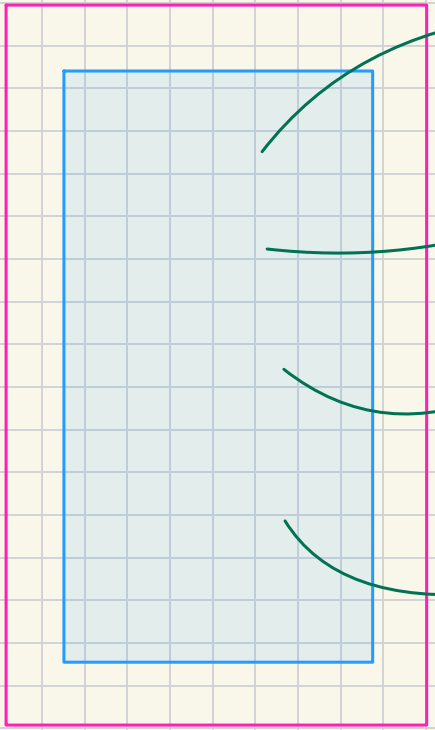


Cache Memory

{Apps running in background}

main Memory

{apps visible on screen}

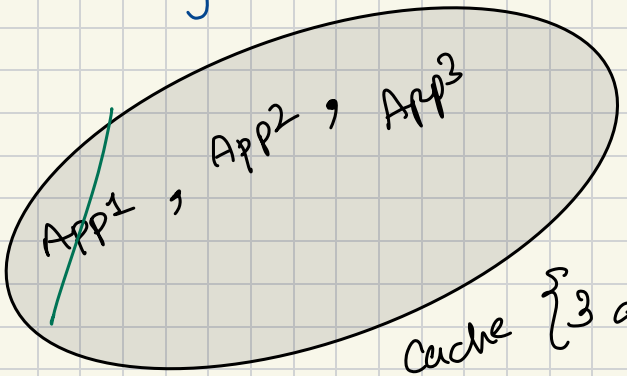


App 1 (minimized)

App 2 (minimized)

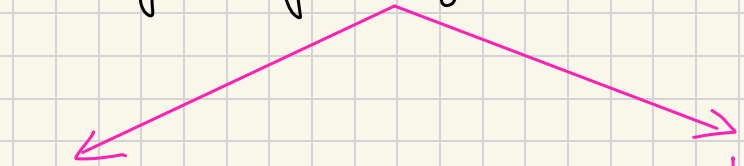
App 3 (minimized)

App 4 (minimized)



Cache {3 applications}

Memory Management System for Cache Memory.



LRU

(least recently used)

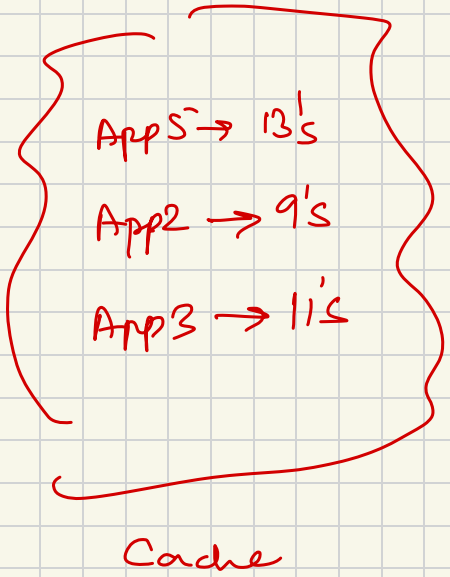
LFU

(least frequently used)

- opened ⟶ moving app to cache memory

0's App1 ⟶ opened
4's App2 ⟶ opened
5's App3 ⟶ opened
7's App4 ⟶ opened
9's App2 ⟶ pop up
11's App3 ⟶ reopened
13's App5 ⟶ opened

Limit = 3 app.



```
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.

return value against an app. Moves to Most Recently used pos.

add/update app to cache memory.

AppId

Brightness

LRU Cache (3)

set(1, 10)

set(3, 20)

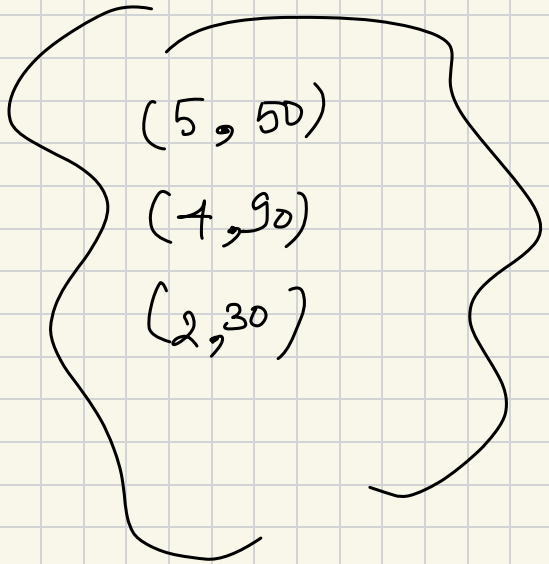
set(2, 40)

get(1) ⟶ 10

set(4, 90)

set(2, 30)

set(5, 50)



Cache

doubly ended linked list

LRU Cache (3)

set(1, 10) ✓

set(3, 20) ✓

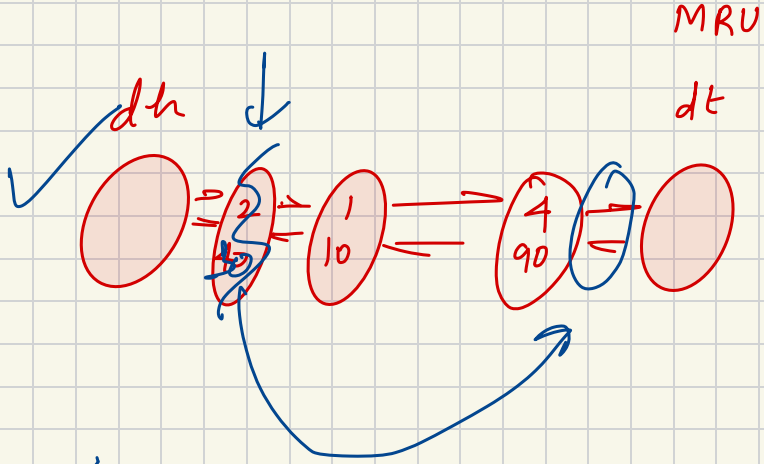
set(2, 40) ✓

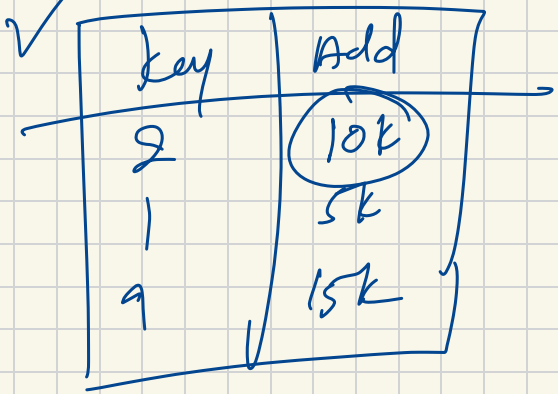
get(1) ⟶ 10 ✓

set(4, 90) ✓

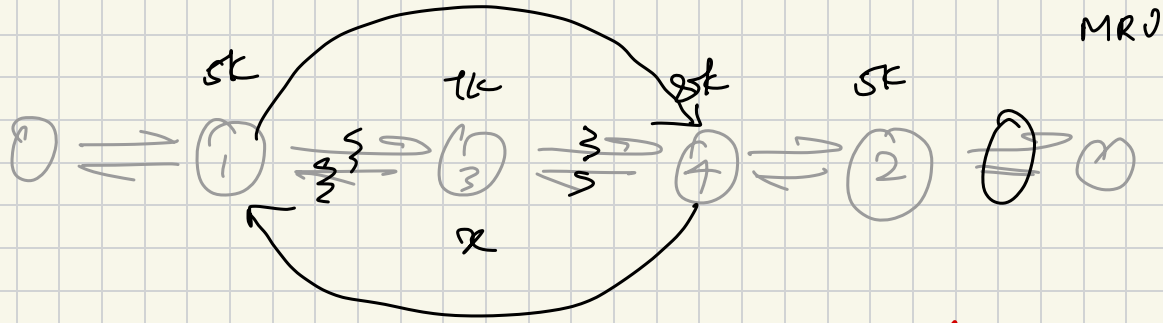
set(2, 30) ✓

set(5, 50)





MRU



Hash Map	
Key	add
2	6K
1	5K
3	7K
4	8K

removeNode (Node)

addLast ( )

Snapshot Array

int[] arr = {0, 0, 0, 0, 0, 0, 0}

10 → 70 → 60

30

20

set(1, 10)

set(2, 30)

snap()

set(1, 70)

set(5, 20)

snap()

set(1, 60)

get(1, 0) ⇝ 10

get(5, 0) ⇝ 0

Snap = 0 {0, 10, 30, 0, 0, 0, 0}

Snap = 1 {0, 70, 30, 0, 0, 20, 0}

HashMap<Snap_id, Array>

get(2, 10) ⇝ 30

int[] arr = {0, 0, 0, 0, 0, 0, 0}