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.

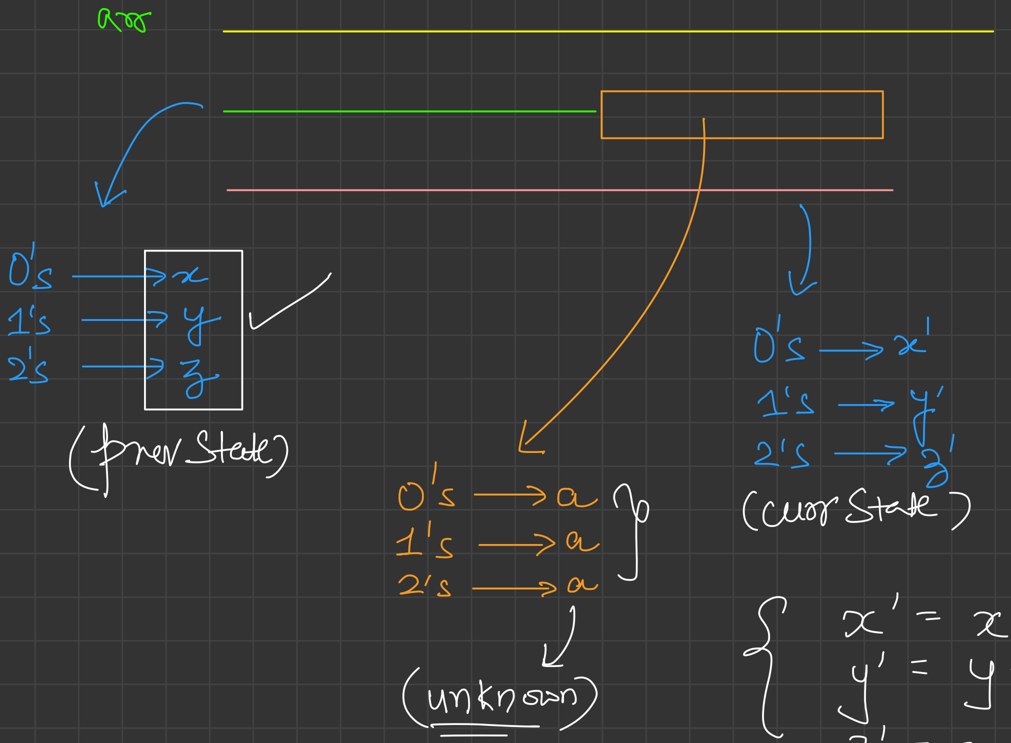
↳ find all subarray, with freq of 0's, 1's & 2's

Calc. longest having equal freq

$\boxed{TC : O(N^2) \quad SC : O(1)}$

O(N) ?

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



$$\begin{cases} x' = x + a \\ y' = y + a \\ z' = z + a \end{cases} \;\Big|\; \begin{cases} y' - x' = y - x \\ z' - y' = z - y \end{cases}$$

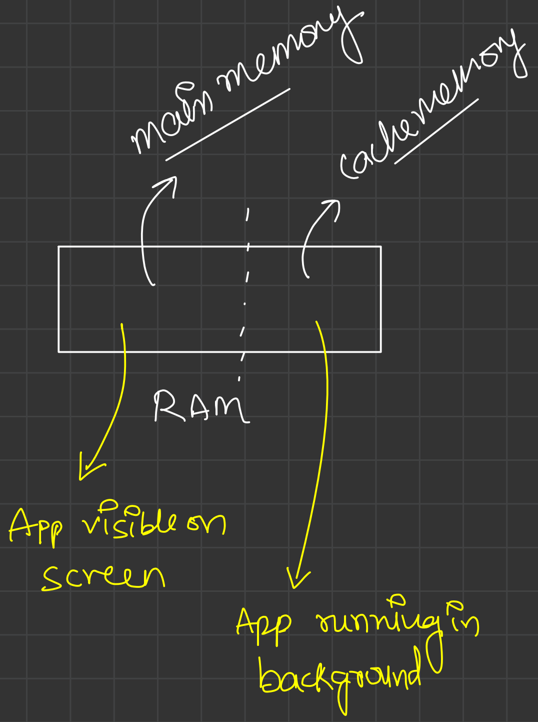
arr[] = {1, 1, 2, 0, 1, 0, 1, 2, 1, 2, 2, 0, 1}

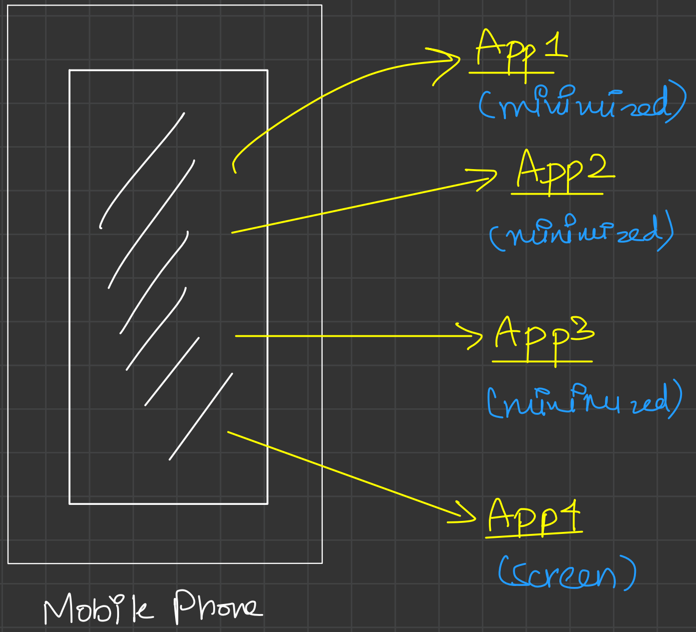
x	0	0	0	0	1	1	2	2	2	2	2	2	3
y	0	1	2	2	2	3	3	4	4	5	5	5	5
z	0	0	0	1	1	1	1	1	2	2	3	4	4
$y-x$	0	1	2	2	1	2	1	2	2	3	3	3	2
$z-y$	0	-1	-2	-1	-1	-2	-2	-3	-2	-3	-2	-1	-1

code (key)

$y-x\ ,\ z-y$ (string)

LRU Cache :





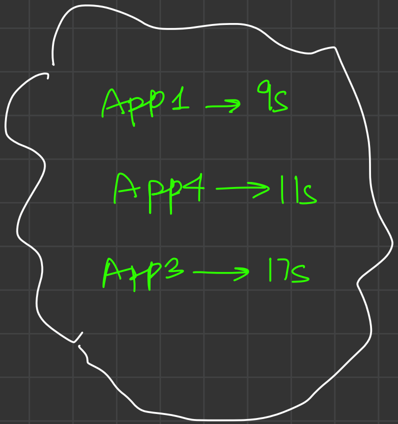
has to be managed !

$$\begin{tikzcd}
 & \text{Cache Memory Managment} \arrow[dl] \arrow[dr] & \\
\text{LRU (least Recently Used)} & & \text{LFU (least frequently used)}
\end{tikzcd}$$

opened ⟶ moving app to cache memory

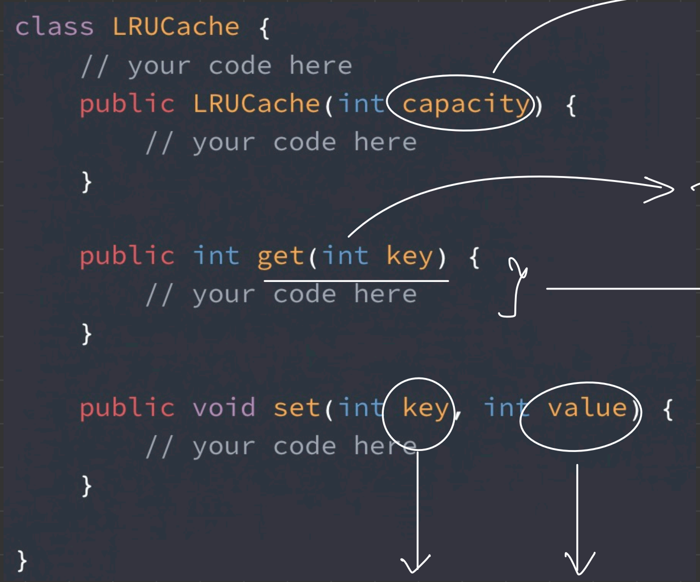
0's	App1 ⟶	opened
2's	App2 ⟶	opened
5's	App3 ⟶	opened
9s	App1 ⟶	reopened
11s	App4 ⟶	opened
17s	App 3 ⟶	output/popup

limit = 3 app



Cache

(LRU)



```java
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 op

→ move this app to most recently used pos.

} adds new application to cache memory or reopen or update a prev app. with a new value.

AppId

brightness

LRU Cache (3)

set (1, 10)

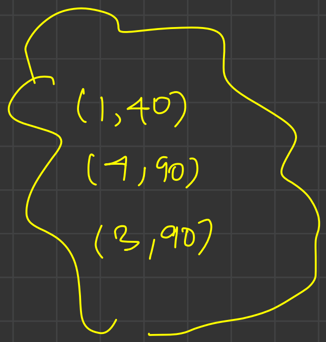
set (2, 20)

set (3, 90)

set (1, 40)

set (4, 90)

$\text{get}(3) \longrightarrow 90$



Cache
(limit = 3)

LRU Cache (3)

set (1, 10)

set (2, 20)

set (3, 90)

set (1, 40)

set (4, 90)

get (3) $\longrightarrow$ 90

LRU Cache (3)

set (1, 10) ✓

set (2, 20) ✓

set (3, 90) ✓

set (1, 40) ✓

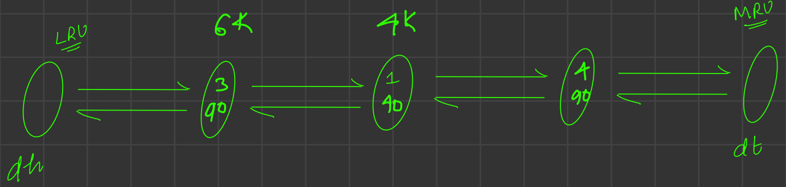
set (4, 90) ✓

get (3) $\longrightarrow$ 90

HashMap

Key $\longrightarrow$ AppId

Value $\longrightarrow$ address



✓ addLast() $\longrightarrow$ ①

✓ removeNode() $\longrightarrow$ ②

moveToLast()

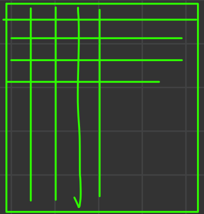
$\hookrightarrow$ ①

$\hookrightarrow$ ②

Snapshot Array

int [] arr = $\left\{ \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0, & 20, & 0, & 0, & 90, & 0, & -11 \end{array} \right\}$

{ 80%
{ 90%



set(1, 10)

set(4, 90)

✓ snap()

set(1, 5)

set(1, 20)

set(6, -11)

✓ snap()

get(1, 1) $\rightsquigarrow$ 20

get(1, 0) $\rightsquigarrow$ 10

snap = 0 $\left\{ \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0, & 10, & 0, & 0, & 90, & 0, & 0 \end{array} \right\}$

snap = 1 $\left\{ \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0, & 20, & 0, & 0, & 90, & 0, & -11 \end{array} \right\}$

key value

✓ HashMap< snap-id, array> ?

more memory!

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \text{int [] arr} = \{ & 0, & 10, & 10, & 0, & 90, & 0, & 0 \ \} \end{array}$$

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

set(1, 10)

set (4, 90)

snap()

get(3, 0) → 0

get(1, 0) → 10

set (2, 10)

snap()

get(1, 1)