longest Subarray with equal number of 0's, 1's and 2's.

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

Brute force origin

↳ find all subarrays; $\rightarrow O(N^2)$

✓ ↳ cnt no of 0's, 1's and 2's $\rightarrow O(N)$

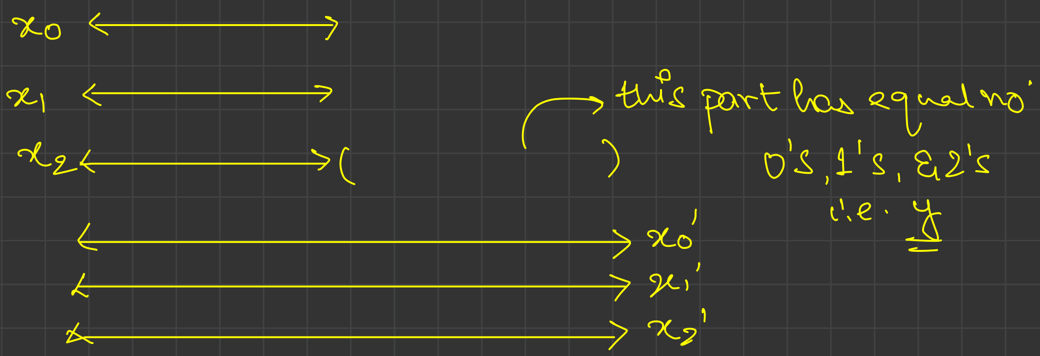
↳ store max len subarray.

$$\left\{ \begin{array}{l} TC: O(N^3) \\ SC: O(1) \end{array} \right.$$

→ while calc the subarray's

$$\left\{ \begin{array}{l} TC: O(N^2) \\ SC: O(1) \end{array} \right.$$

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



$$x_0' - x_0 = y \quad (1)$$

&

$$x_1' - x_1 = y \quad (2)$$

$$x_2' - x_2 = y \quad (3)$$

eq $\textcircled{2}$ = eq $\textcircled{1}$

$$x_1' - x_1 = x_0' - x_0 \rightarrow x_1' - x_0' = x_1 - x_0 \quad \text{———} \quad \textcircled{4}$$

eq $\textcircled{2}$ = eq $\textcircled{3}$

$$x_1' - x_1 = x_2' - x_2 \rightarrow x_2' - x_1' = x_2 - x_1 \quad \text{———} \quad \textcircled{5}$$

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

x_0	0	0	0	0	1	1	2	2	2	2	2	2	3	3
x_1	0	1	2	2	2	3	3	4	4	5	5	5	5	6
x_2	0	0	0	1	1	1	1	1	2	2	3	4	4	4
$x_1 - x_0$	0	1	2	2	1	2	1	2	2	3	3	3	2	3
$x_2 - x_1$	0	-1	-2	-1	-1	-2	-2	-3	-2	-3	-2	-1	-1	-2

0\$0 1\$-1 2\$-2

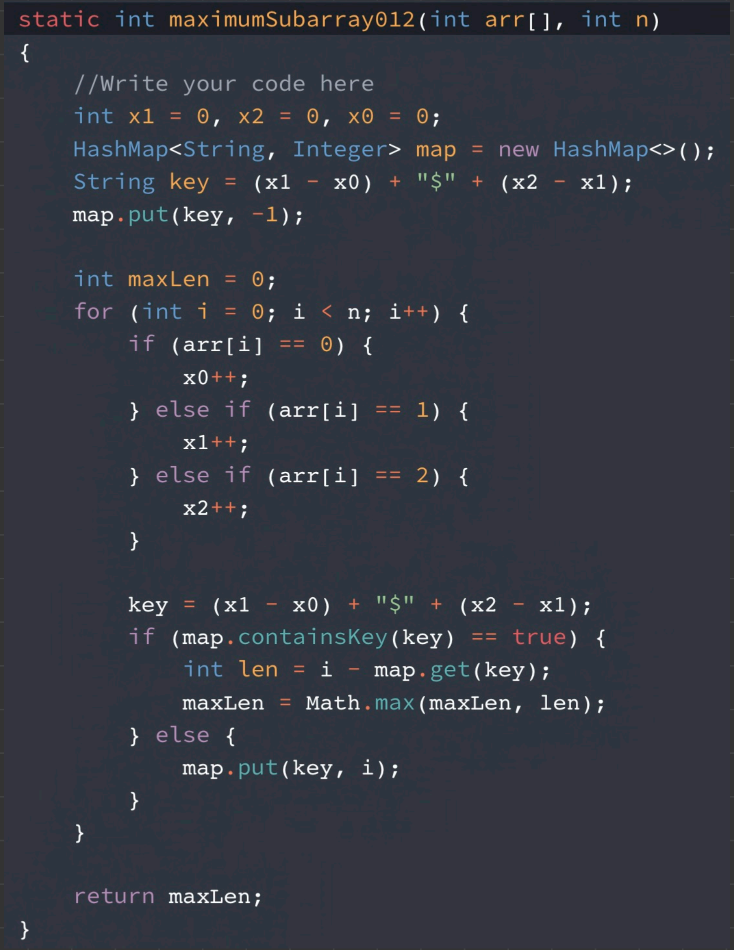
2\$-2

maxLen = ~~0~~ ~~3~~ 9

Key = $(x_1 - x_0)$ \$ $(x_2 - x_1)$

HashMap<String, Integer>

index



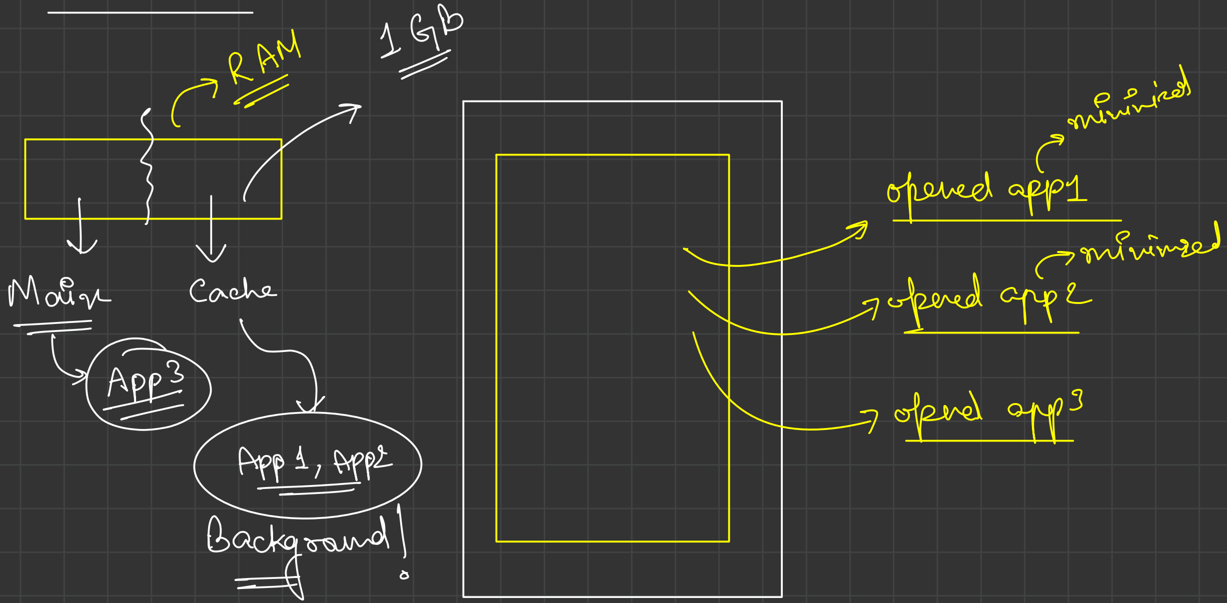
```
static int maximumSubarray012(int arr[], int n)
{
    //Write your code here
    int x1 = 0, x2 = 0, x0 = 0;
    HashMap<String, Integer> map = new HashMap<>();
    String key = (x1 - x0) + "$" + (x2 - x1);
    map.put(key, -1);

    int maxLen = 0;
    for (int i = 0; i < n; i++) {
        if (arr[i] == 0) {
            x0++;
        } else if (arr[i] == 1) {
            x1++;
        } else if (arr[i] == 2) {
            x2++;
        }

        key = (x1 - x0) + "$" + (x2 - x1);
        if (map.containsKey(key) == true) {
            int len = i - map.get(key);
            maxLen = Math.max(maxLen, len);
        } else {
            map.put(key, i);
        }
    }

    return maxLen;
}
```

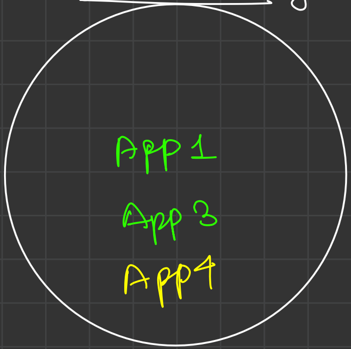
LRU Cache


RAM
1 GB
Main
Cache
App3
App 1, App2
Background!
opened app1
minimized
opened app2
minimized
opened app3

$$\begin{tikzcd}
 & \text{to free cache memory} \arrow[dl] \arrow[dr] & \\
\text{lru} \arrow[d] & & \text{lfu} \arrow[d] \\
\text{least recently used} & & \text{least frequently used}
\end{tikzcd}$$

LRU

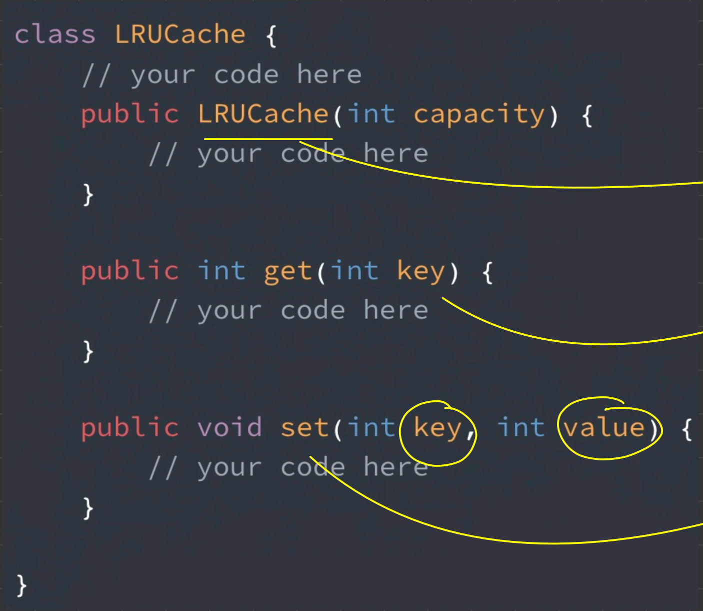
Cache memory



handle atmax = 3app

✓

t = 0's	App 1	(open)
t = 10's	App 2	(opened)
t = 15's	App 1	(opened)
t = 20's	App3	(opened)
t = 25s	App4	(opened)
t = 30's	App5	(opened)



```
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
    }

}
```

→ tells capacity of cache memory, i.e. number of app it can run.

→ put a app to most recently used place

→ Move app to Most Recently used

→ { open's a new app. , or reopens a open. app. }

↓

Move app to Most recently used

set(1, 10)

set(2, 10)

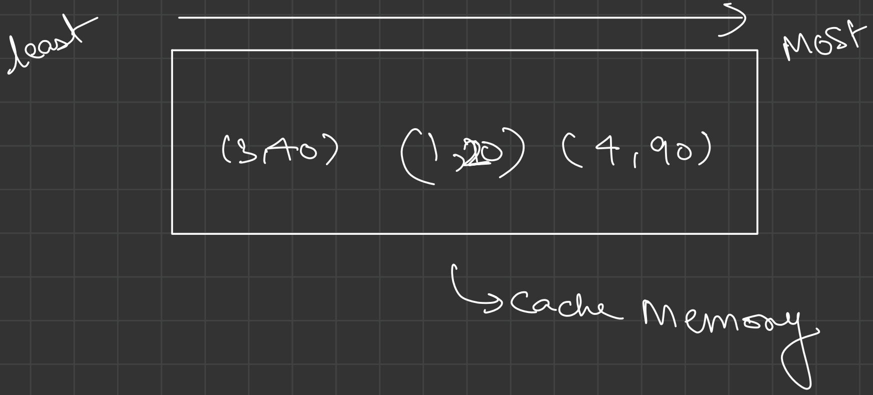
set(1, 20)

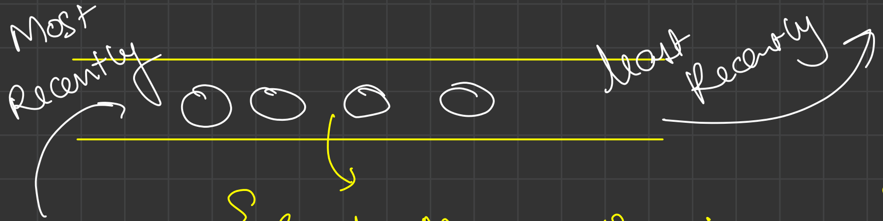
set(3, 40)

get(1)

set(4, 90)

set(1, 100)





{ Cache Memory is linear DS }

Queue X

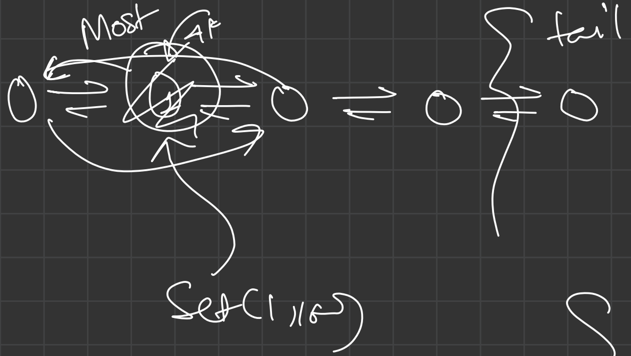
Stack X

Array Deque X

Array •

linked list •

doubly ended linkedlist $\rightarrow$ O(1)



HashMap

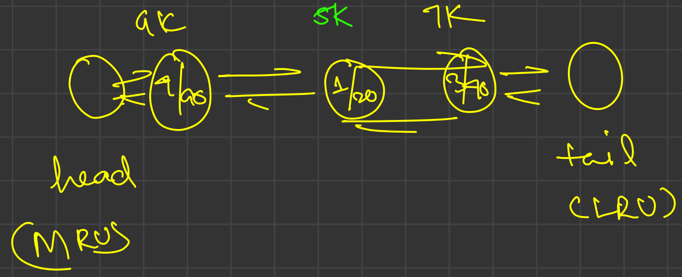
(key, Address)

(1, 4K)

find O(1)

1. set(1, 10)
2. set(2, 10)
3. set(1, 20) → movetofront
4. set(3, 40)
5. get(1)
6. set(4, 90)

set(1, 100)



1	5K
4	9K
3	7K

✓ addfront()

✓ removeNode()

Break till 9:50 pm

Dry Run!

Maximum Path Sum $\longrightarrow$ {route any two nodes of a tree}