Hashing day 4

- Minimum window substring
- Longest subarray with equal freq 0's, 1's and 2's
- LRU Cache
- Snapshot Array

Minimum window substring :

str 1 : a d b a c b b a c a f d a

str2 : c b a b d

{ find the smallest substring of str1 , containing each char atleast of same freq of str2

Brute force

- generate all the substring of str 1
- try to find atleast freq of char in str 2 in the substring

TC : $O(N^2 \times M)$ SC : $O(N)$

str2 : c b a b d

freq Map

c	$\longrightarrow$	1
b	$\longrightarrow$	2
d	$\longrightarrow$	1
a	$\longrightarrow$	1

ans = "dbacb"

TC : O(M+N)

SC : O(1)

inc

str1 : a d b a c b b a c a f d a

exc

freq Map

d	$\longrightarrow$	1
a	$\longrightarrow$	~~1~~ ~~2~~ 3
b	$\longrightarrow$	1
c	$\longrightarrow$	1
f	$\longrightarrow$	1

str1.substr(exc+1, inc+1)

str2 : c b a b d

freq Map

c $\longrightarrow$	1
b $\longrightarrow$	2
d $\longrightarrow$	1
a $\longrightarrow$	1

dmcnt = 5

mcnt = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

ans = "dbacb"

$\overset{0}{mc}$

$\downarrow$

str1 : a d b a c b b a c a f d a

$\uparrow$

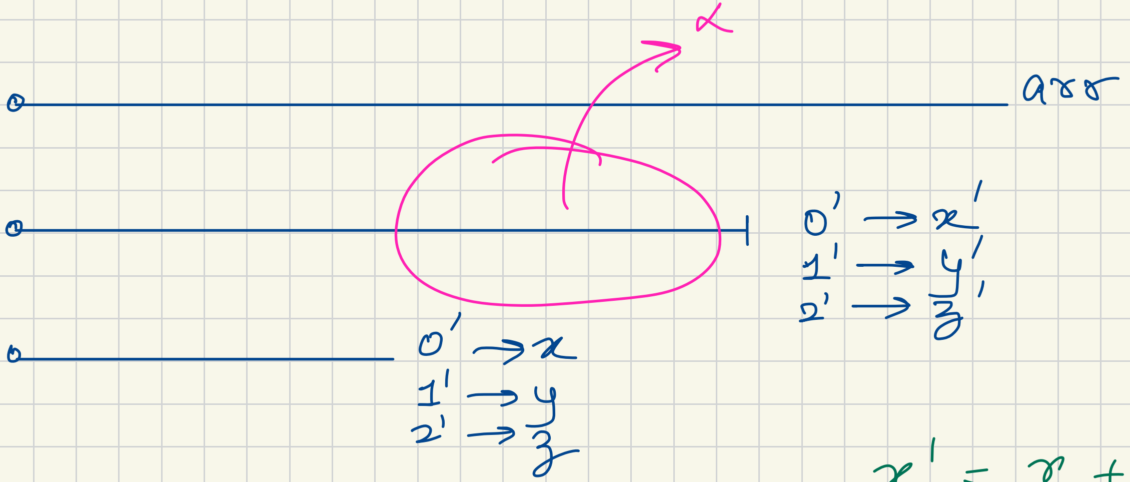
acq

freq Map

a $\longrightarrow$	1
b $\longrightarrow$	~~1~~ 2
c $\longrightarrow$	1

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

$$\begin{array}{ccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \text{int[] arr} = \{ & 0, & 2, & 0, & 1, & 1, & 0, & 2, & 1, & 2 \} \end{array}$$



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

$y' - x' = y - x$ $\qquad\qquad$ $z' - y' = z - y$

unknown!

equal freq of 0's, 1's, 2's

α
α
α

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

current freq of 0's, 1's, 2's

x
y
z

prev freq of 0's, 1's, 2's

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

Pattern

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

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

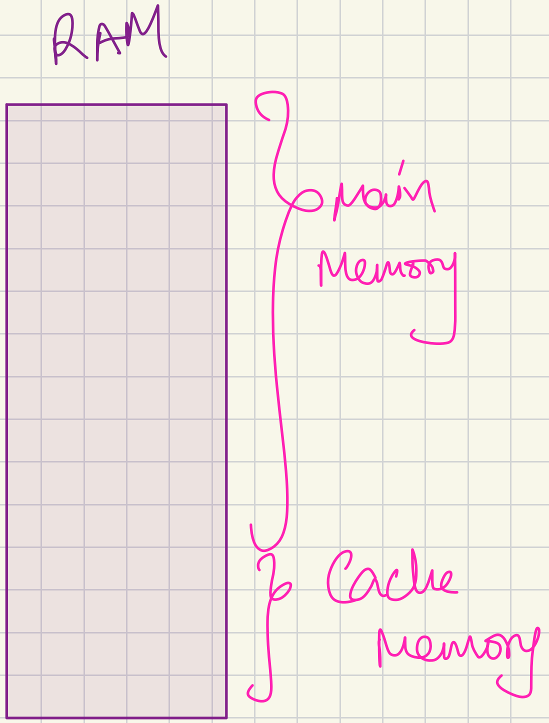
String

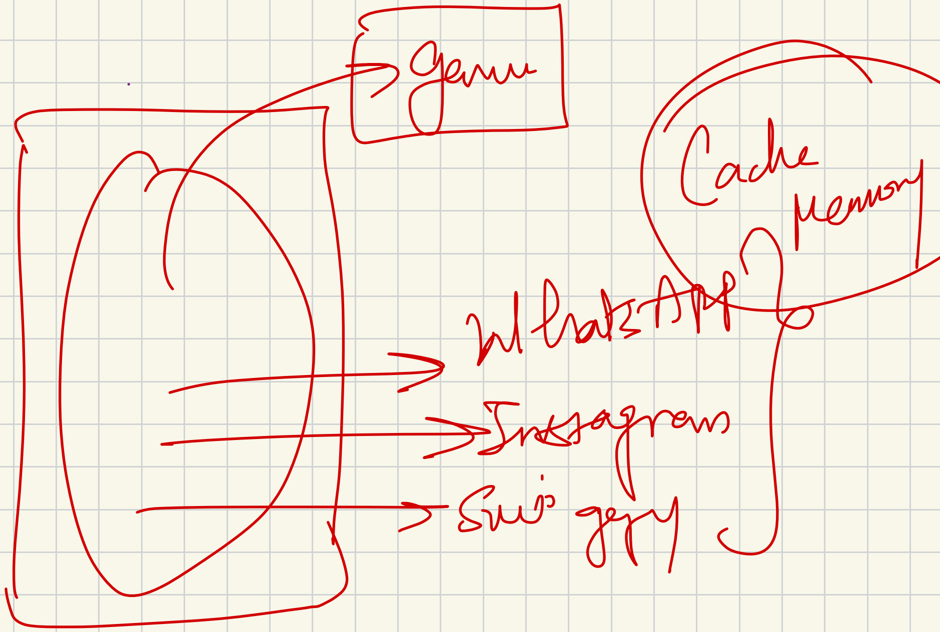
$(y-x)\$(z-y)$

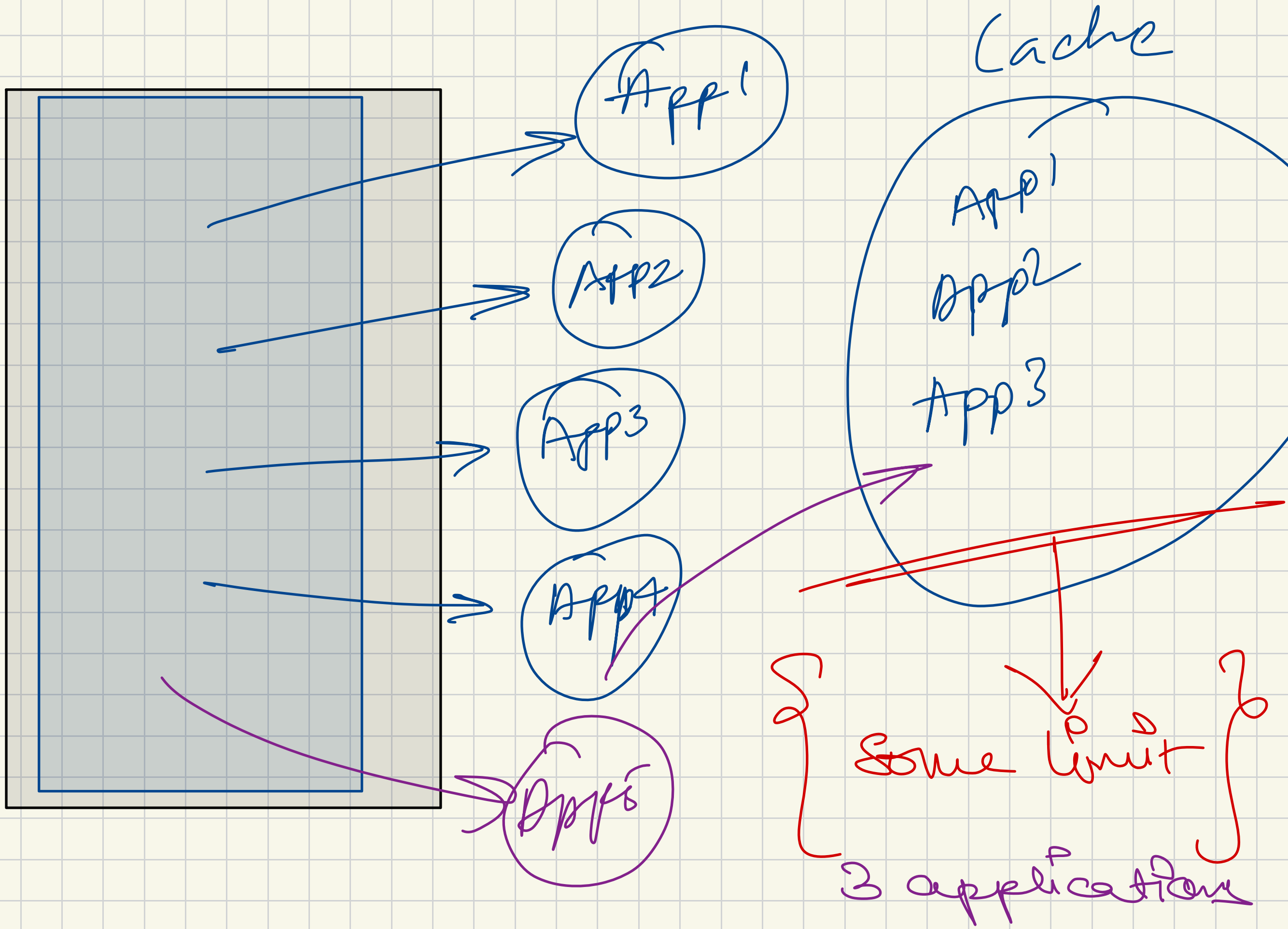
$\boxed{\{\text{keys}}$ map 0\$0 -1\$0

```
               0  1  2  3  4  5  6  7  8
int[] arr = {  0, 2, 0, 1, 1, 0, 2, 1, 2 };
```

LRU Cache

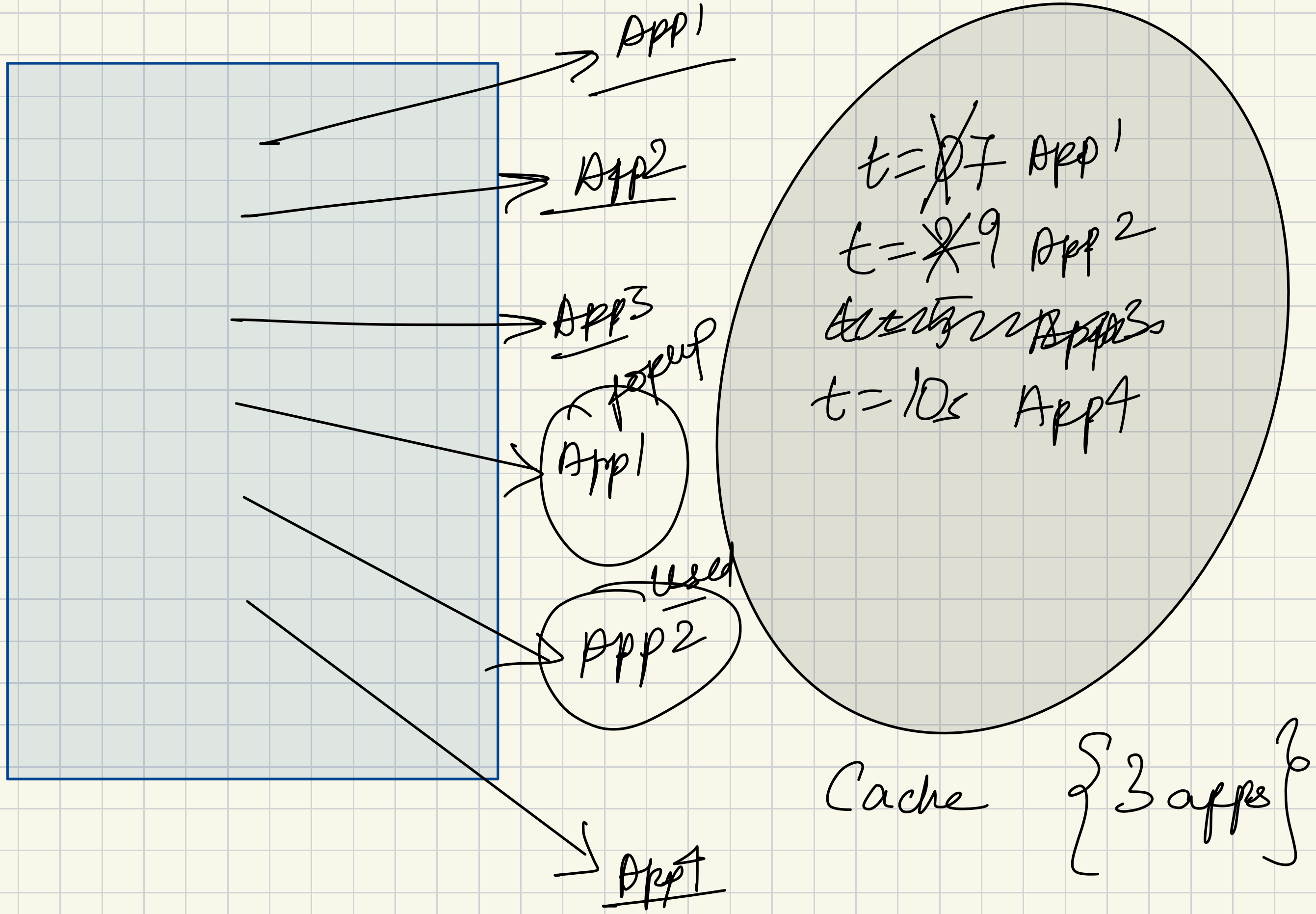

RAM
main Memory
Cache Memory

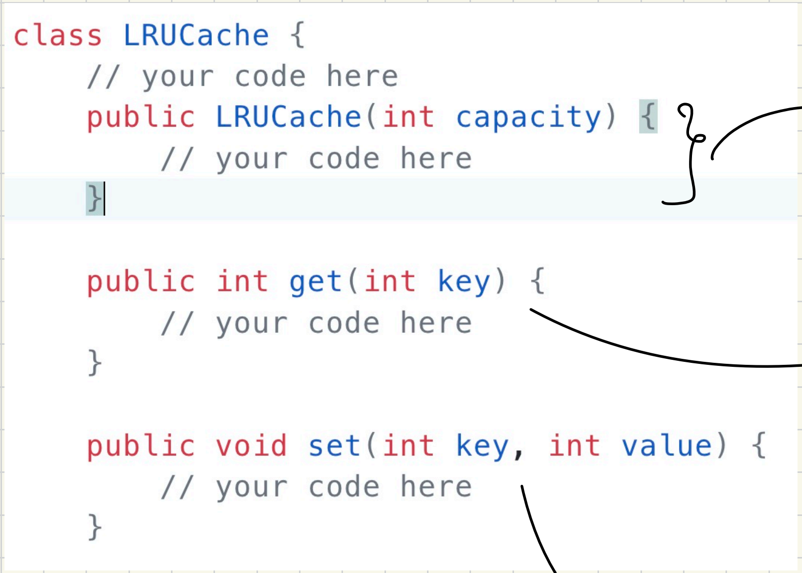

Gemu
Cache Memory
WhatsApp
Instagram
Swiggy


Cache
App1
App2
App3
App4
App5
App1
App2
App3
Some limit
3 applications

Cache Memory Managment System

$$\begin{tikzcd}
 & \text{Cache Memory Managment System} \arrow[dl] \arrow[dr] & \\
\text{Least recently used (LRU)} & & \text{Least frequently used (LFU)}
\end{tikzcd}$$


App1
App2
App3
popup
App1
used
App2
App4
t=~~0~~7 App1
t=~~8~~9 App2
~~t=5 App3~~
t=10s App4
Cache {3 apps}



```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
    }
```

tells the max capacity of cache memory

move the key (App) to most recently used pos

opens/update an existing app.

LRUCache (4)

set(1, 10)

set(2, 30)

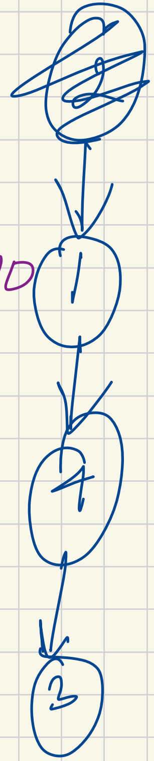
get(1) ⇝ 10

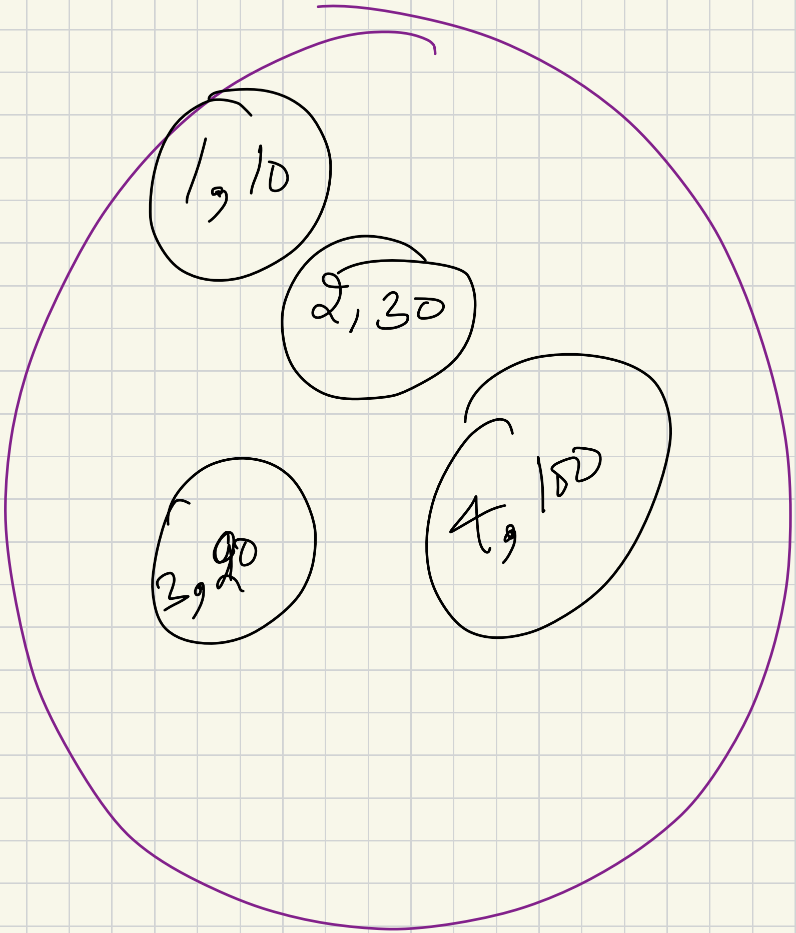
set(3, 90)

set(4, 100)

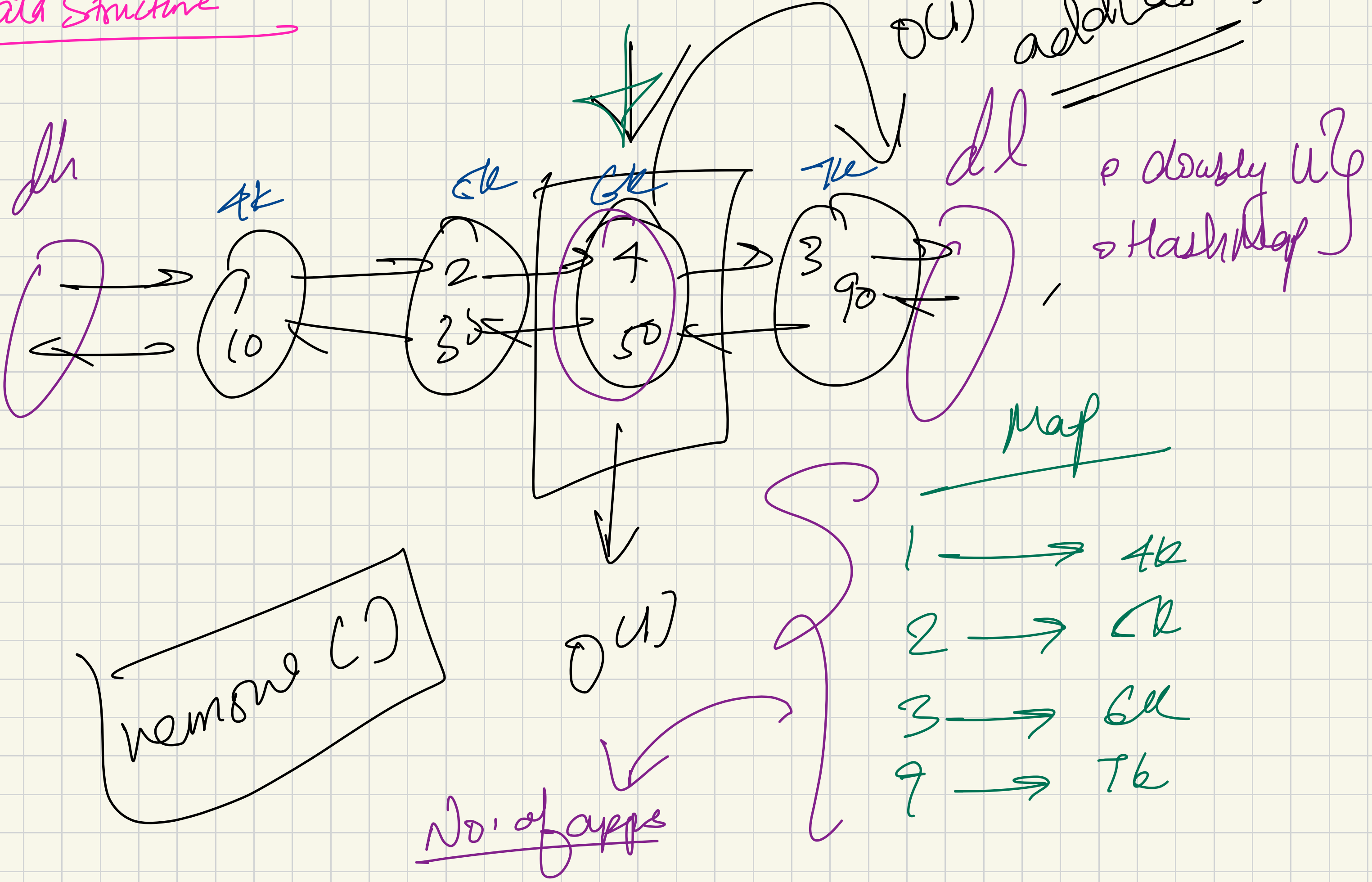
set(3, 20)

set(5, 100)





Data Structure



dll
4k
6k
6k
7k
10
2
35
4
50
3
90
O(1)
addLast()
dl
• doubly ll
• HashMap
remove()
O(1)
No: of apps
Map
1 → 4k
2 → 6k
3 → 6k
9 → 7k