C++

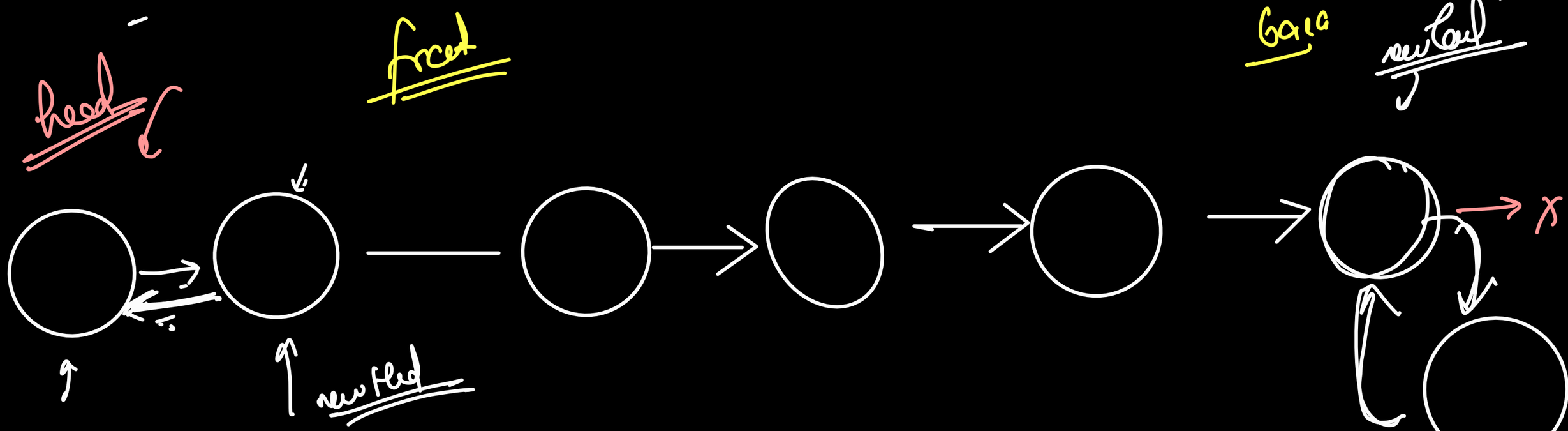
Stacks, vectors ...

Deque → sliding window

↳ it's a type of a queue

[double ended queue]

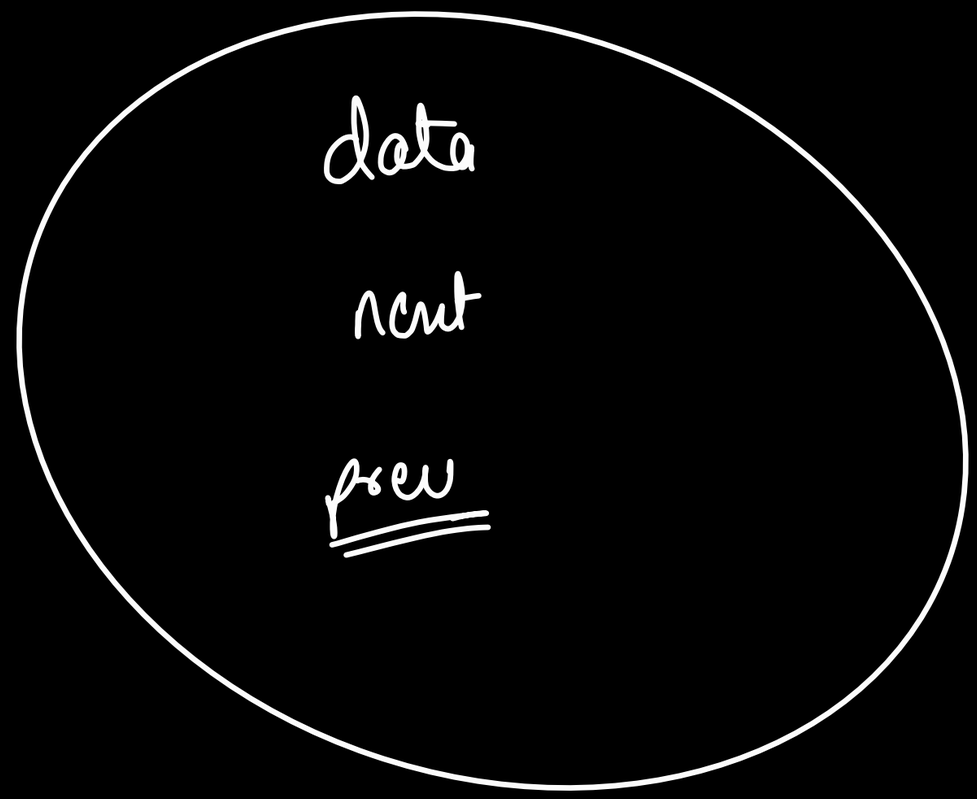
Doubly Linked List to implement deque



addAtHead → addAtFront
removeAtHead → removeAtFront
addAtTail → addAtBack
removeAtTail → removeAtBack

→ O(1)

Take 5min → implement a deque yourself



Back for

[1, 3, -1, -3, 5, 3, 6, 7]

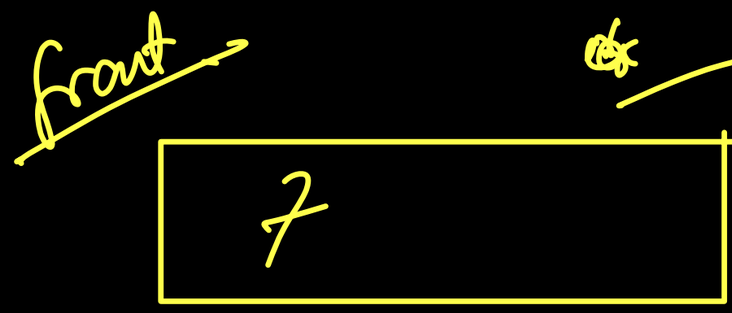
k = 3

O(n)

→ when the window moves right we add an elem
→ when the window moves right one elem must be discard from left

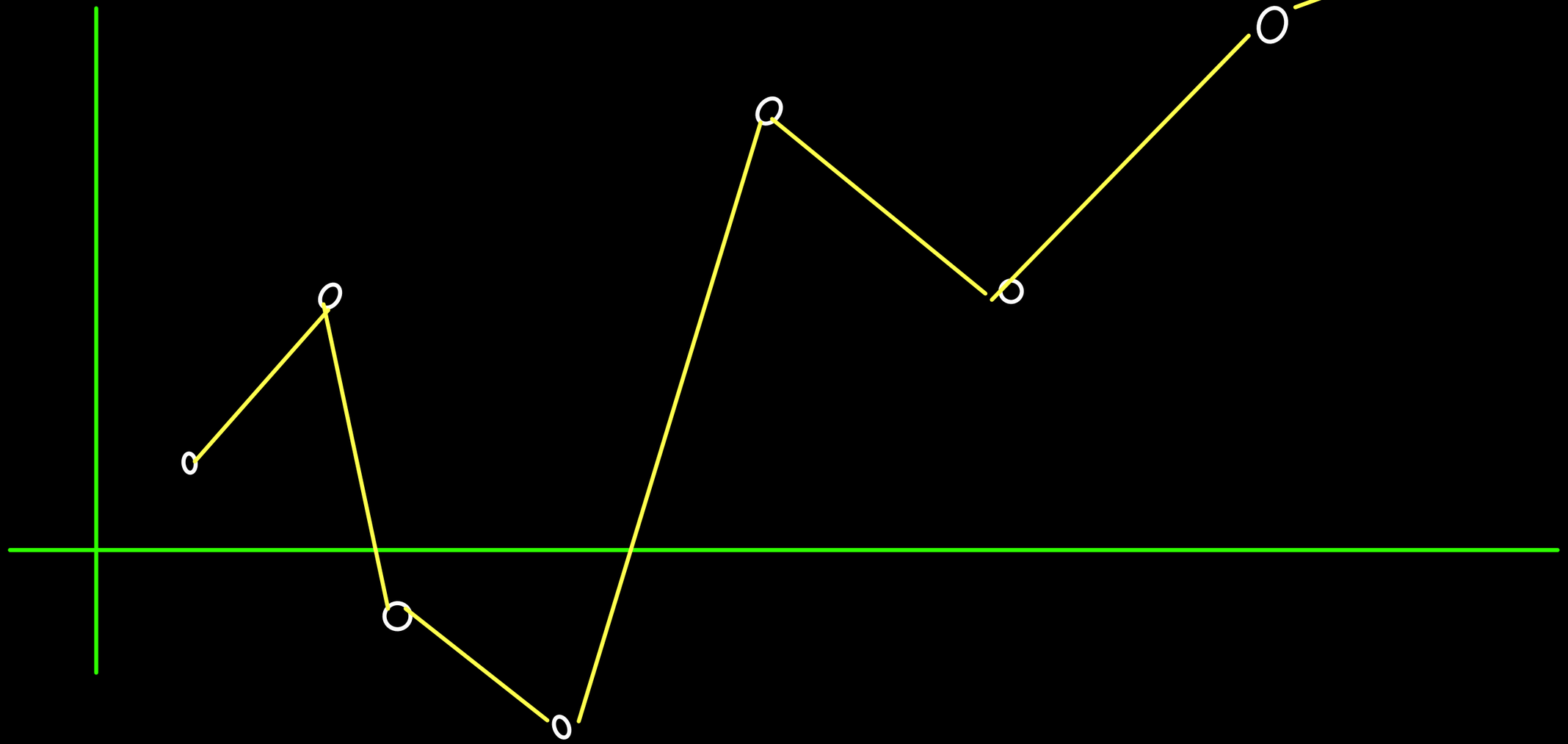
3 → 5, -5, 6, 7
3

O(n) Space O(1) O(k)



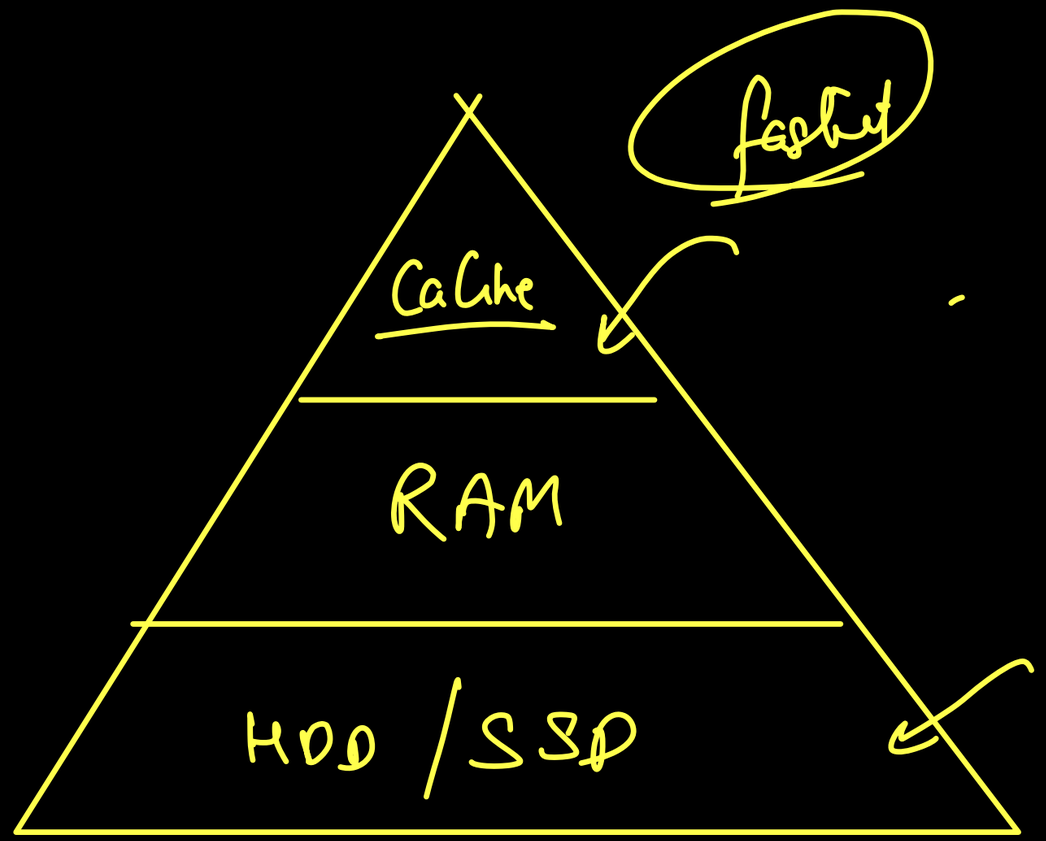
0 1

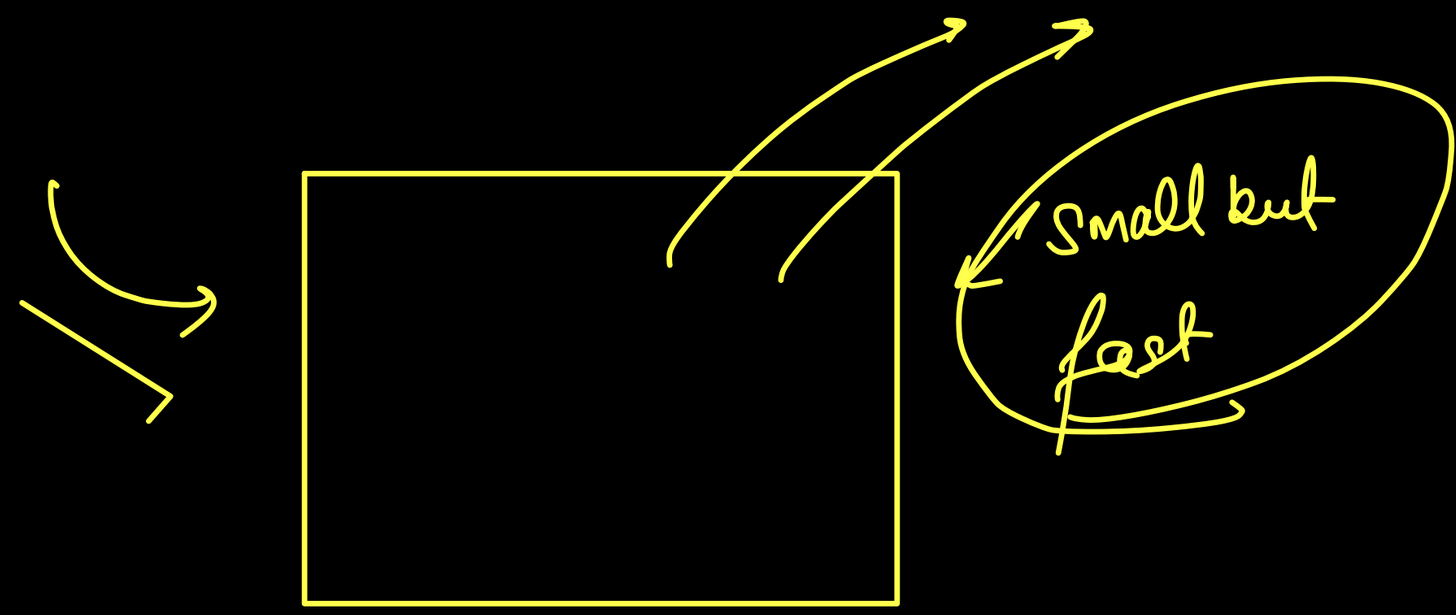
1, 3, -1, -3, 5, 3, 6, 7
0 1 2 3 4 5 6 7



5 4 3 2 1

$k = 3$



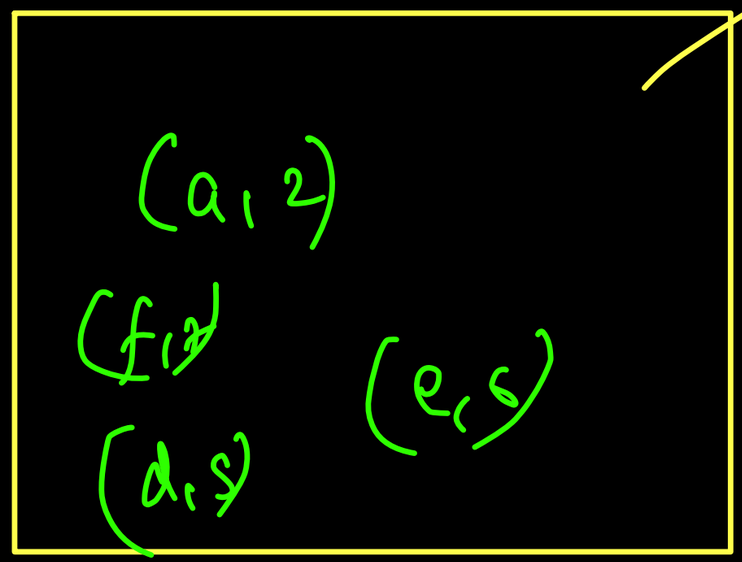


eviction
strategy

LRU

LFU

LRU



k = 4

get

put

put (a, 2)
put (b, 2)
put (c, 3)
get (b) → 2
get (a) → 2

put (f, 1)

put (d, 5)
put (e, 6)

{

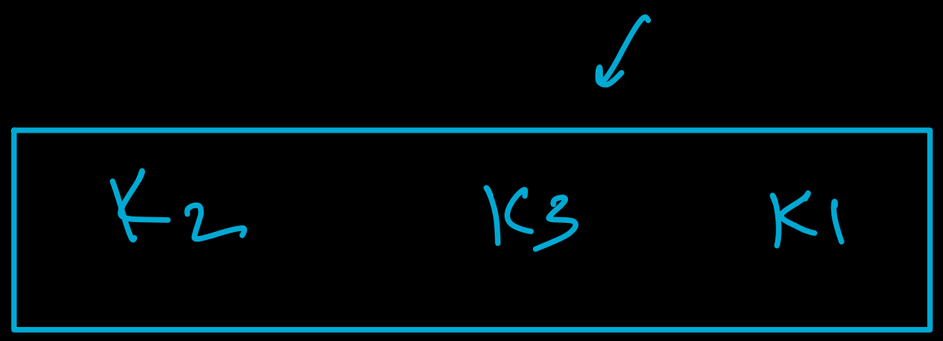
key - value

}

→ We need an efficient to check if a key exist or not.

↳ Map → { key-value }

→ when capacity exceeds we have to remove elements



put(K1)
put(K2)
put(K3)
get(K2)

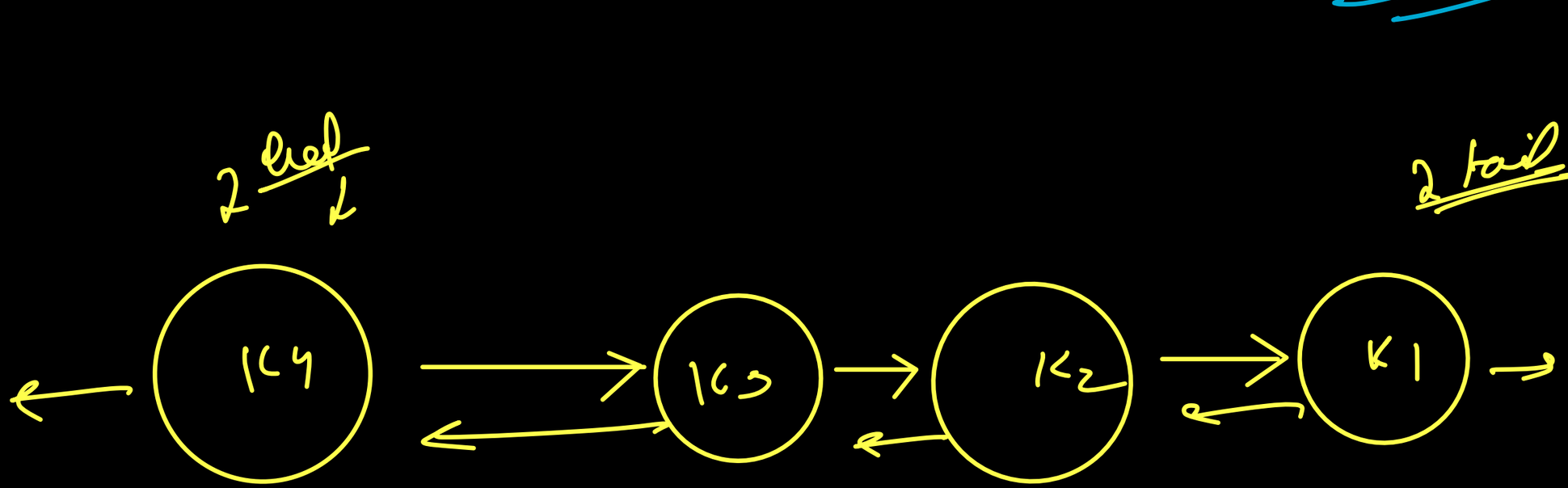
removal in btw

removed at last

added at start

Start → Recently used

last → Least Recently used



put (k1, 2)

put (k2, 3)

put (k4, 4)

put (k3, 5)

get (k2)

C

k1 : 2

k2 : 3

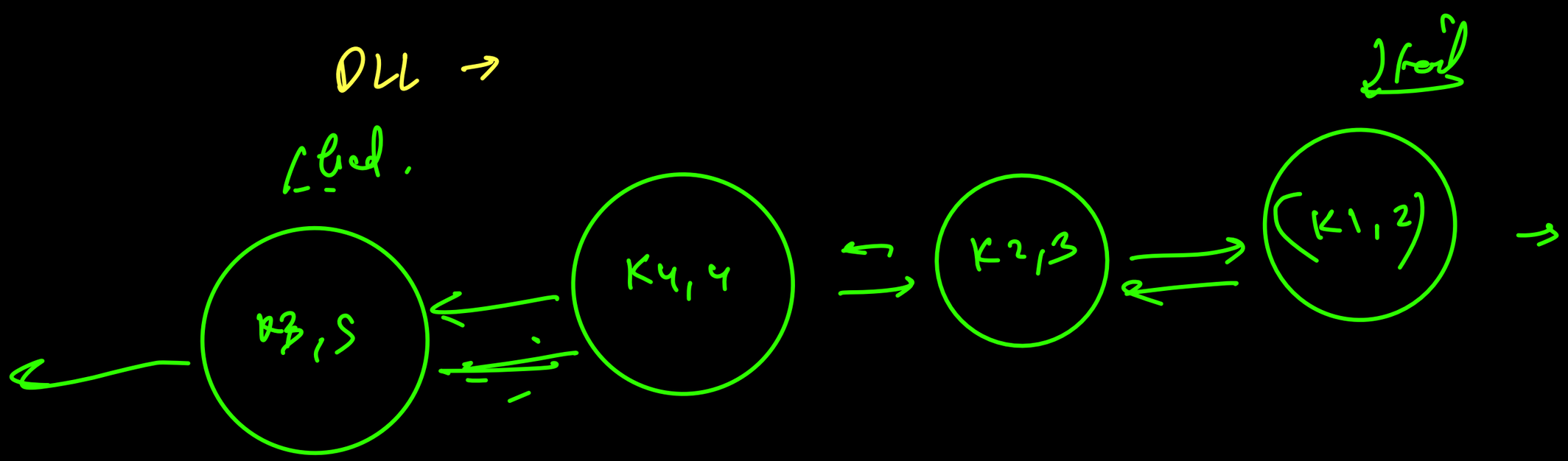
k4 : 4

k5 : 5

}

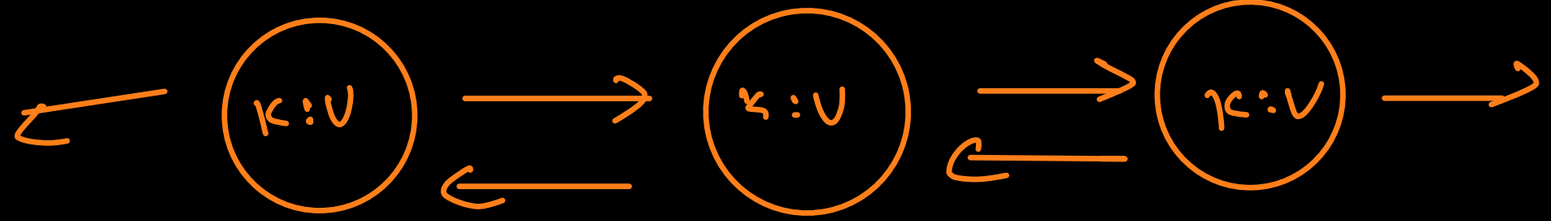
DLL →

Head.



Map → { key : addr of node }

DLL →



put (K1, 2) ✓
put (K2, 3)
put (K3, 4)
put (K4, 5)
get (K2) ✓
put (9, 5)

{
K1 : 4K
K2 : 2K
K3 : 10K
K4 : 25K
}

DLL
+
map (K : addr)

O(1)

} Head