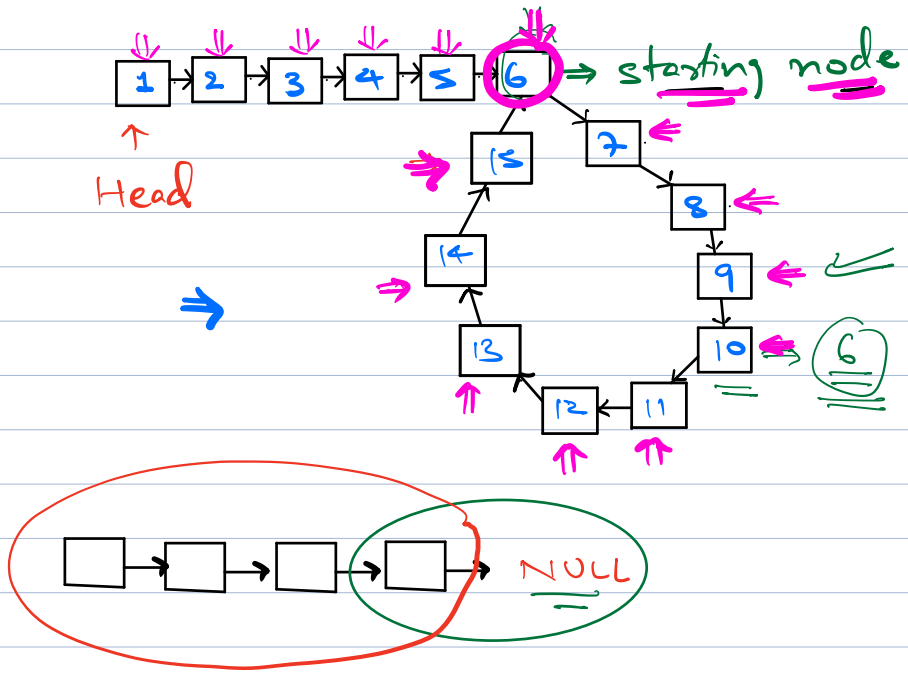
Q. Given a linked list
Return true if a cycle is present in the linked list.



1) Check if a cycle is present

2) Find the first node / start of the cycle.

Sol~

1) If you ever visit any node twice that means the LL contains a cycle.

⇒ Hashset of visited nodes.

Hashset < Node >

⇓

reference

(Iterate over the LL. O(N)

for every node :

O(1) { check if node is present (first node of the cycle)
↳ return True;
else
↳ return Node
add the node to the set.

)

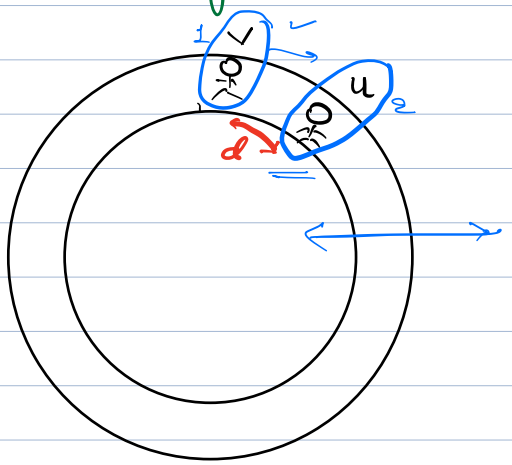
return false;

T.C. = O(N)

S.C. = O(N)

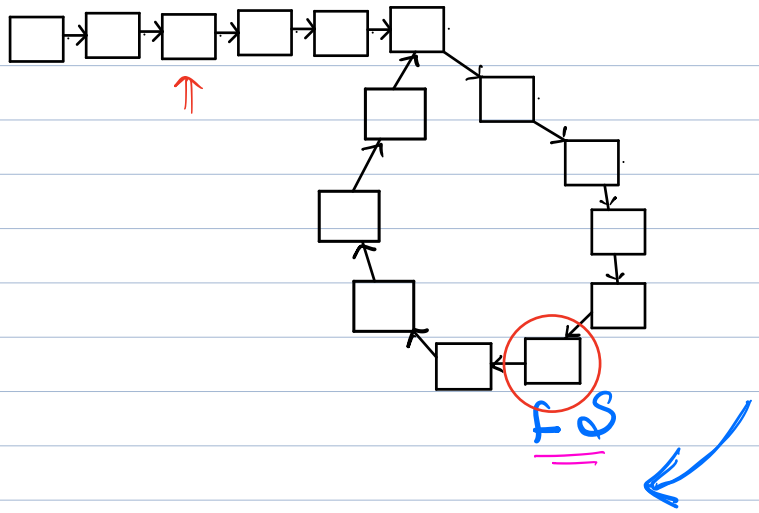
⇒ No extra space is allowed

2) Slow & fast pointer



if (v > u)

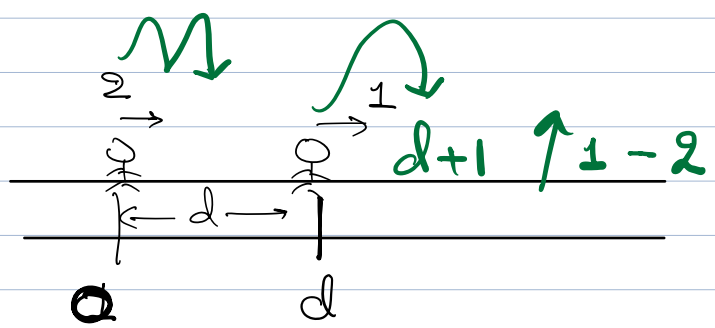
S →
f →
Compare



Code

```
fast = head
slow = head

while ( fast != NULL && fast.next != NULL) {
    slow = slow.next;
    fast = fast.next.next;
    if (slow == fast) {
        return True;
    }
}
```

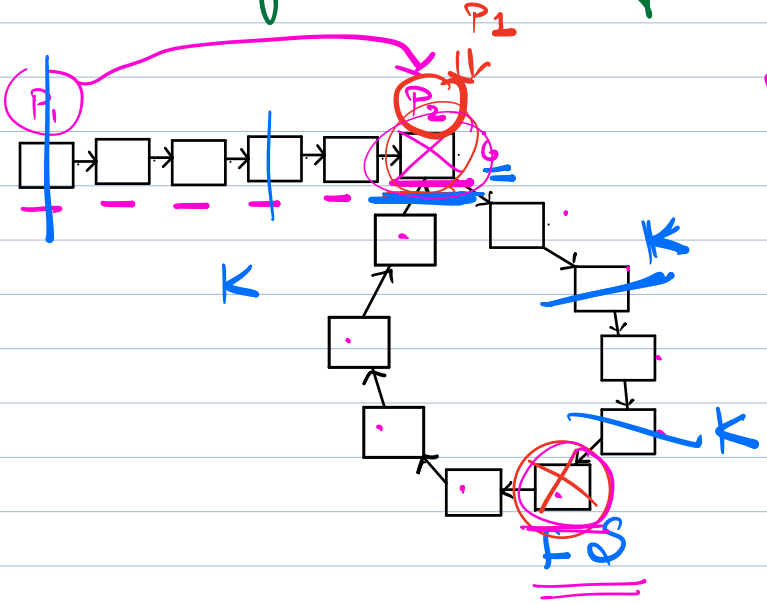


1 iteration ⇒ d ⇒ d-1

$10 \rightarrow 9 \rightarrow 8 \rightarrow 7 \ldots\ldots$ 0

2) Find the first element of the cycle.

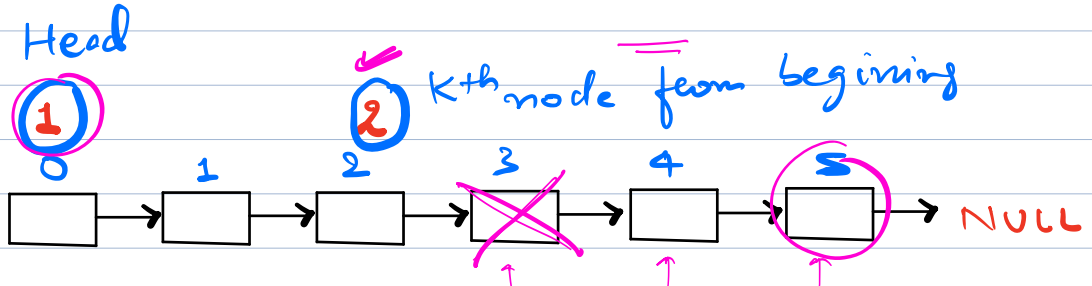
1)



P1 P2

$P_2 = P1 + K$

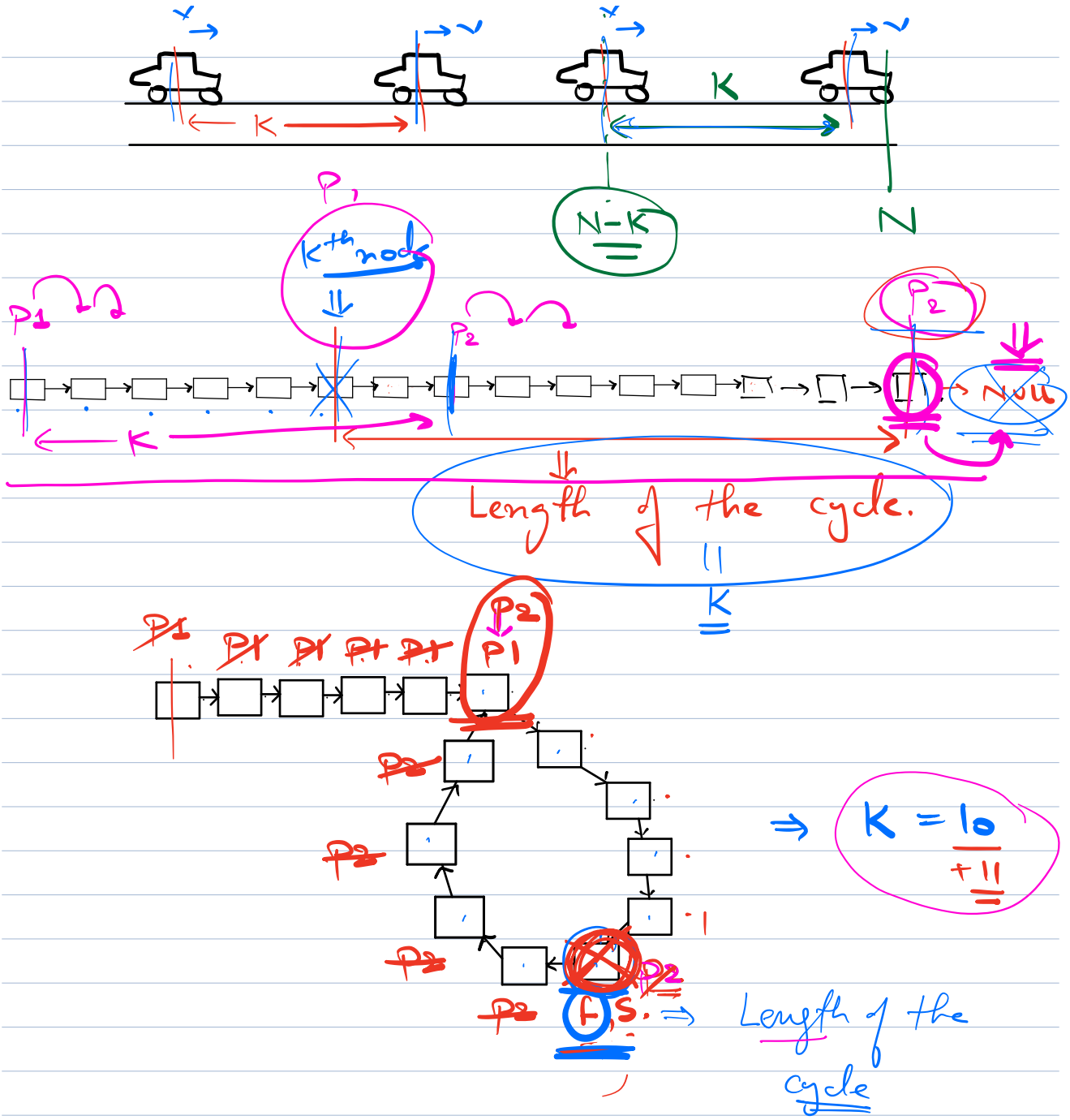
Length − the cycle



(Kth node from the end)

3rd node from the end.

Length — (length − 3rd node)



Steps

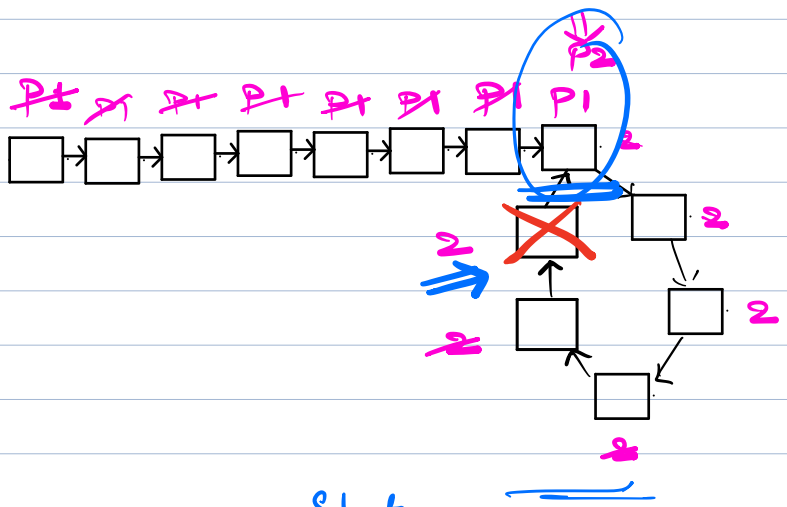
1) Using fast & slow pointer check if cycle is present

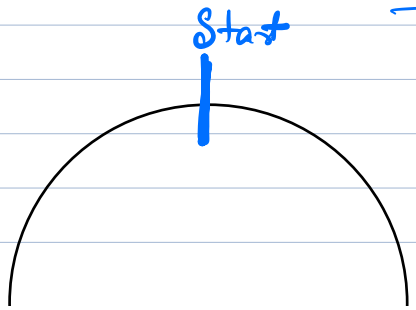
2) Keep the fast pointer fixed, & move the slow pointer one by one to find the length of the cycle.

3) P1 = Head
P2 = (K+1)th node from head. > run at the same pace, one node at a time

Iterate and find the point where P1 & P2 meets.

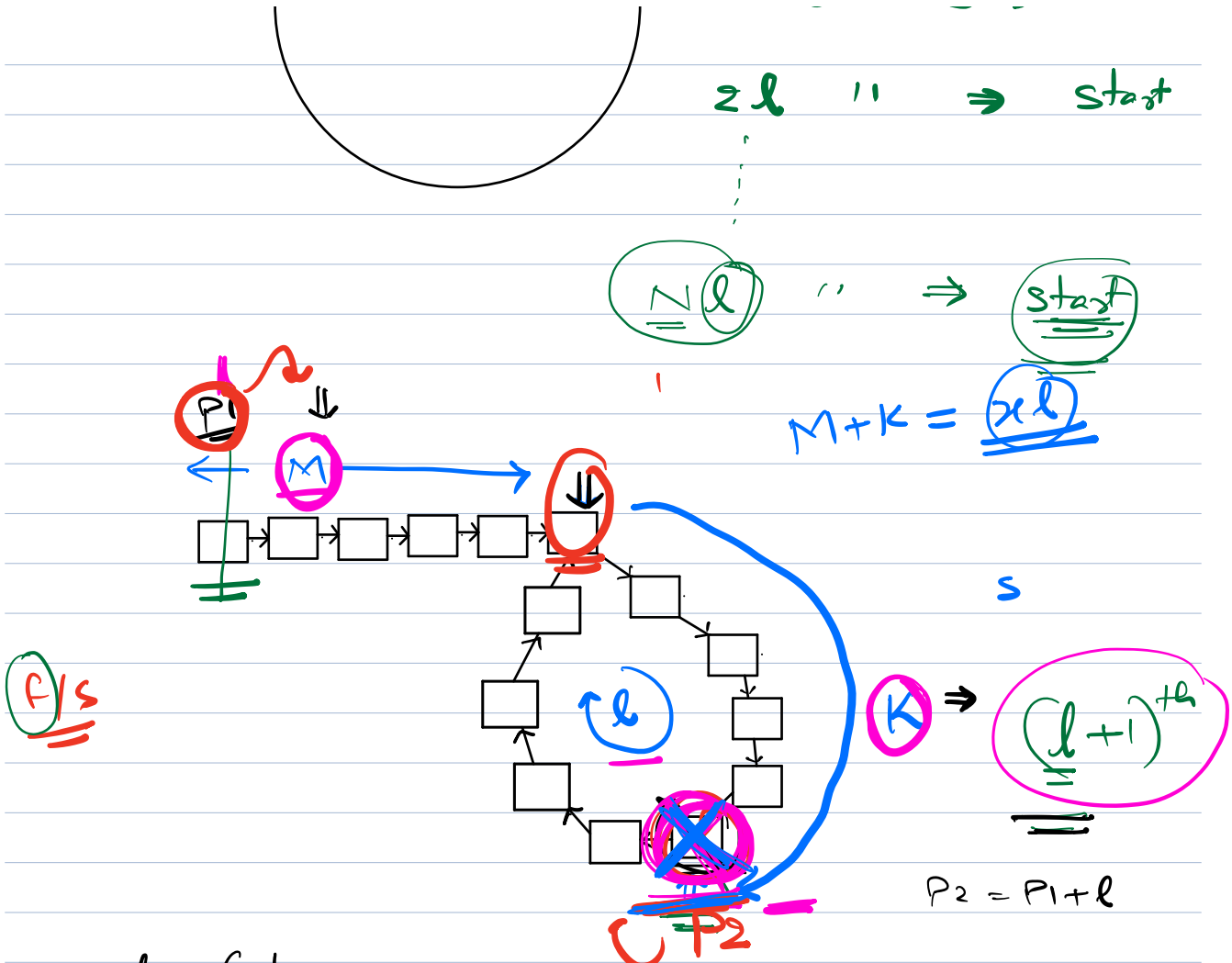
That is the first element of the cycle.





Length = l

l iterations → start



$$dist(fast) = M + xl + K$$

$$dist(slow) = M + yl + K$$

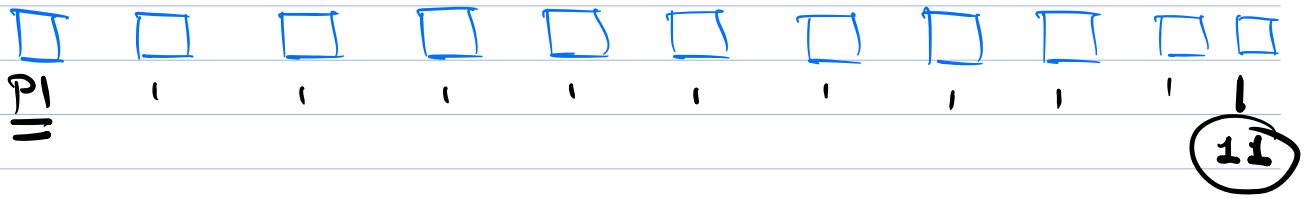
$$dist(fast) = 2(dist(slow))$$

$$M + xl + K = 2(M + yl + K)$$

$$xl - 2yl = 2M + 2K - M - K$$

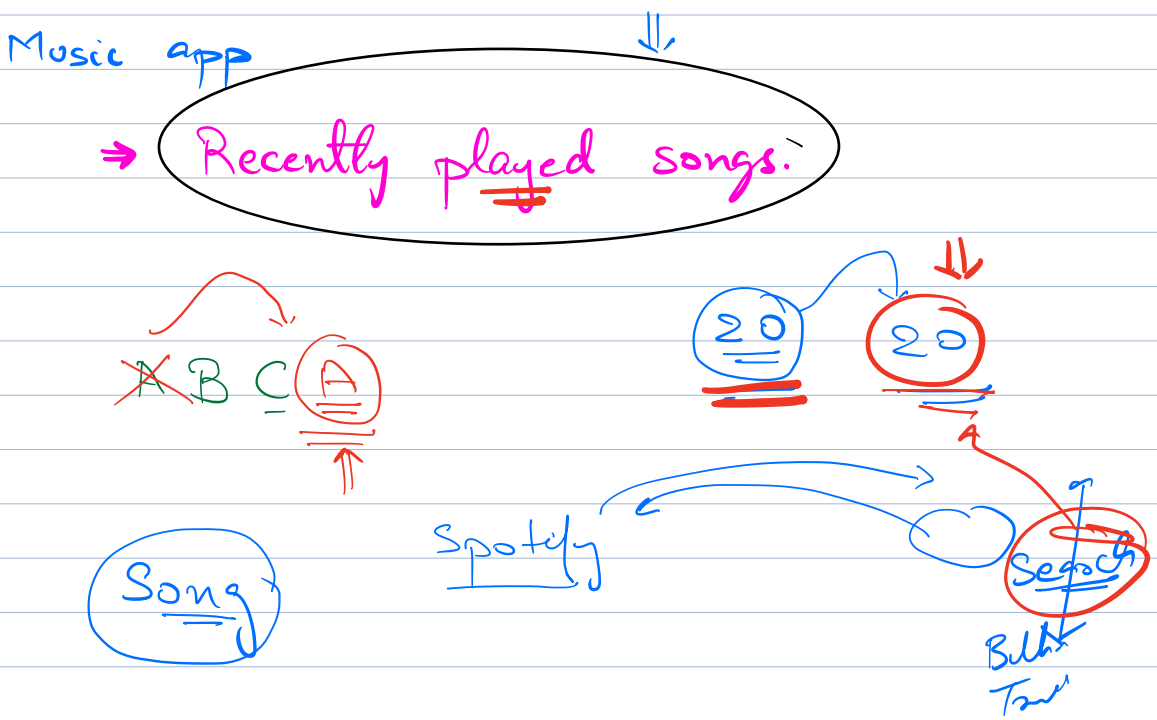
$$(x - 2y)l = M + K$$

$(M+K)$ is an integral multiple of L



length = 10

Break till 11:02 PM

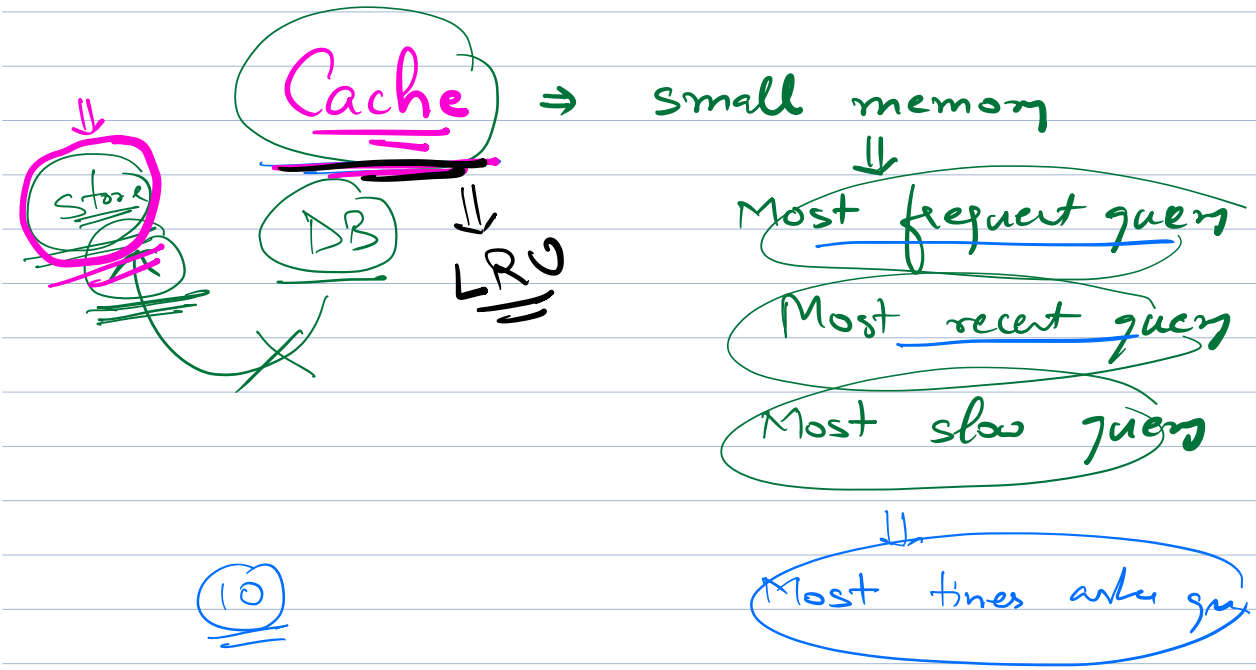


website ⇒ x seconds.

⇓

ajax ⇒ < x

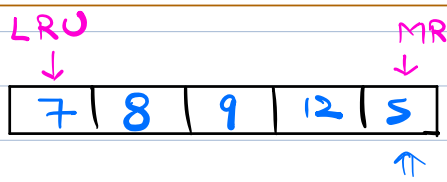
Eclipse / Word / Excel ⇒



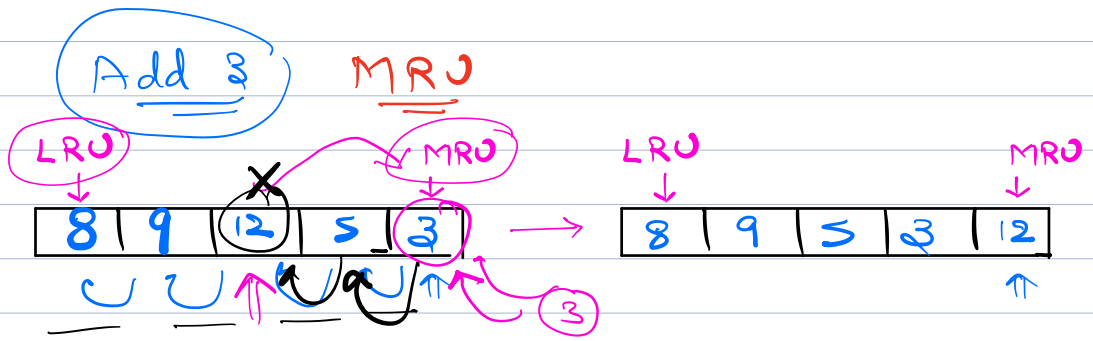
Music ⇒ Recently played songs.

cacheSize = 4

LRU ⇒ Least Recently used.



7, 8, 9, 12, 5, 3, 12



data : x

Search

Not found

found

if (currSize == Capacity)

1) Remove from curr position

2) Insert at MRU

YES

NO

⇒ Remove from LRU

⇒ Insert at MRU.

⇒ Insert at MRU

1) Search (x)

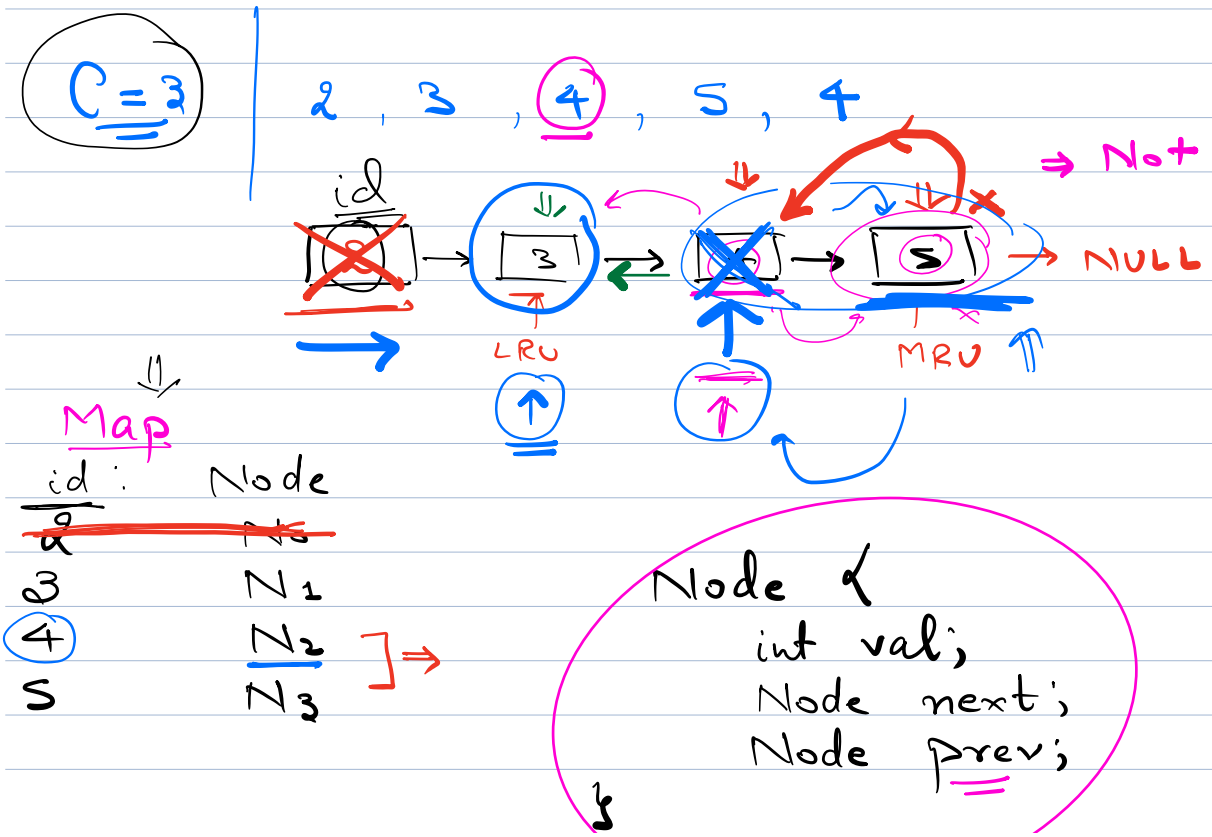
2) Deletion

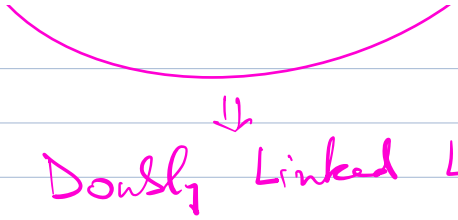
3) Insert at one end

	Array	LL	+HM + DLL
Search	O(N)	O(N) → O(1) using HM	O(N) → O(1) using HM
Delete	O(N)	O(1) (after search)	O(1)
Insert	O(1)	O(1) using tail.	O(1)

HashMap < Int, Node >

↑ address of node in cache




Doubly Linked List