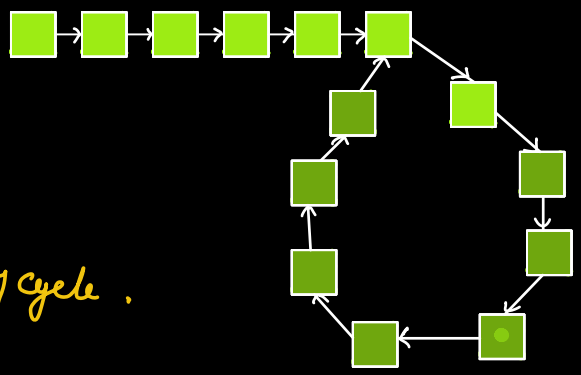
Cycle detection

1. Detect if there is a cycle.
2. Find the first node / start node of cycle.



Approach 1

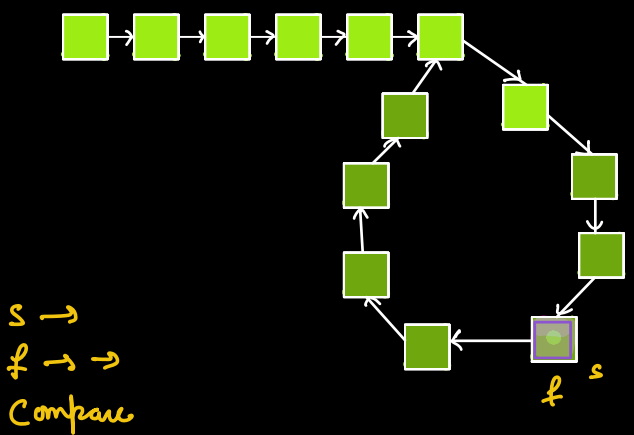
Use Set / Map

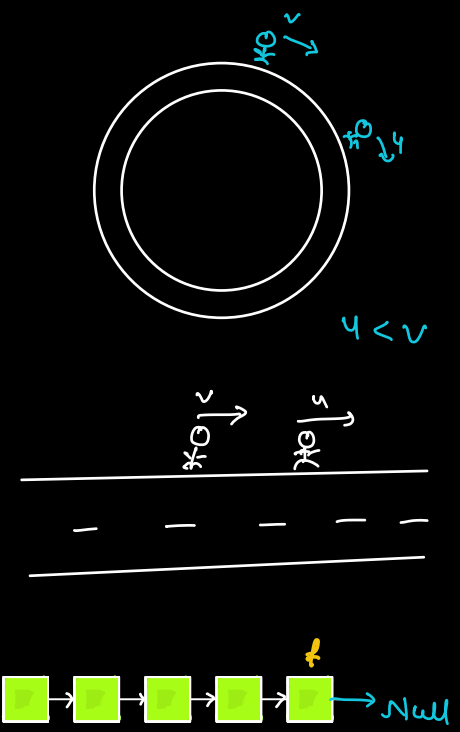
HashSet<ListNode> set;

- Iterate over the LL
  for every node:
    Check if it present in set
      if present → Cycle is present & curr node is start of it
      else add in set

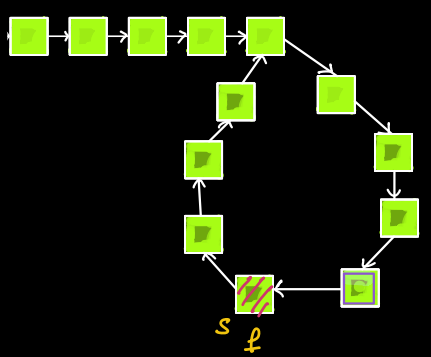
TC: O(N)
SC: O(N)

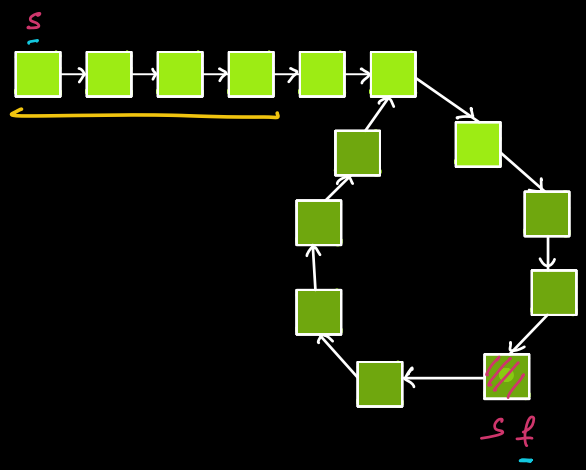
fast & slow pointers



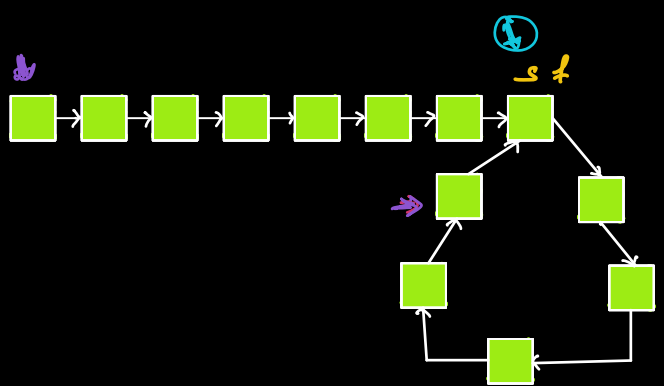


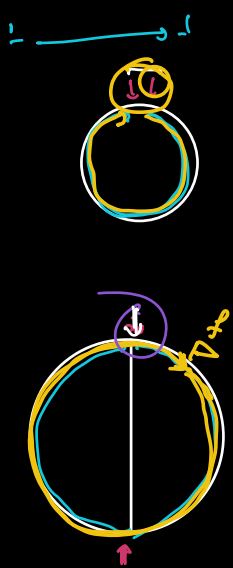

```
if (head == null) ret false;
while (fast != null && fast.next != null) {
    slow = slow.next;
    fast = fast.next.next;
    if (slow == fast)
        ret true;
}
```





$N = 9$





If length is l

& we start from the start of cycle

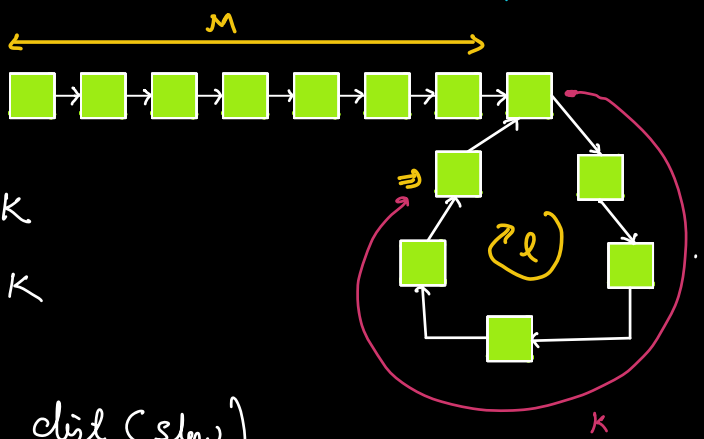
l iterate $\longrightarrow$ starting point

$2l$ iterate $\longrightarrow$ starting point

$3l$ iterate $\longrightarrow$ starting point

$\vdots$

Nl $\longrightarrow$ Start



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

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

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

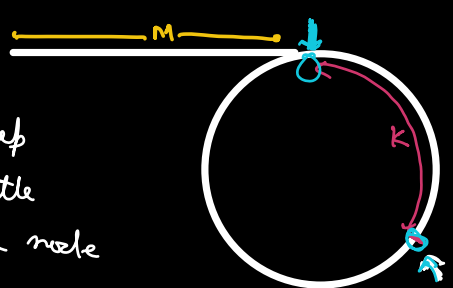
$M + xl + K = 2M + 2yl + 2K$

$xl - 2yl = M + K$

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

$(M+K) \Rightarrow$ integral multiple of l

If we start iterating from the 1st node of cycle $\xrightarrow{(M+K)}$ end up at the same node



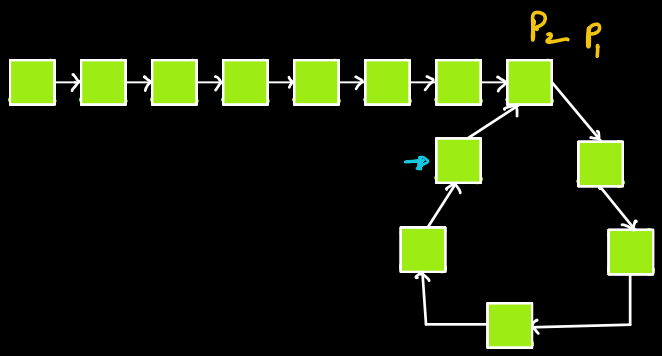
The meeting point is at K dist from the start.

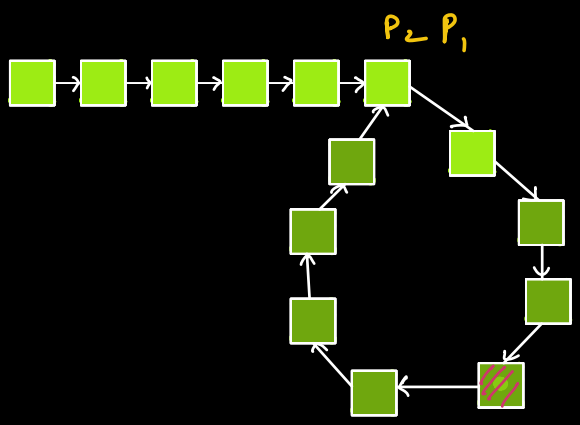
So M iterations from the meeting point $\longrightarrow$ start node

To do M iterations from the meeting point:

start one pointer from head

the other from meeting point $\longrightarrow$





TC : O(N)

d

↓

d-1

↓

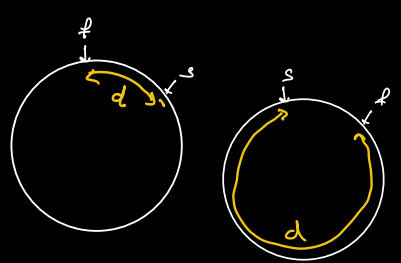
d-2

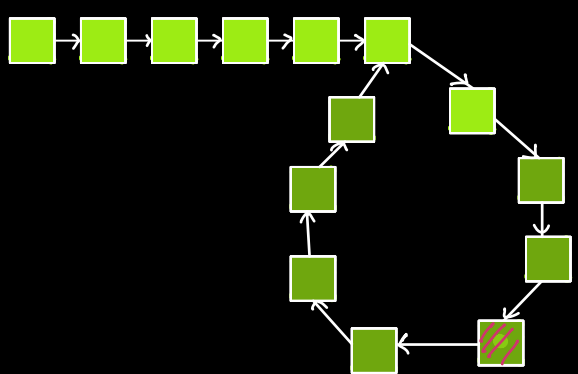
↓

d-3

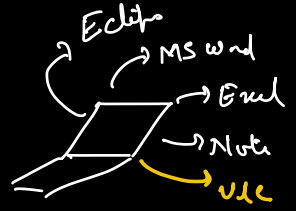
⋮

0





Cache



Cache Eviction Remove from the cache.

LRU : Least Recently Used
(Cache eviction policy)

Music

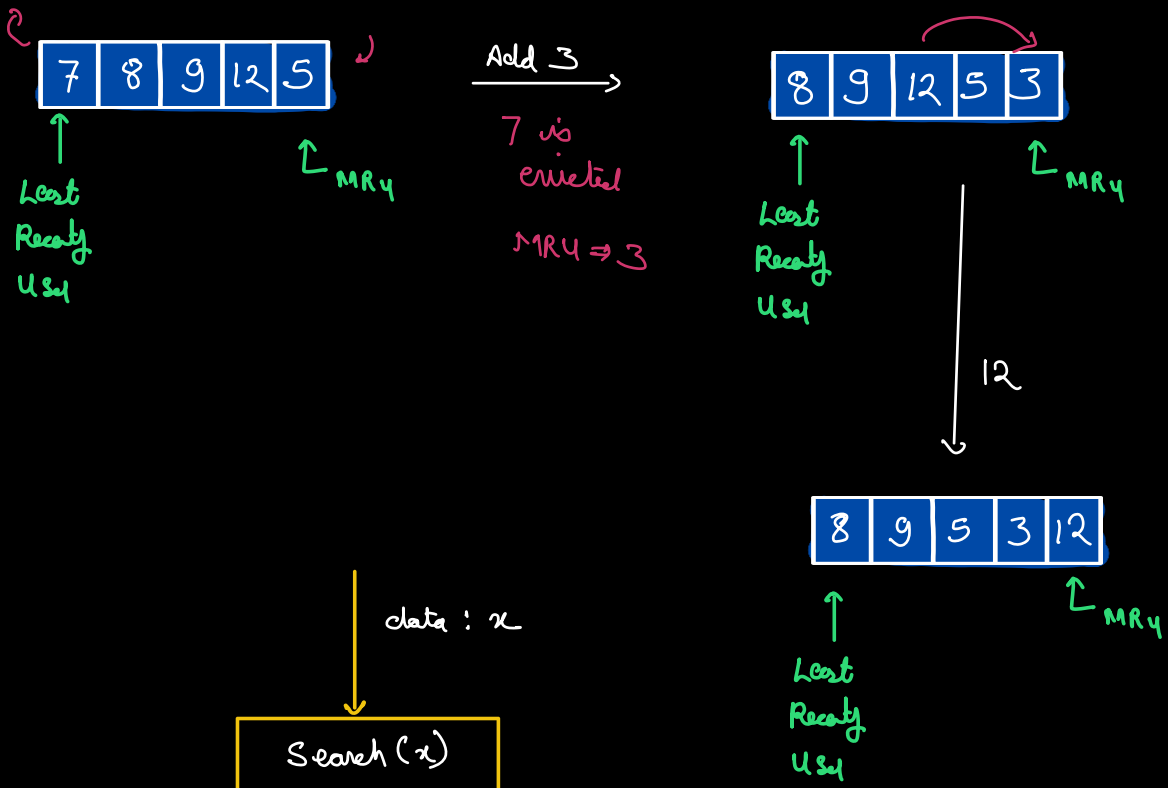
MRP ⇒ Munni
Rock
Enemy
LRP ⇒ Believer

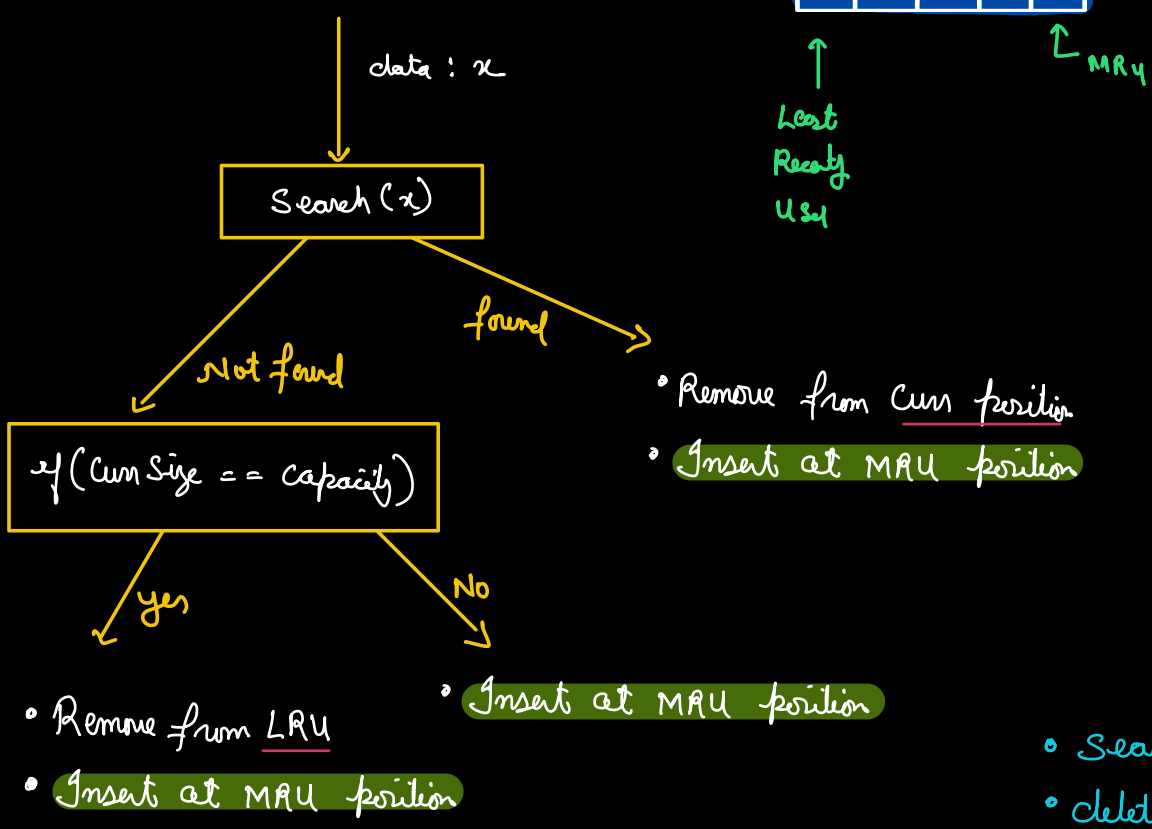
Swiggy

MRO → On 3
On 1
On 2
LRO → On 4

Browser history

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





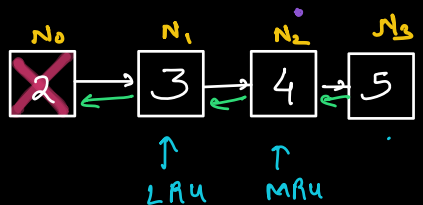
- Search(x)
- delete
- Insert at one end

	Arrays	LL	DLL + HM
Search	O(N)	O(N) → O(1) (HM)	O(1)
delete	O(N)	O(1) (close after search)	O(1)
Insert (at one end)	O(1)	O(1)	O(1)

To optimize search ⟶ HashMap

HashMap < Integer, ListNode > map;

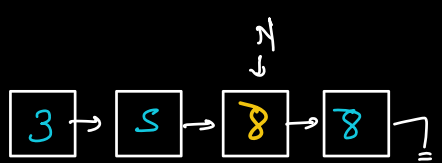
C = 3 | 2, 3, 4, 5, 4



Size : ∅ ≠ 2 ≠ 3

Map

id	:	Node
~~2~~	:	~~N_0~~
3	:	N_1
4	:	N_2
5	:	N_3



MFU, LFU

TTL + LRU
↳ time to live.

```
class DLL {
    int data;
    DLL next;
    DLL prev;
}
```