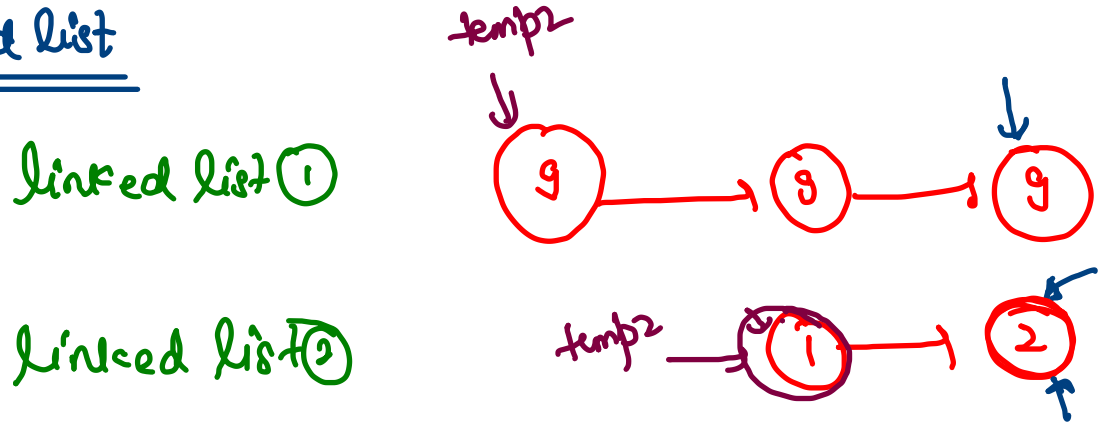
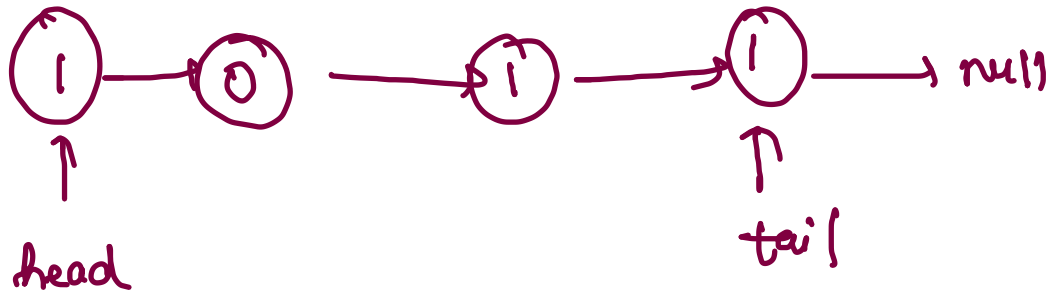
Add two linked list

Time Complexity → O(n)

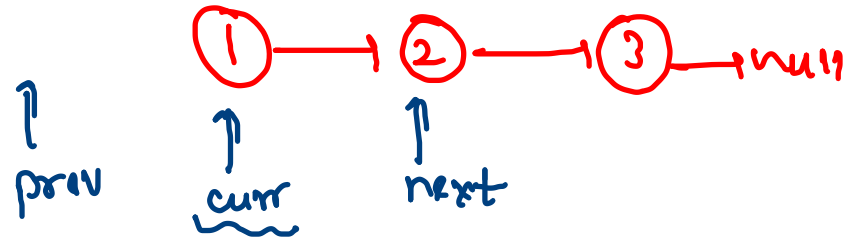


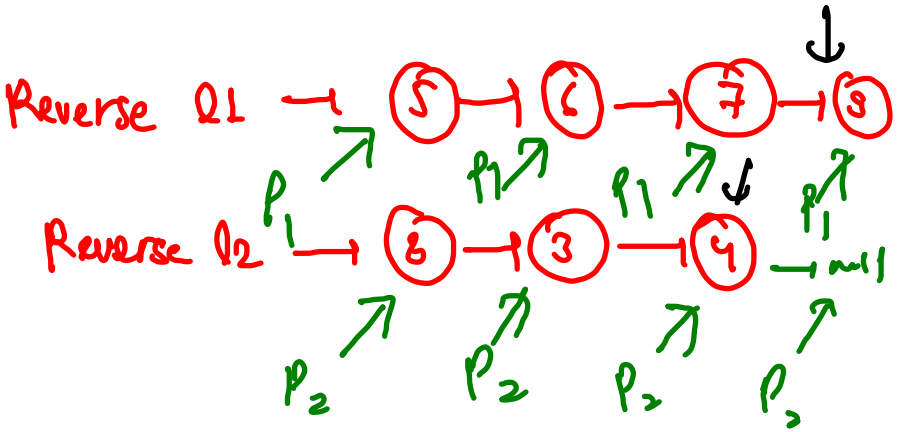
Result →



Reverse Pointer → Algo →

1. curr_next = prev
2. prev = curr
3. curr = next
4. next = next.next; } Manage curr





carry = ~~0~~ ~~1~~ ~~1~~ 1

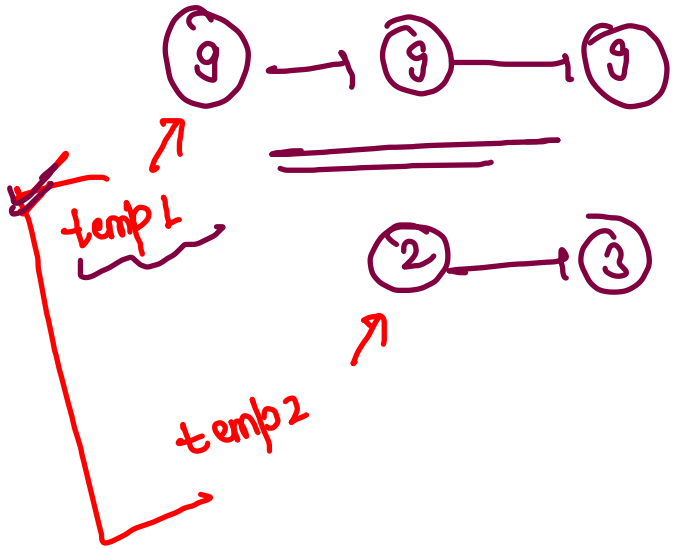
```
Sum =   p1.data + p2. data + carry
val =   sum % 10;
carry = sum / 10;
p1 = p1.next;
p2 = p2.next;
```

```
while(p1 != null) {

}
while( p2 != null){

}
```

carry :



10^6 ← $l_1.size()$??

$l_2.size()$??

return $\{10^6 + 1\}$

|| carry != 0) {

while(p1 != null || p2 != null)

int val1 = p1 == null ? 0 : p1.data;

int val2 = p2 == null ? 0 : p2.data;

}

$p_1 = p_1.next$;

$p_2 = p_2.next$

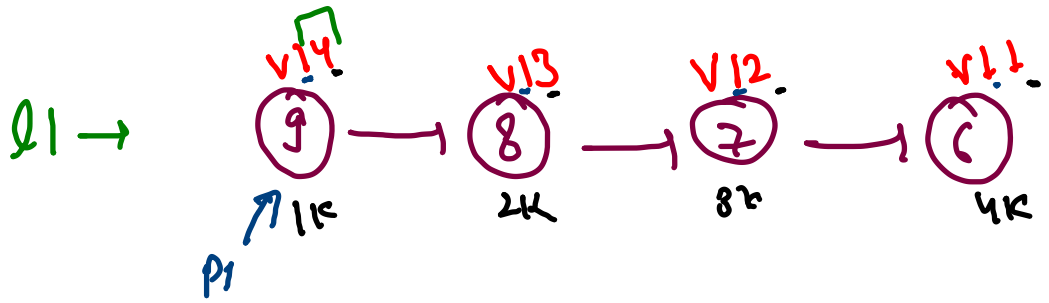
ternary operator.

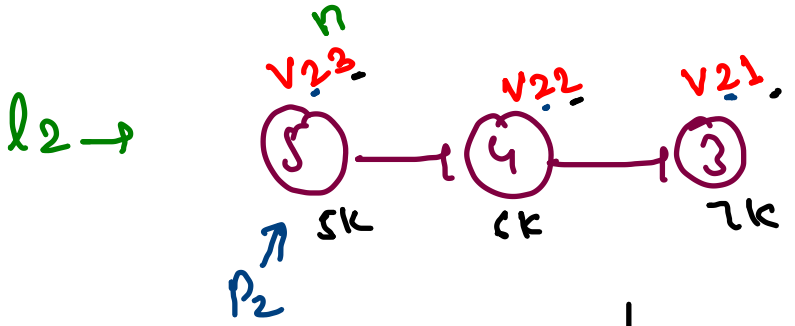
// p1 = p1 == null ? null : p1.next;

while( )

}

while( ) {





Addition →

V11 + V21

V12 + V22

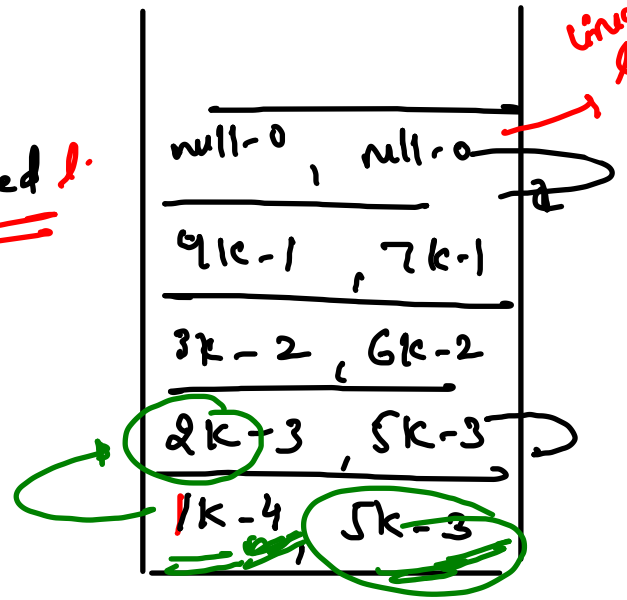
V13 + V23

Post]

LinkedList L
- Index
- pointer
- Linked List 2

LinkedLinked l.

carry = 0



V14 + carry.

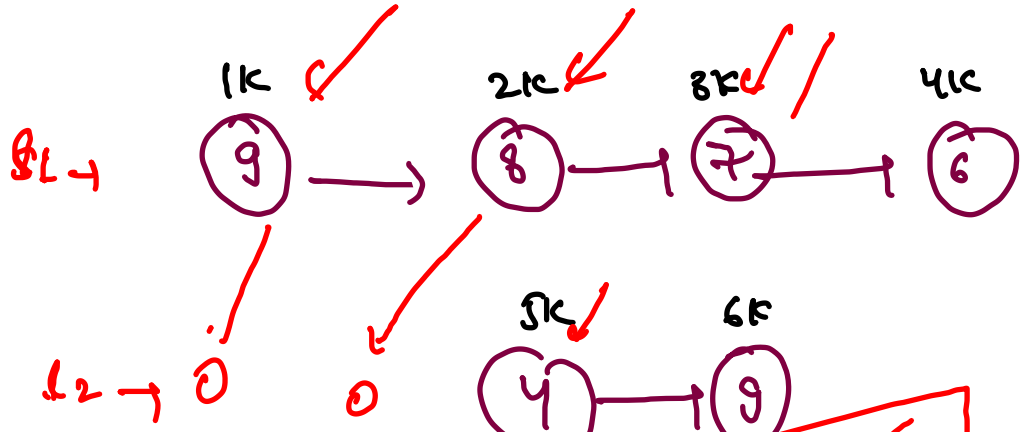
V15 + carry.

p1 = head1;

p2 = head2;

p1 = p1.next

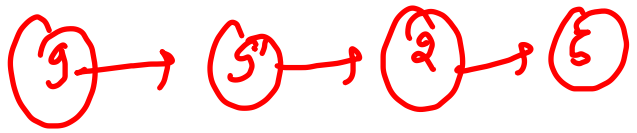
(5-1);



carry = 1 1 0

res = null

head tail size

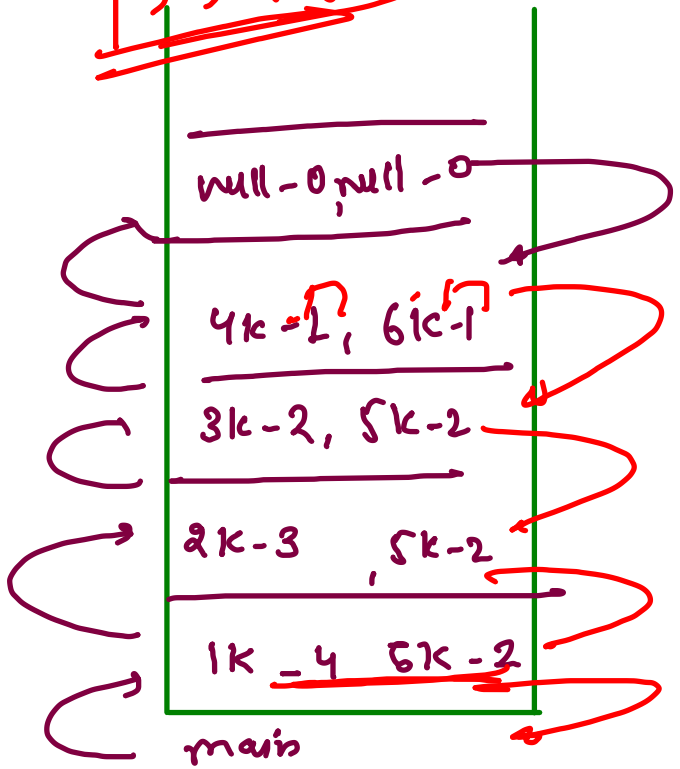


s1 = 4

s2 = 2

p1 = 1K

p2 = 5K



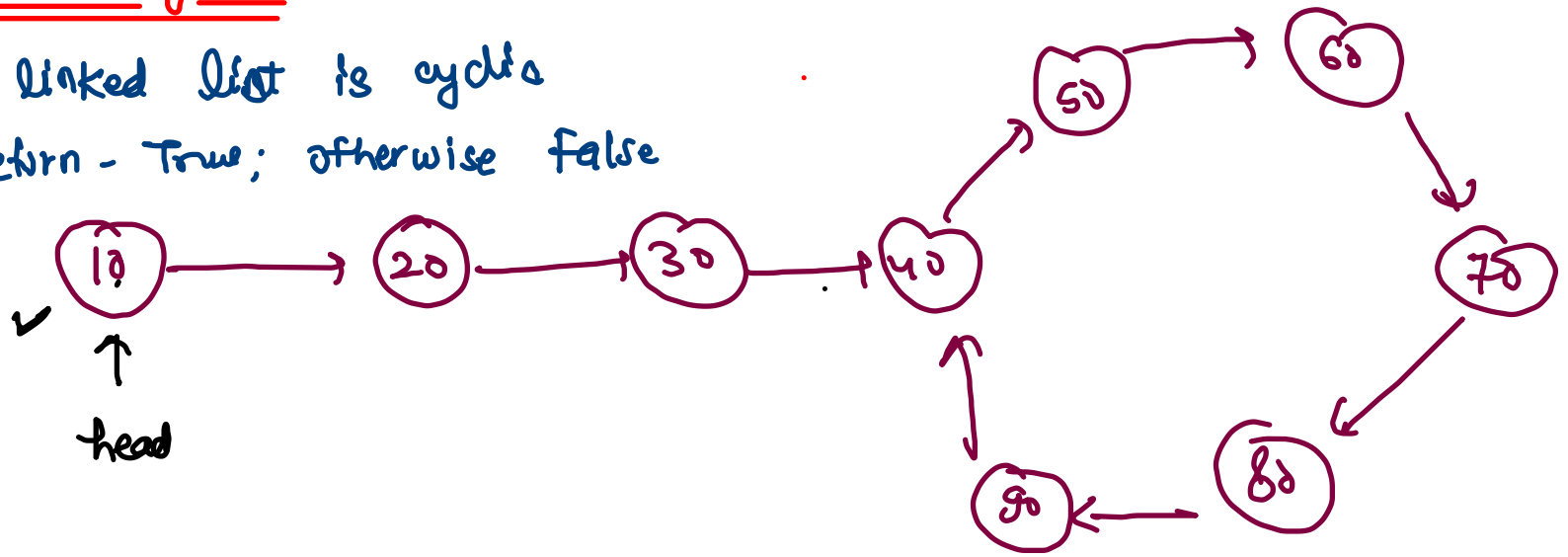
if

```
        res = new LinkedList();
        return;
    }
    int v1 = 0;
    int v2 = 0;
    if(indx1 == indx2) {
        recLsum(p1.next, p2.next, indx1 - 1, indx2 - 1);
        v1 = p1.data;
        v2 = p2.data;
    } else if(indx1 > indx2) {
        recLsum(p1.next, p2, indx1 - 1, indx2);
        v1 = p1.data;
    } else {
        recLsum(p1, p2.next, indx1, indx2 - 1);
        v2 = p2.data;
    }
    int sum = v1 + v2 + carry;
    res.addFirst(sum % 10);
    carry = sum / 10;
}
```

```
public static LinkedList addTwoLists(LinkedList l1, LinkedList l2) {
    // write your code here
    int s1 = l1.size();
    int s2 = l2.size();
    Node p1 = l1.head;
    Node p2 = l2.head;
    recLsum(p1, p2, s1, s2);
    if(carry != 0) {
        res.addFirst(carry);
```

Detect Cycle

if linked list is cyclic
Return - True; otherwise false



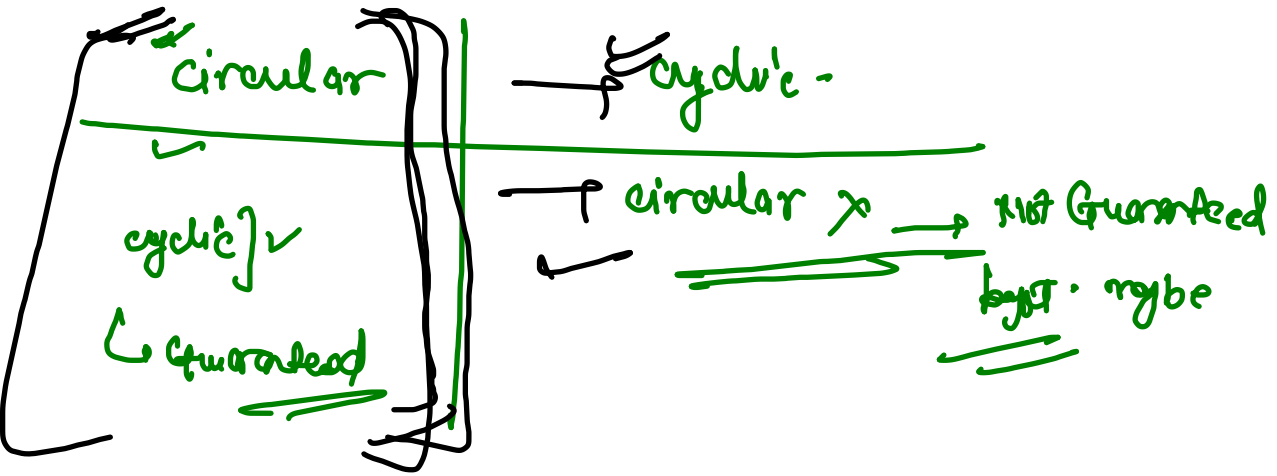
Hashmap

Node - visited

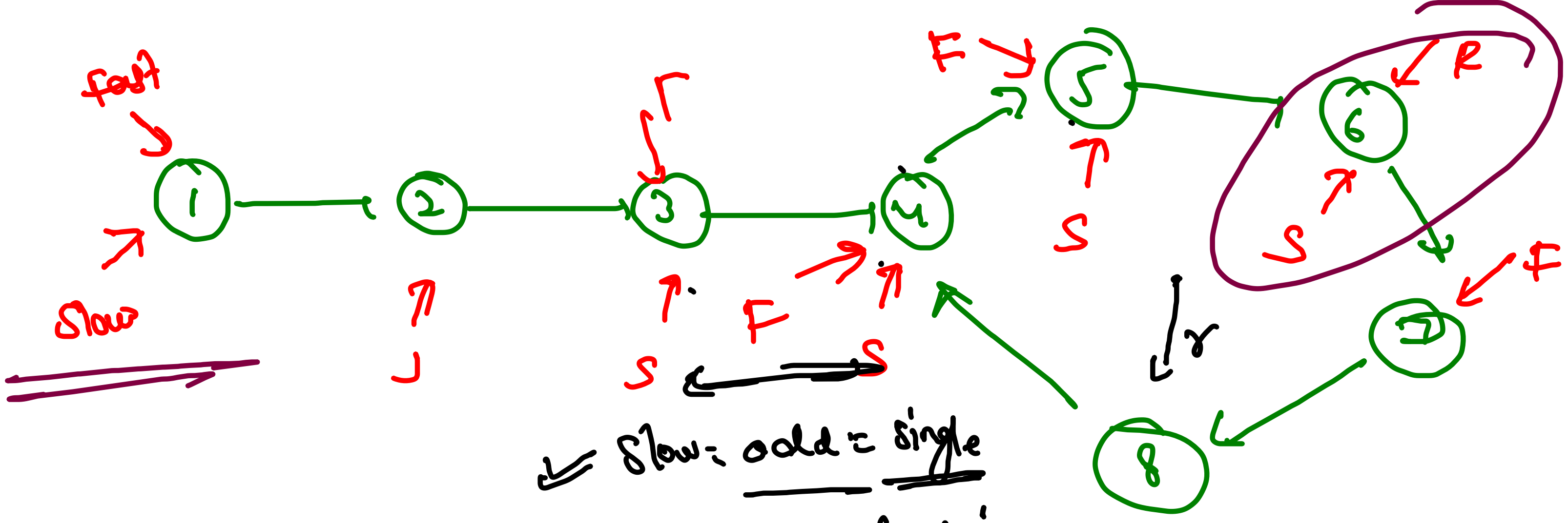
visited

Space

Amortized O(1)



Floyd cycle Detection method



slow == fast

slow = odd = single step jump

$2x$

Fast = Even - 2 jump

slow = slow.next

fast = fast.next.next;

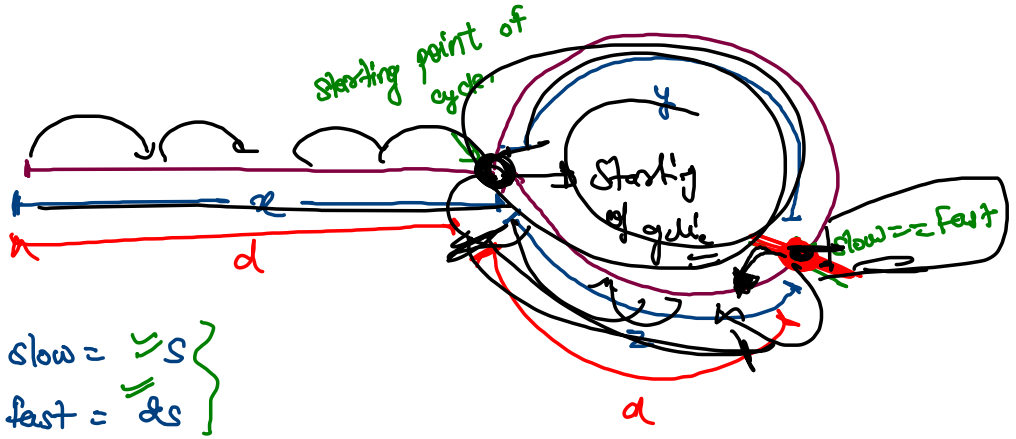
x,

| slow - single |

fast = $2x$

slow = x

= $2\pi r$



speed = distance / time.

time = distance / speed.

speed of slow = s
speed of fast = 2s

time taken to meet = 't'

$t = \frac{x+y}{s}$

distance travelled by slow = $x+y$

distance " " fast = $x + m(y+z) + y$. $t = \frac{x + m(y+z)+y}{2s}$

As 't' is equal for both

$$\frac{x+y}{s} = \frac{x+m(y+z)+y}{2s}$$

$2(y+z)$

$(y+z) + (y+z)$

$$2x+2y = x + m(y+z) + y$$

$$2x - x = m(y+z) + y - 2y$$

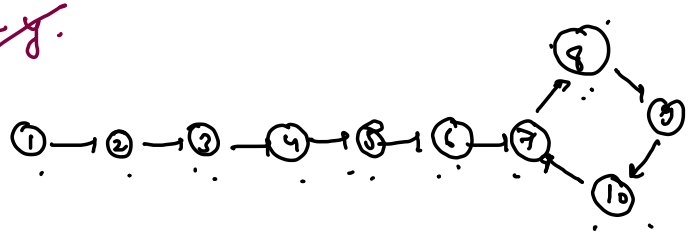
$$x = (m-1)(y+z) + (y+z) - y$$

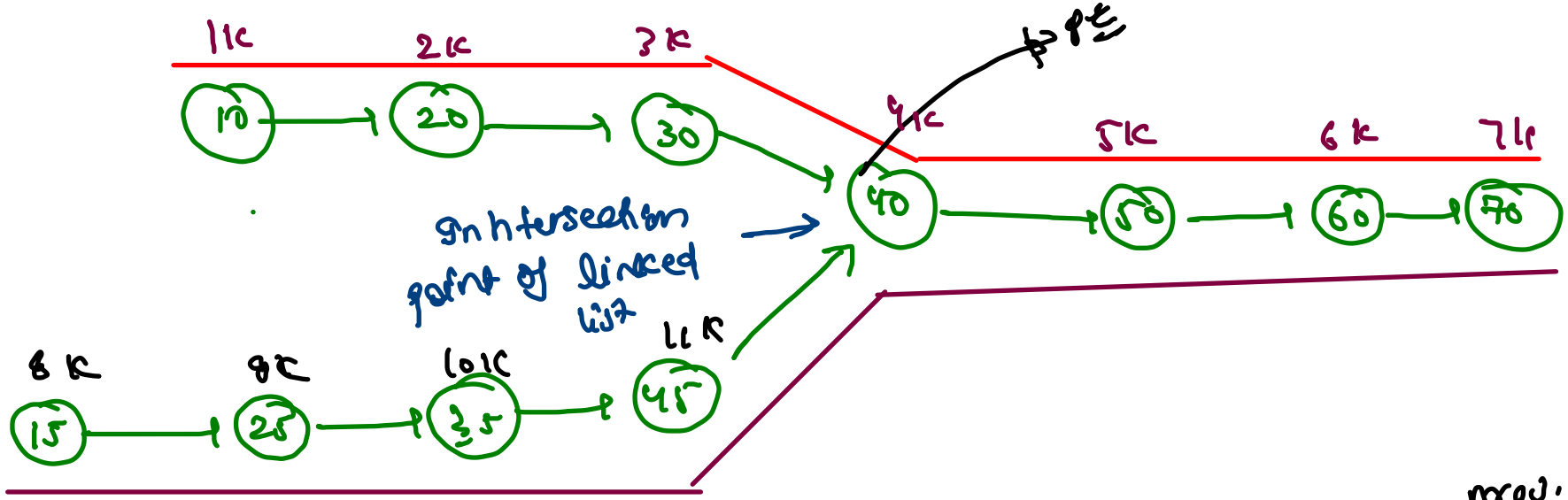
$$\boxed{x = (m-1)(y+z) + z}$$

displacement = 0 if $m = 1$

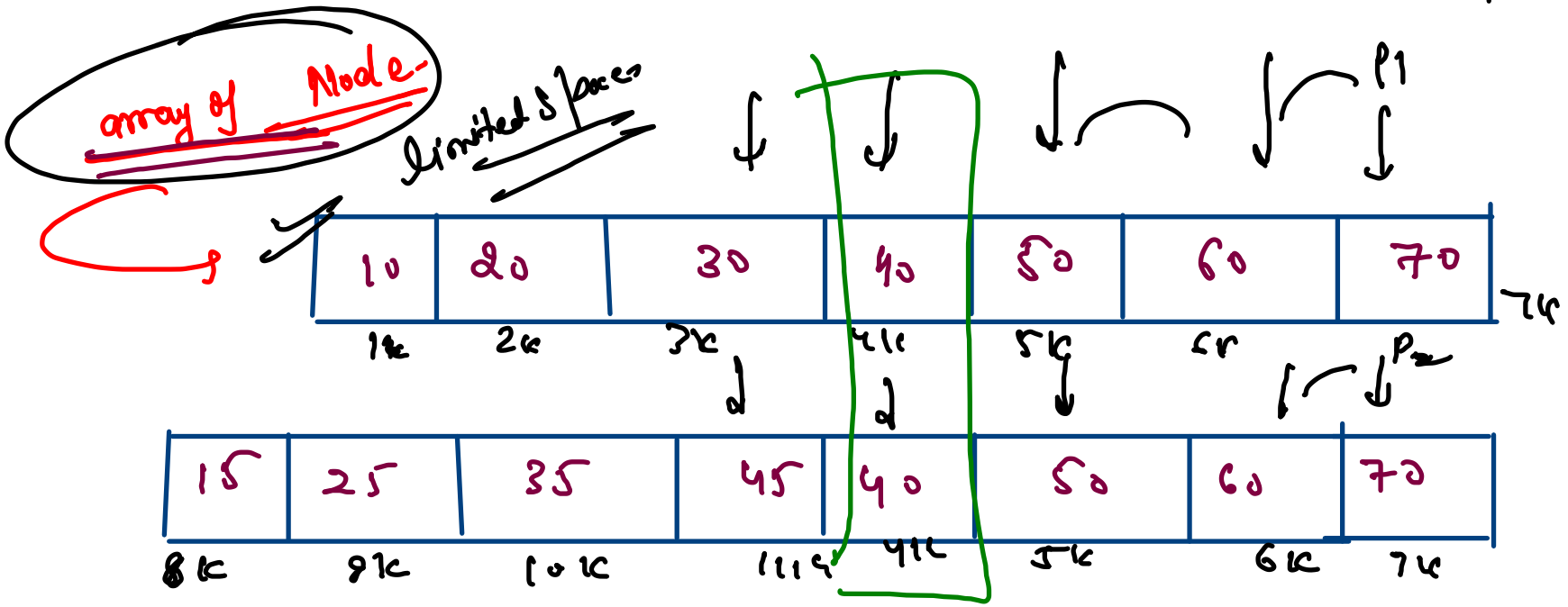
$x = (1-1)(y+z) + z$

$\boxed{x = z}$

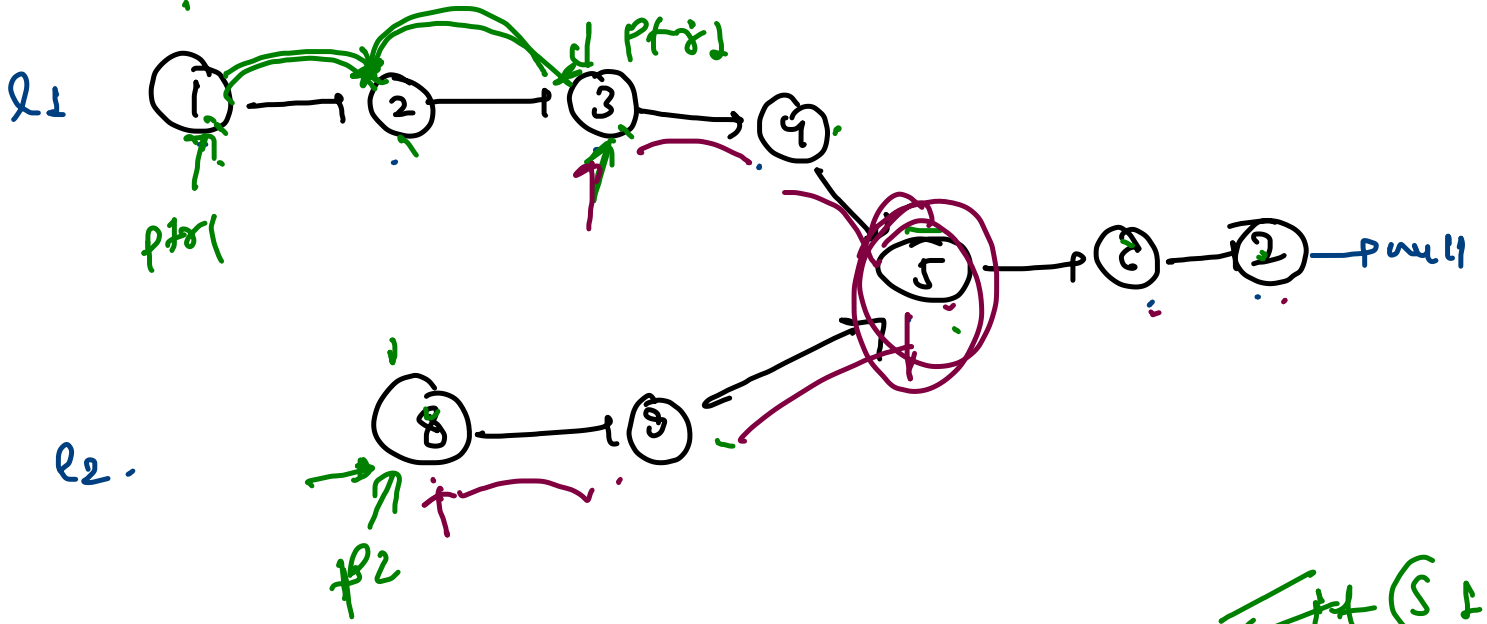




prev. = 70 65 40



Space Not allow



if (S1 > S2)

S2 → 7

S2 → 5

ptr1 ⟶ make it forward for (S1 − S2) time

ptr2

null;

while (ptr1 != null) {

if (ptr1 == ptr2) {

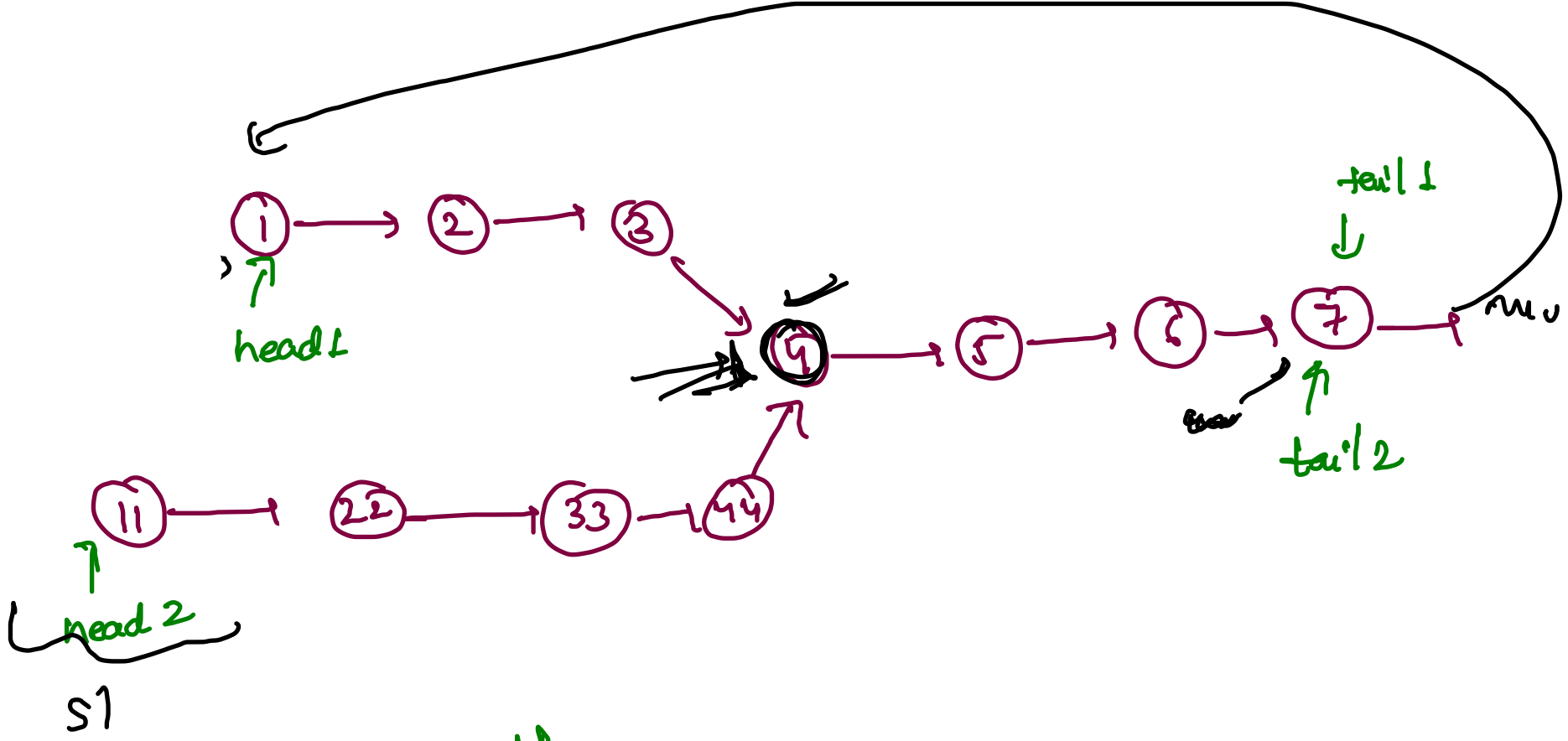
return ptr1;

}

ptr1 = ptr1.next;

ptr2 = ptr2.next;

}



n tail = tail 1;

n tail . next = head 1

find starting of cycle;

n tail . next = null;

Slow & fast start from head 2

OR

n tail . next = head 2;

pseudo code · → Geeks' ≤

check cyclic ??

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

```
if isCyclic == true {
    slow = head;
    while( slow != fast){
        slow = slow.next;
        fast = fast.next;
    }
    return slow.data;
```

= Intersection of two linked list??