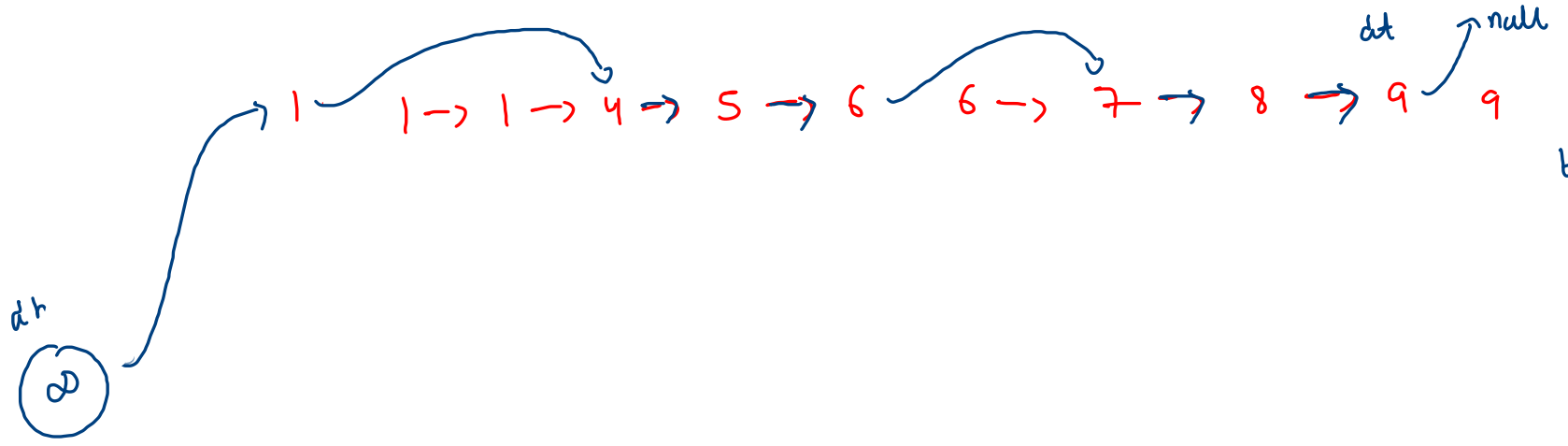
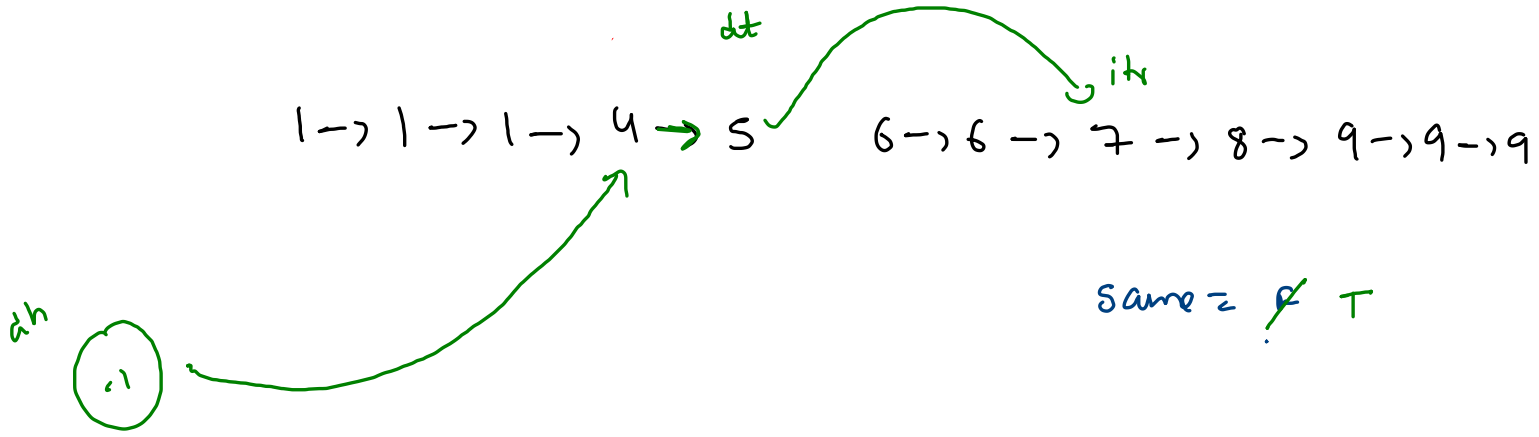
Remove Duplicate From Sorted Linkedlist



1->1->1->4->5->6->6->7->8->9->9->9->null

Remove All Duplicates From Sorted Linkedlist



(i) o(n) : T

(ii) o(1) : S

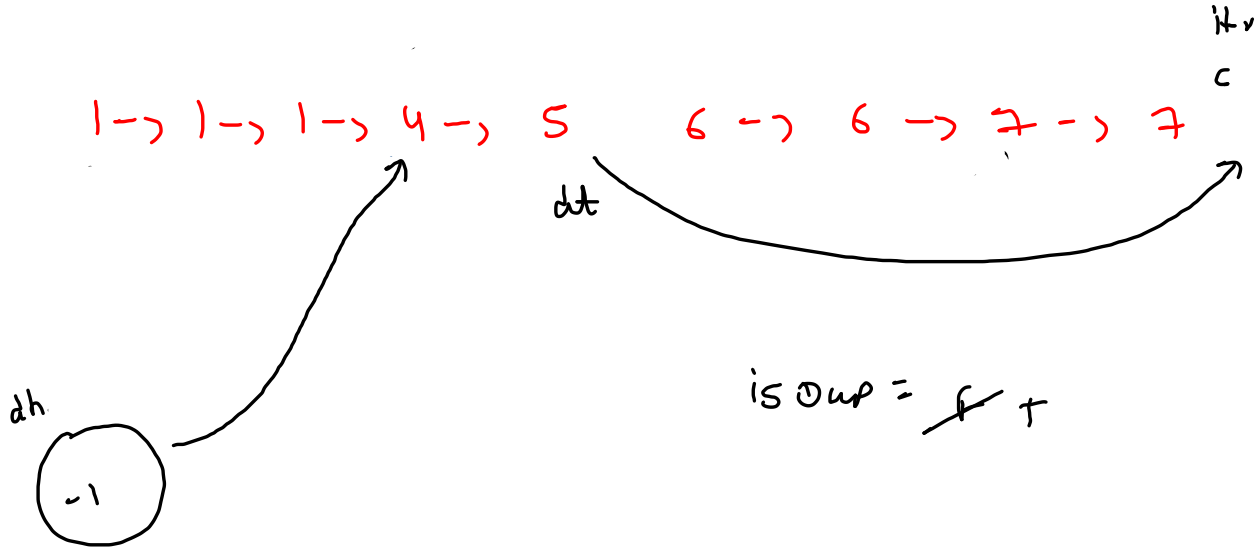
same = ~~F~~ T

(i) extra space is not allowed.

output : 4 -> 5 -> 7 -> 8

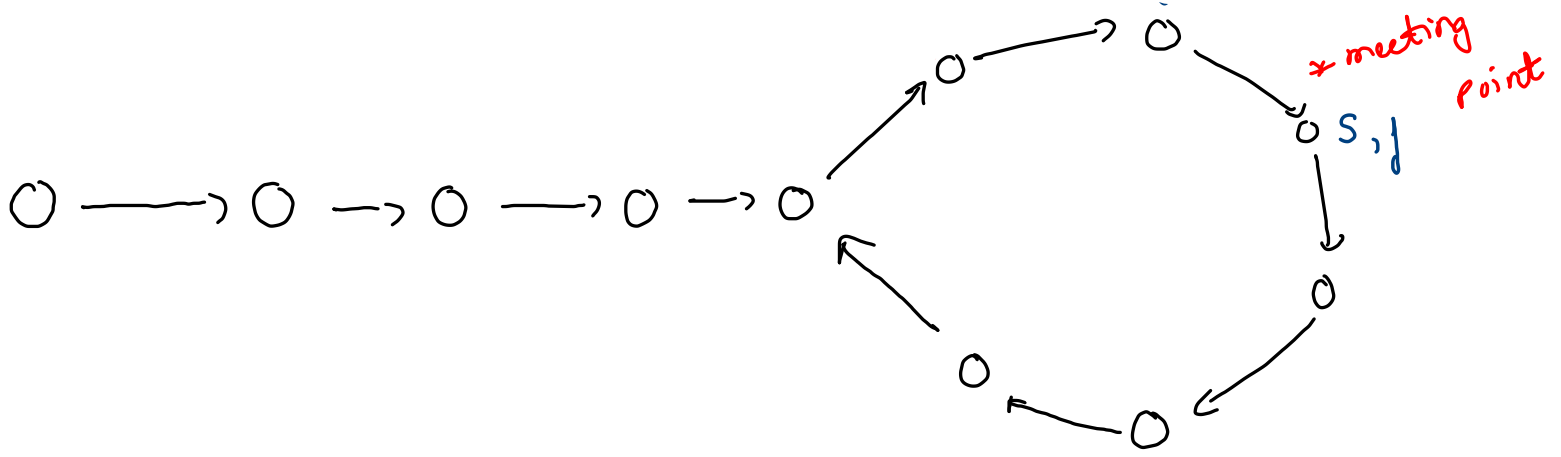
1->1->1->4->5->6->6->7->8->9->9->9->null

```
ListNode itr = head;
ListNode curr = itr.next;

dt.next = itr;
```



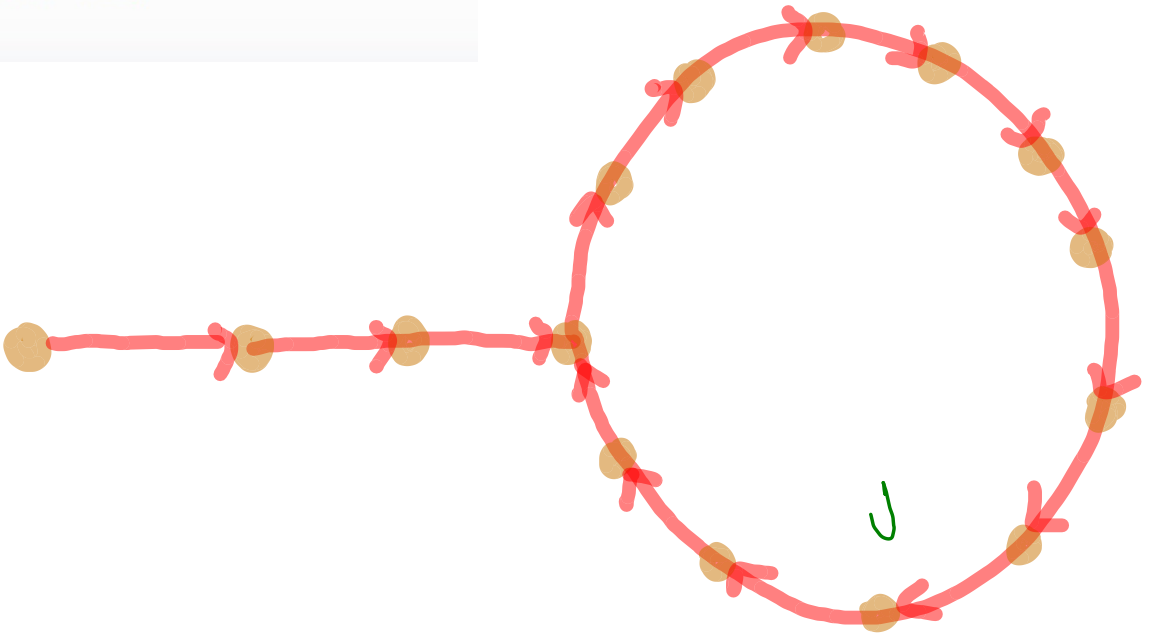

```
while(curr != null) {
    boolean isDuplicate = false;

    while(curr != null && itr.val == curr.val) {
        isDuplicate = true;
        curr = curr.next;
    }

    if(isDuplicate == true) {
        dt.next = curr;
    }
    else {
        dt = dt.next;
    }

    itr = curr;

    if(itr != null)
    curr = itr.next;
}
```

Is Cycle Present In Linkedlist



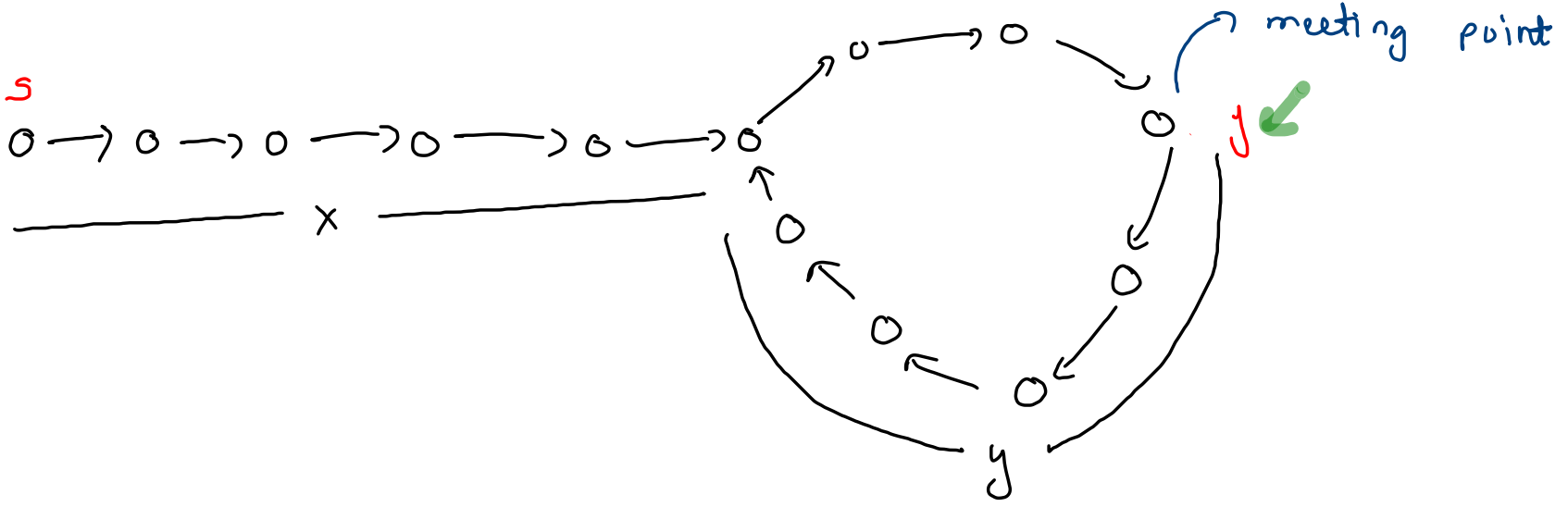
if slow is able to catch fast, there is a cycle LL.

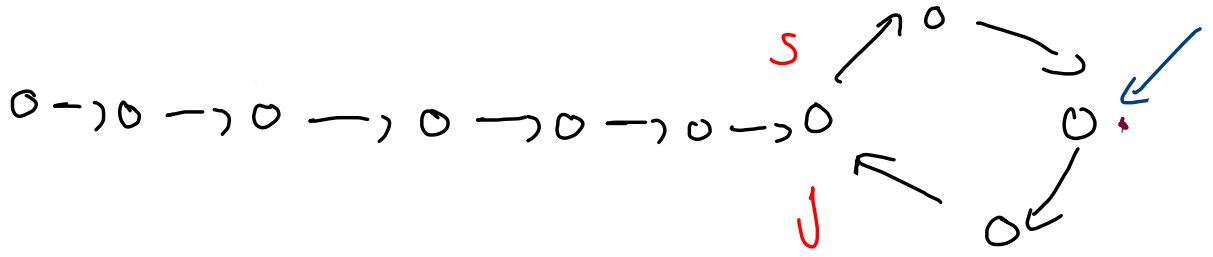
```java
ListNode slow = head;
ListNode fast = head;

while(fast != null && fast.next != null) {
    slow = slow.next;
    fast = fast.next.next;

    if(slow == fast) {
        return true;
    }
}
```



$r_s = 0$

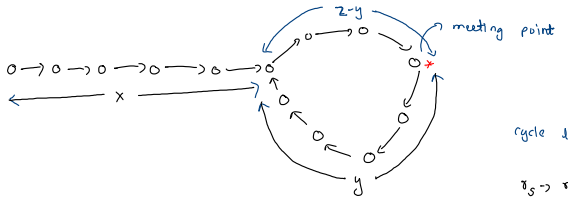
$r_f = 1$





$X = 2k + y$

$Z = 4 * 1 + 3$



cycle length → z

r_s → rotation by slow.
r_f → rotation by fast.
} before meeting

$d_s = x + r_s(z) + z - y$

$d_f = x + r_f(z) + z - y$

$t_s = t_f$ $\quad s = \frac{d}{t}$.

$t_s = \dfrac{x + r_s(z) + z - y}{z}$ — (1)

$t_f = \dfrac{x + r_f(z) + z - y}{2}$ — (2)

(1) = (2)

$$x + r_s(z) + z - y = \frac{x + r_f(z) + z - y}{2}$$

$$2x + 2r_s(z) + 2(z-y) = x + r_f(z) + z - y$$

$$x + 2r_s(z) + z - y = r_f(z)$$

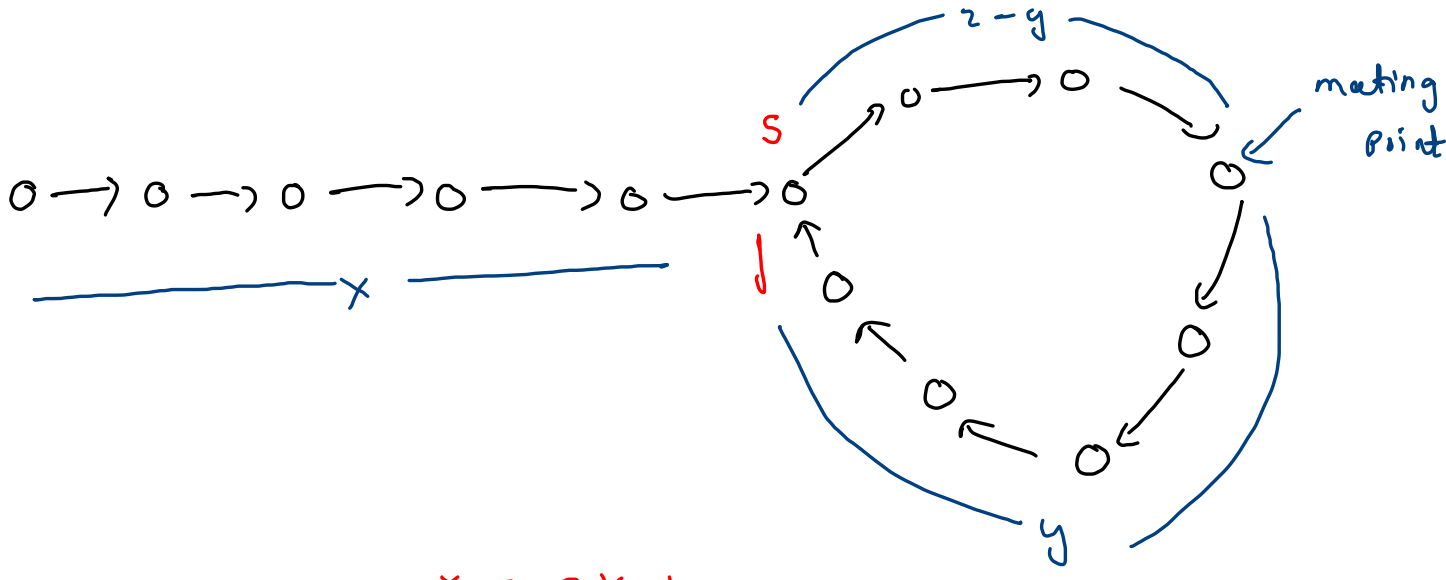
$$x = r_f(z) - 2r_s(z) - (z - y)$$

$$x = z(r_f - 2r_s) - (z - y)$$

$$\boxed{x = z(r_f - 2r_s - 1) + y}$$

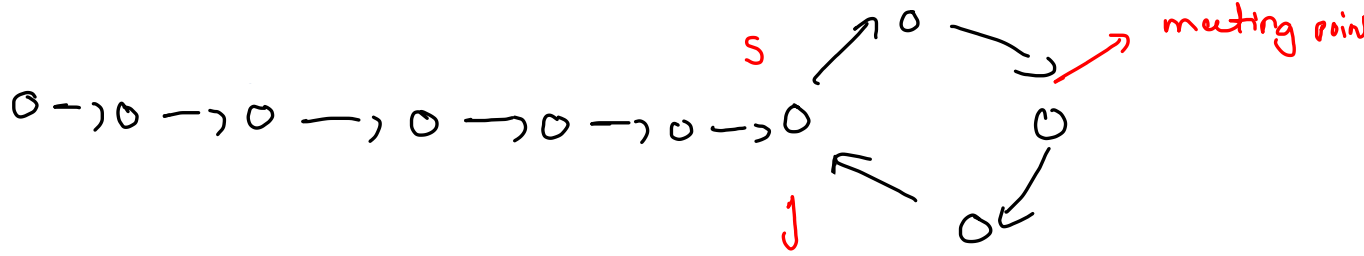
$$\boxed{x = zk + y}$$

```java
while(fast != null && fast.next != null) {
    slow = slow.next;
    fast = fast.next.next;

    if(slow == fast) {
        //keep fast at meeting point and move
        slow = head;

        while(slow != fast) {
            slow = slow.next;
            fast = fast.next;
        }

        return slow; //cycle start node
    }
}
```



$x = 2k + y$

$(k = 0)$

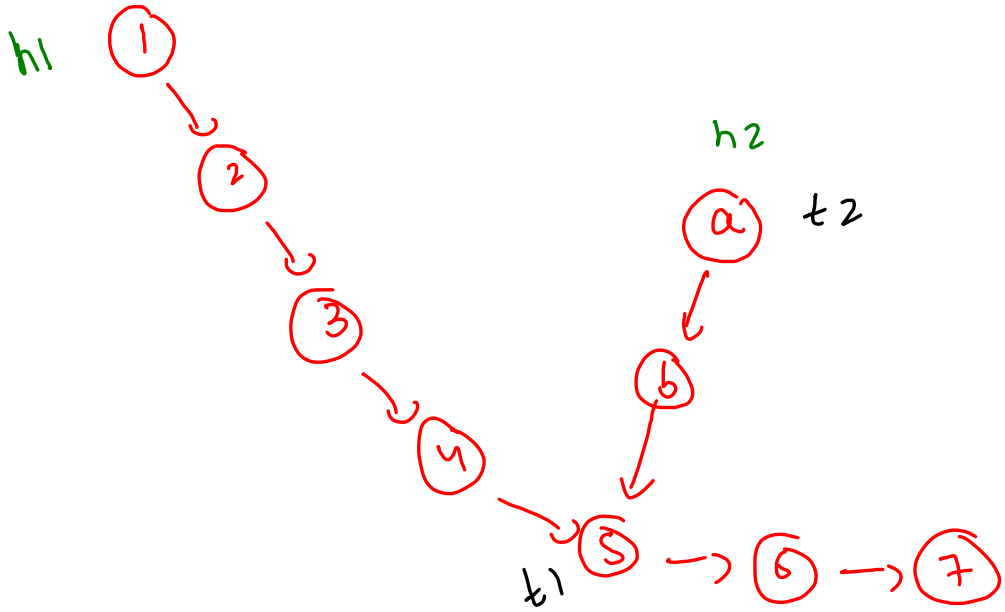



```
while(fast != null && fast.next != null) {
    slow = slow.next;
    fast = fast.next.next;

    if(slow == fast) {
        //keep fast at meeting point and move
        slow = head;

        while(slow != fast) {
            slow = slow.next;
            fast = fast.next;
        }

        return slow; //cycle start node
    }
}
```

$x = 2k + y$

$(k = 1)$

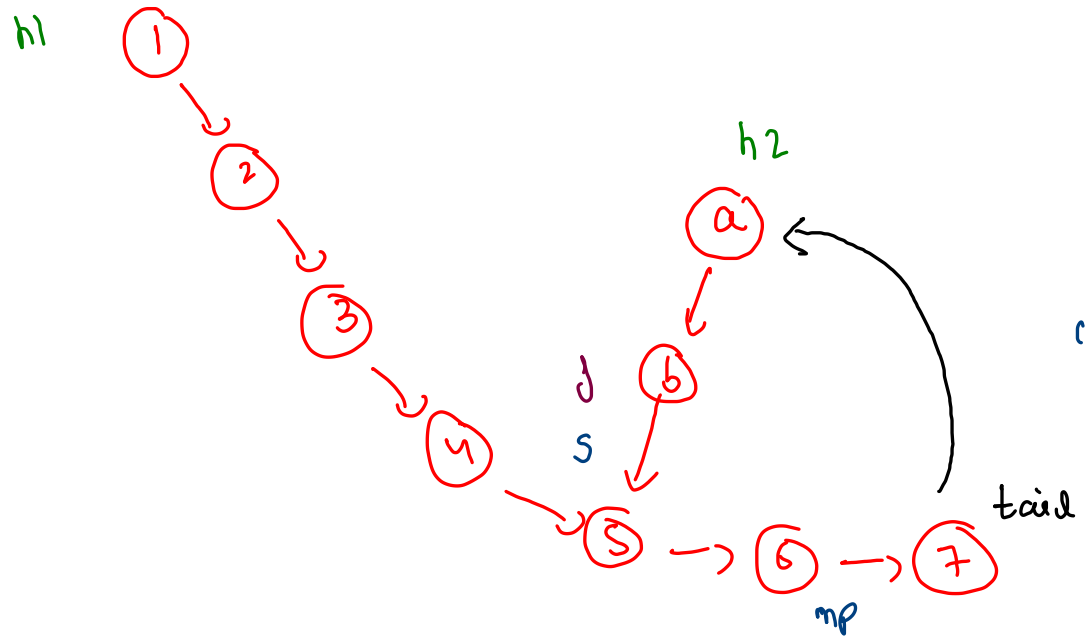
Intersection Node In Two Linkedlist Using Difference Method



abs. gap = $|d_1 - d_2|$

$= |7 - 5| = 2$

Intersection Node In Two Linkedlist
Using Floyad Cycle Method

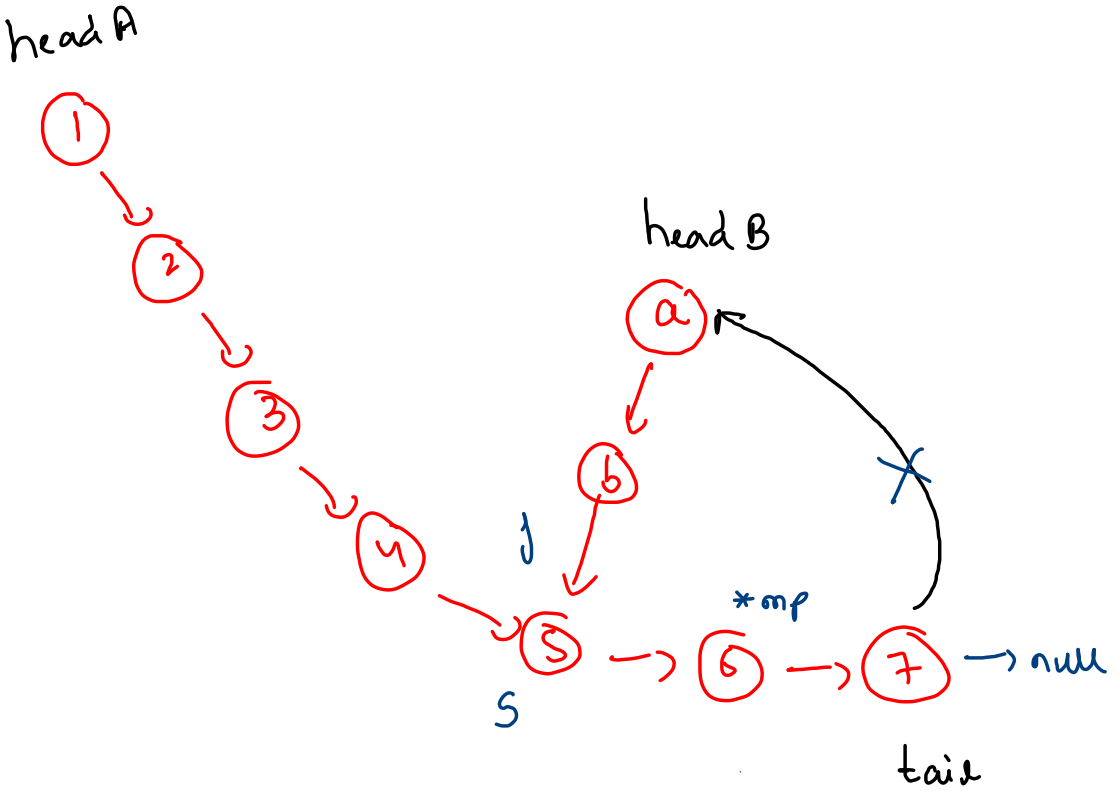


find tail

tail.next = h2;

cycle start = intersection node

```java
ListNode temp = headA;

while(temp.next != null) {
    temp = temp.next;
}

ListNode tail = temp;

tail.next = headB; //to create cycle, if there is an intersection point

//this cycle start point is the intersection point
ListNode slow = headA;
ListNode fast = headA;

while(fast != null && fast.next != null) {
    slow = slow.next;
    fast = fast.next.next;

    if(slow == fast) {
        slow = headA;

        while(slow != fast) {
            slow = slow.next;
            fast = fast.next;
        }

        tail.next = null;
        return slow;
    }
}

tail.next = null;
return null; //there was no intersection point
```



```
ListNode temp = headA;

while(temp.next != null) {
    temp = temp.next;
}

ListNode tail = temp;

tail.next = headB; //to create cycle, if there is an intersection point

//this cycle start point is the intersection point
ListNode slow = headA;
ListNode fast = headA;

while(fast != null && fast.next != null) {
    slow = slow.next;
    fast = fast.next.next;

    if(slow == fast) {
        slow = headA;

        while(slow != fast) {
            slow = slow.next;
            fast = fast.next;
        }

        tail.next = null;
        return slow;
    }
}

tail.next = null;
return null; //there was no intersection point
```