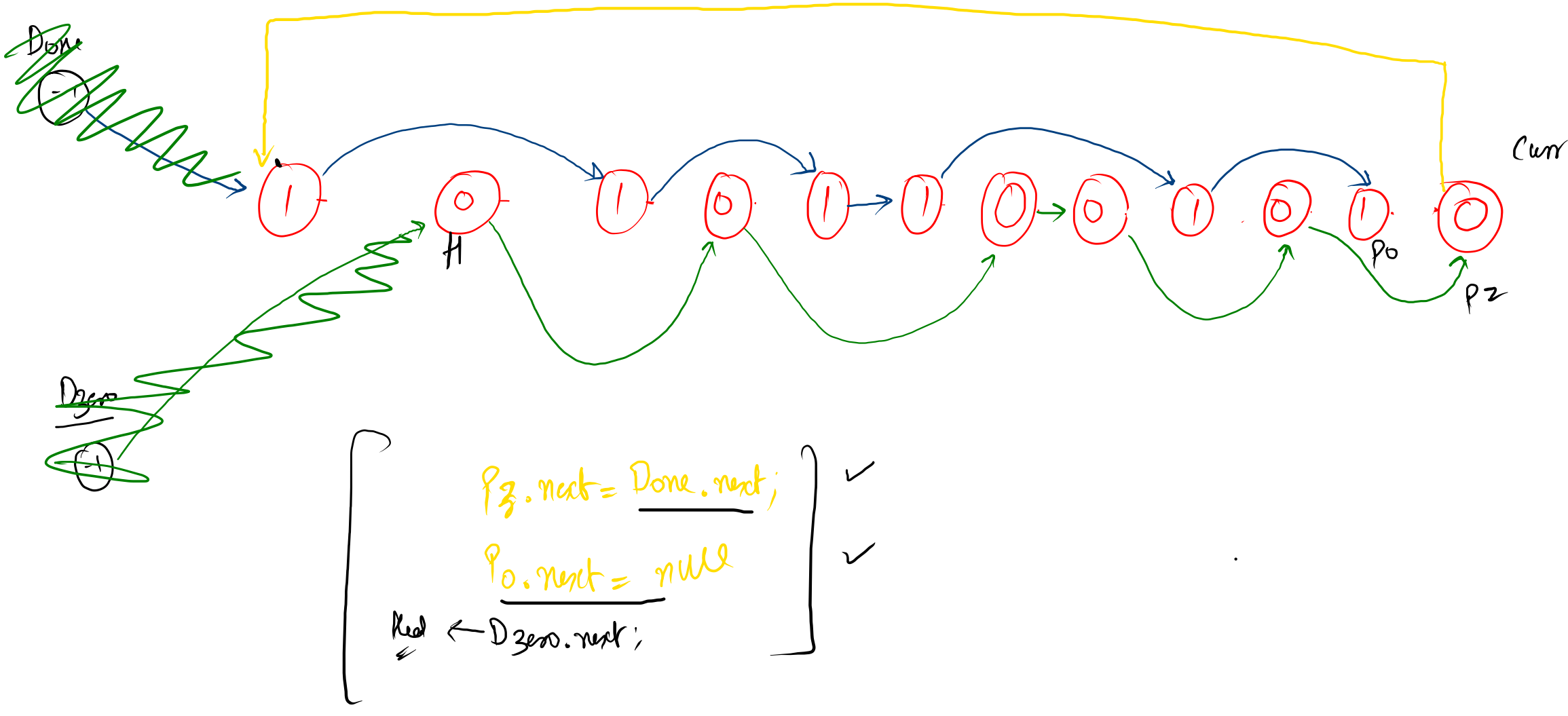
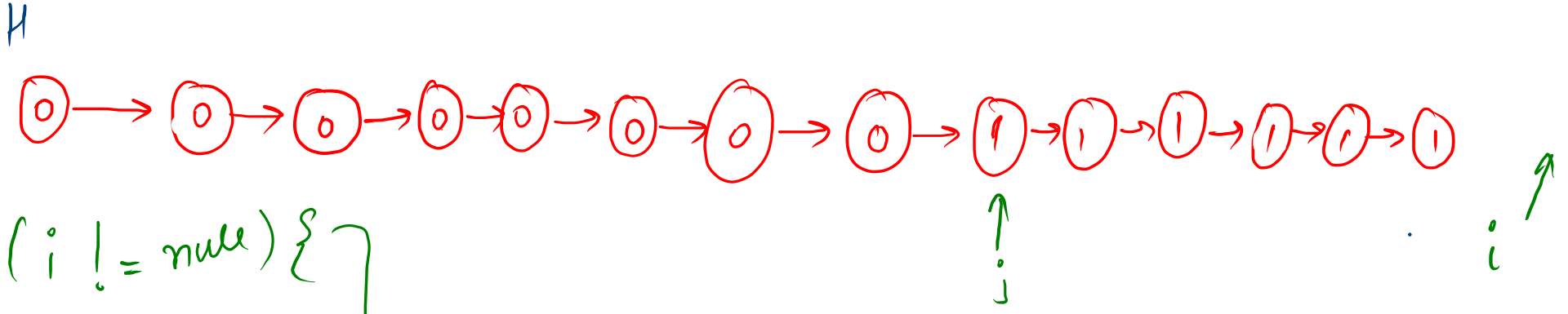
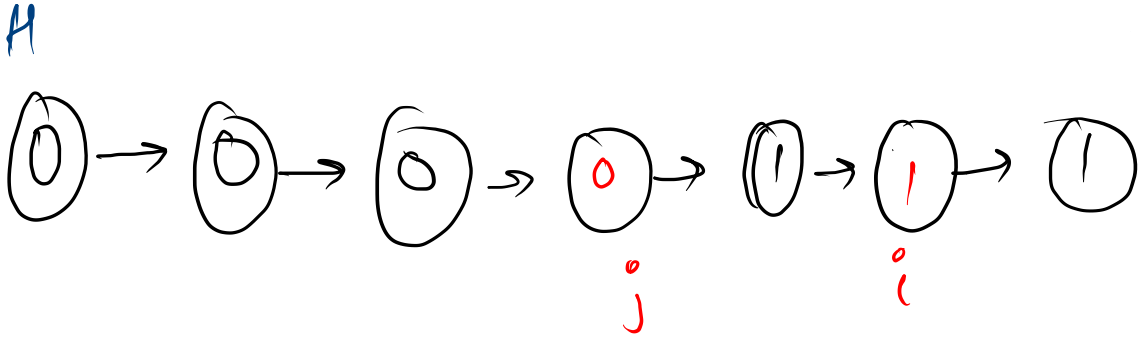
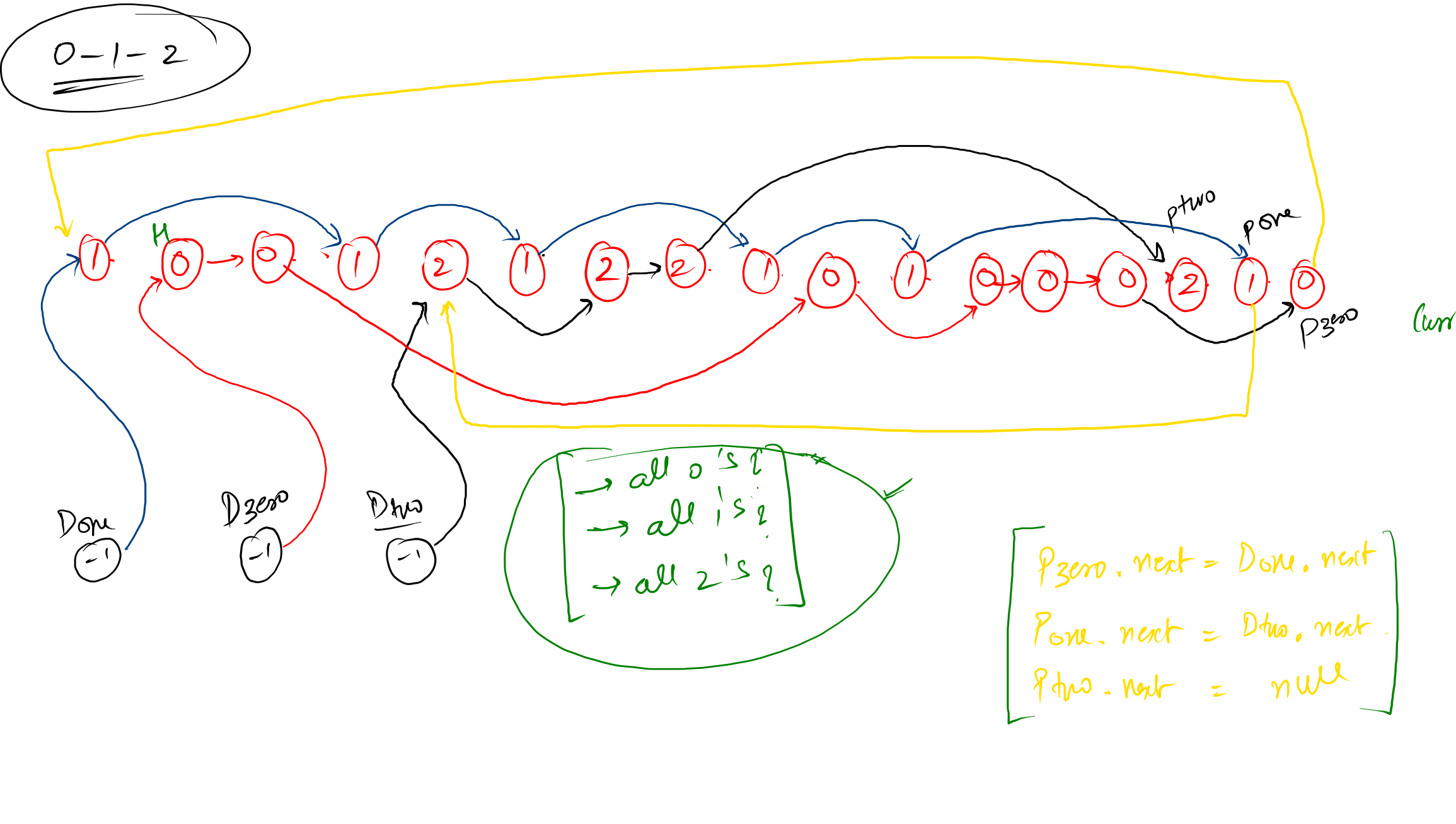
```
1->0->1->0->0->1->1->1->1->1->1->null
```

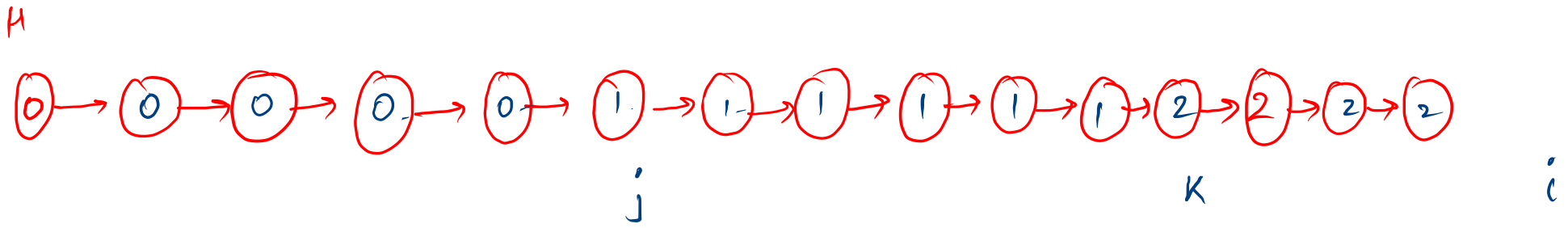





```
while (i != null) {
    if (i.val == 1) {
        i = i.next;
    } else {
        Swap
        i = i.next;
        j = j.next;
    }
}
```



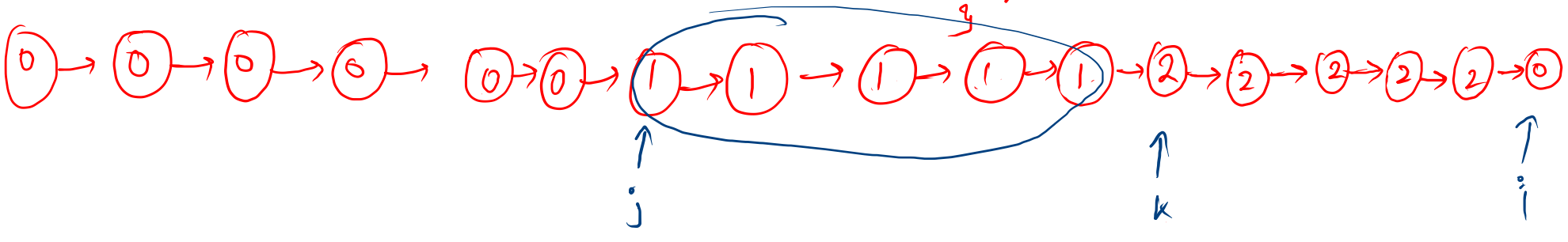

0-1-2
H
ptwo
pone
pzero
Curr
Done
-1
Dzero
-1
Dtwo
-1
→ all 0's ?
→ all 1's ?
→ all 2's ?
Pzero . next = Done . next
Pone . next = Dtwo . next
Ptwo . next = null

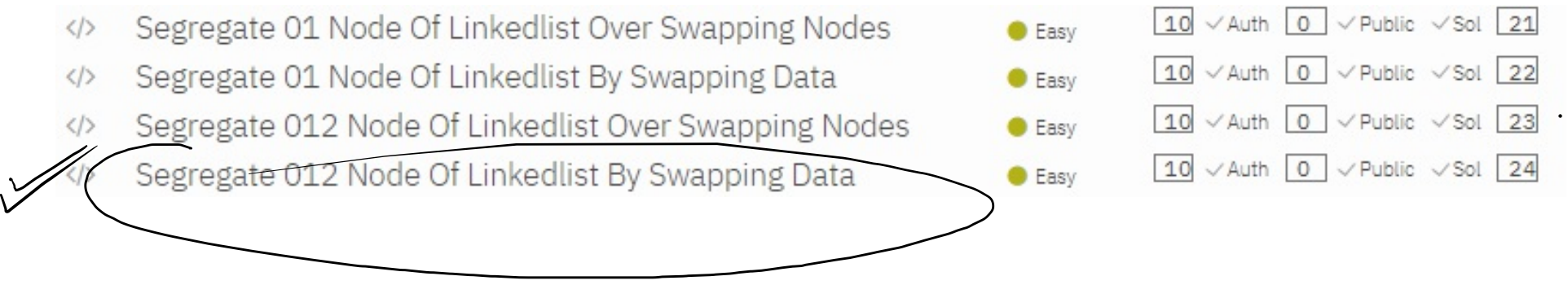


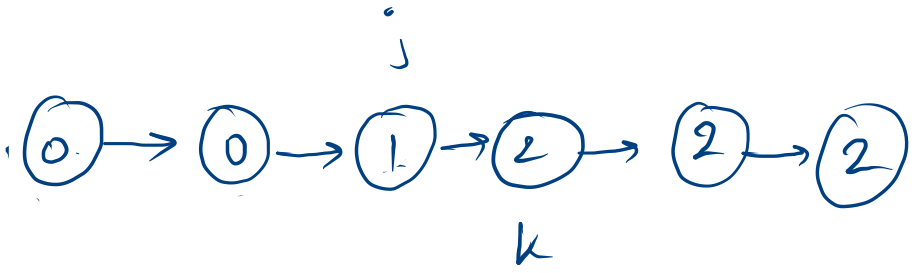

```
if(i.val==2){
    i=i.next;
}
```

```
if(i.val==1){
    swap(i,k);
    i=i.next;
    k=k.next;
}
```

```
if(i.val==0){
    swap(i,j);
    swap(i,k);
    i=i.next;
    j=j.next
    k=k.next;
```

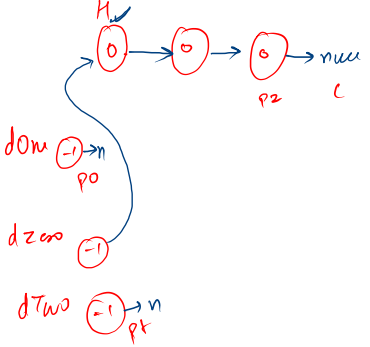


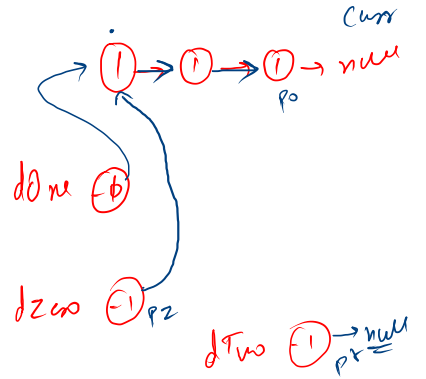

Segregate 01 Node Of Linkedlist Over Swapping Nodes
Easy
10 Auth 0 Public Sol 21
Segregate 01 Node Of Linkedlist By Swapping Data
Easy
10 Auth 0 Public Sol 22
Segregate 012 Node Of Linkedlist Over Swapping Nodes
Easy
10 Auth 0 Public Sol 23
Segregate 012 Node Of Linkedlist By Swapping Data
Easy
10 Auth 0 Public Sol 24

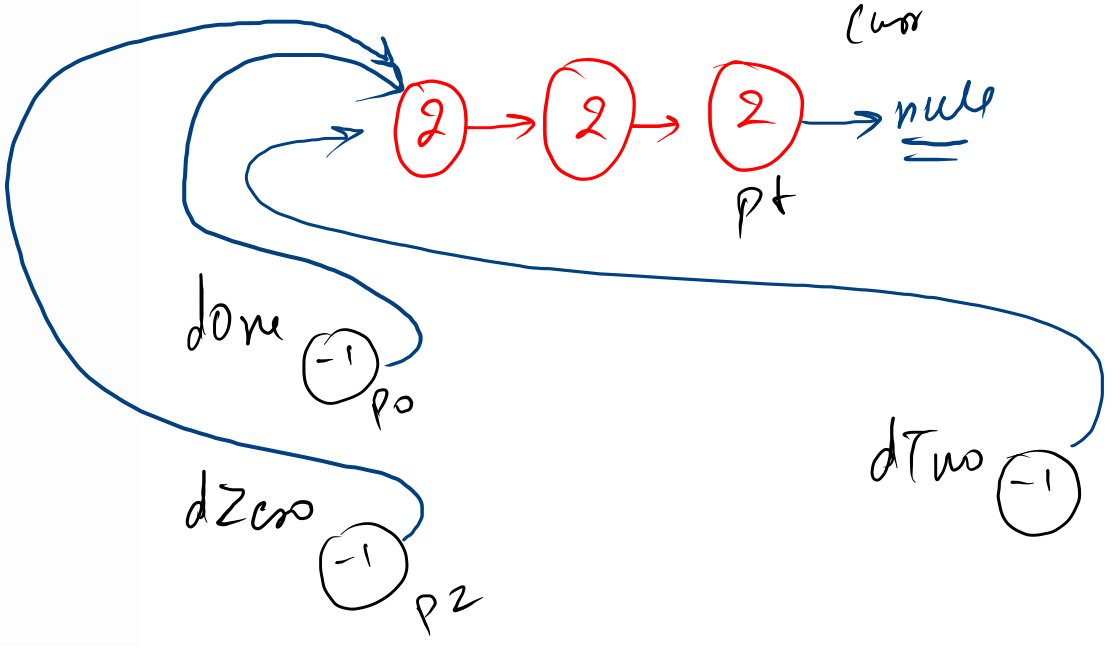



```
ListNode i = head , j = head , k = head;

while(i != null){
    if(i.val == 2){
        i = i.next;
    }else if(i.val == 1){
        swap(i,k);
        i = i.next;
        k = k.next;
    }else{
        swap(i,j);
        swap(i,k);
        i = i.next;
        k = k.next;
        j = j.next;
    }
}

return head;
```

```java
public static ListNode segregate012(ListNode head) {
    if(head == null || head.next == null) return head;

    ListNode dOne = new ListNode(-1);
    ListNode dZero = new ListNode(-1);
    ListNode dTwo = new ListNode(-1);
    ListNode po = dOne , pz = dZero , pt = dTwo , curr = head;

    while(curr != null){
        if(curr.val == 1){
            po.next = curr;
            po = po.next;
        }else if(curr.val == 0){
            pz.next = curr;
            pz = pz.next;
        }else{
            pt.next = curr;
            pt = pt.next;
        }
        curr = curr.next;
    }

    pz.next = dOne.next;
    po.next = dTwo.next;
    pt.next = null;

    return dZero.next;
}
```





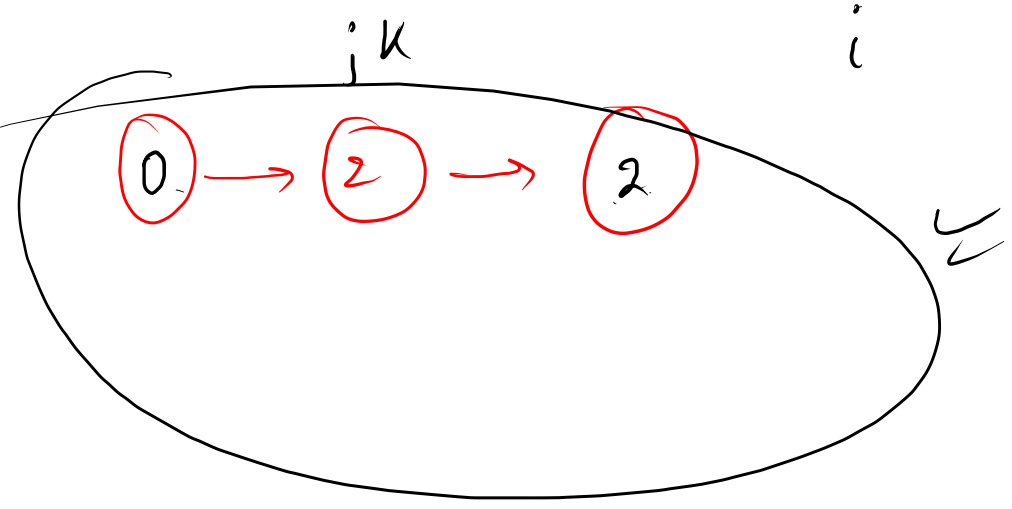


Node Swap

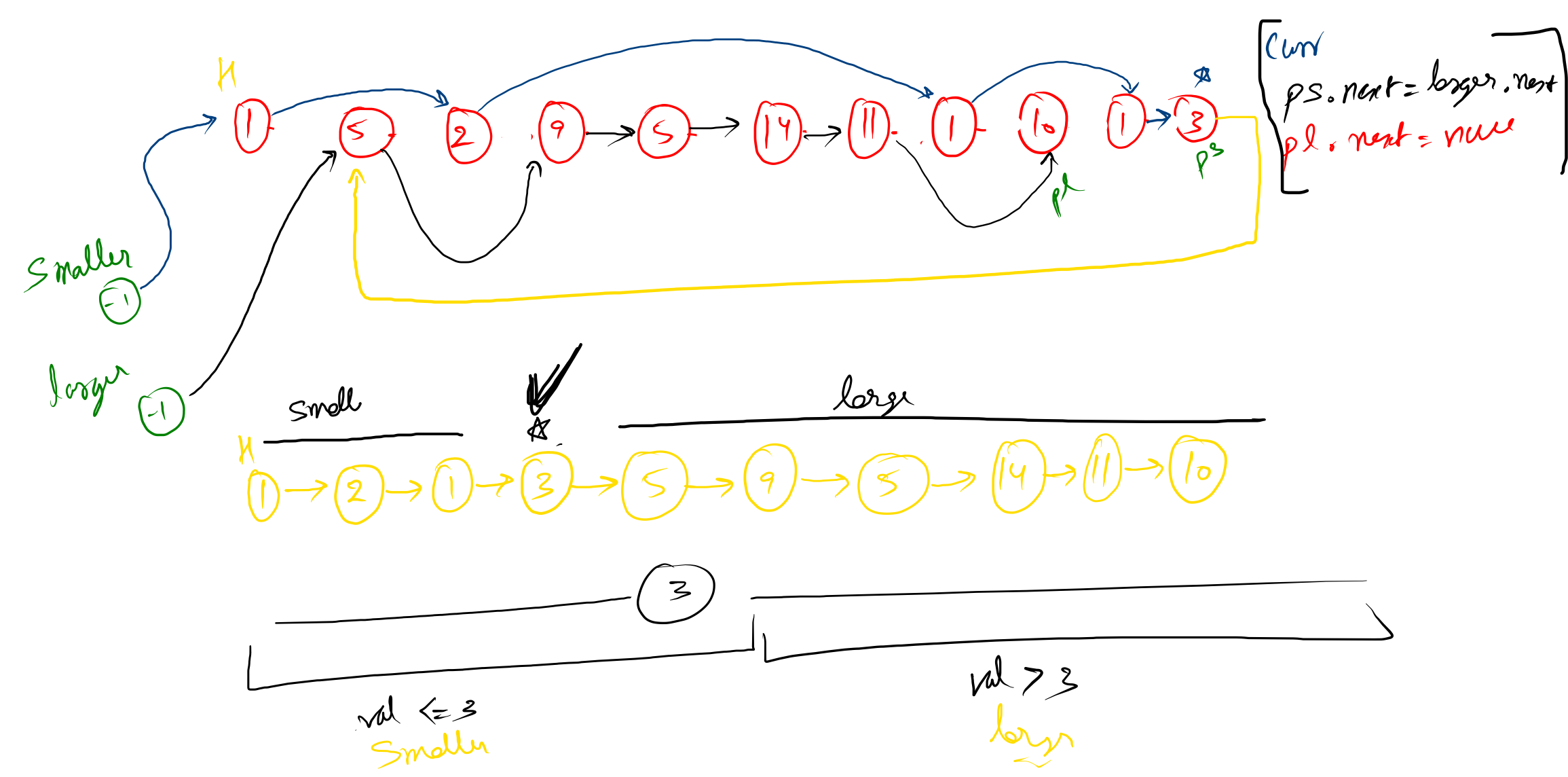
```java
public static ListNode segregate012(ListNode head) {
    if(head == null || head.next == null) return head;

    ListNode dOne = new ListNode(-1);
    ListNode dZero = new ListNode(-1);
    ListNode dTwo = new ListNode(-1);
    ListNode po = dOne , pz = dZero , pt = dTwo , curr = head;

    while(curr != null){
        if(curr.val == 1){
            po.next = curr;
            po = po.next;
        }else if(curr.val == 0){
            pz.next = curr;
            pz = pz.next;
        }else{
            pt.next = curr;
            pt = pt.next;
        }
        curr = curr.next;
    }

    po.next = dTwo.next;
    pz.next = dOne.next;
    pt.next = null;

    return dZero.next;
}
```

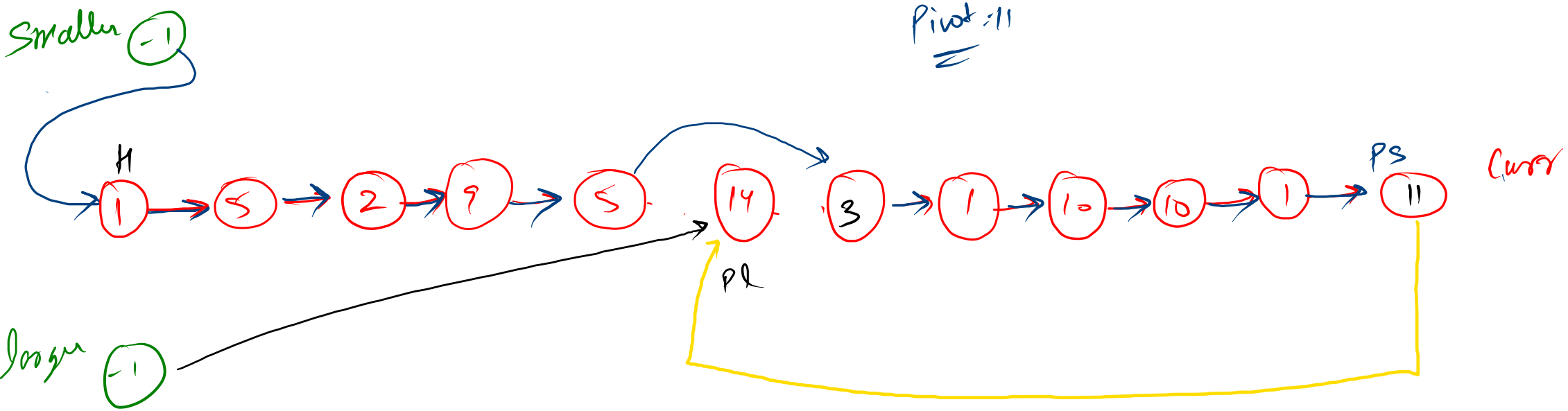
Data Swap

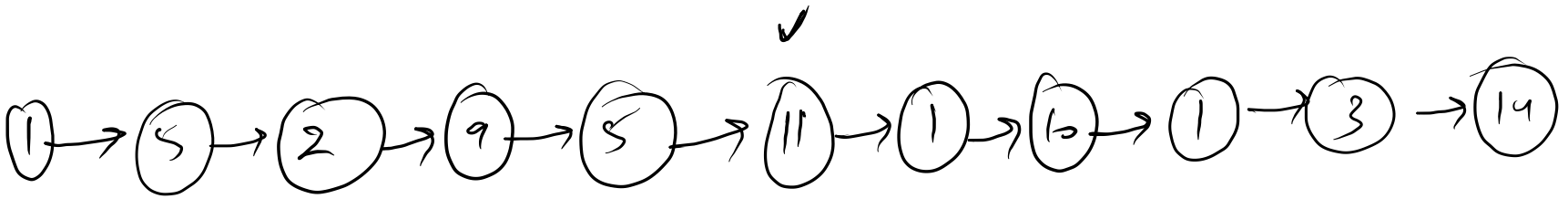
```java
public static ListNode segregate012(ListNode head) {

    ListNode i = head , j = head , k = head;

    while(i != null){
        if(i.val == 2){
            i = i.next;
        }else if(i.val == 1){
            swap(i,k);
            i = i.next;
            k = k.next;
        }else{
            swap(i,j);
            if(i.val == 1){
                swap(i,k);
            }
            i = i.next;
            k = k.next;
            j = j.next;
        }
    }

    return head;
}
```

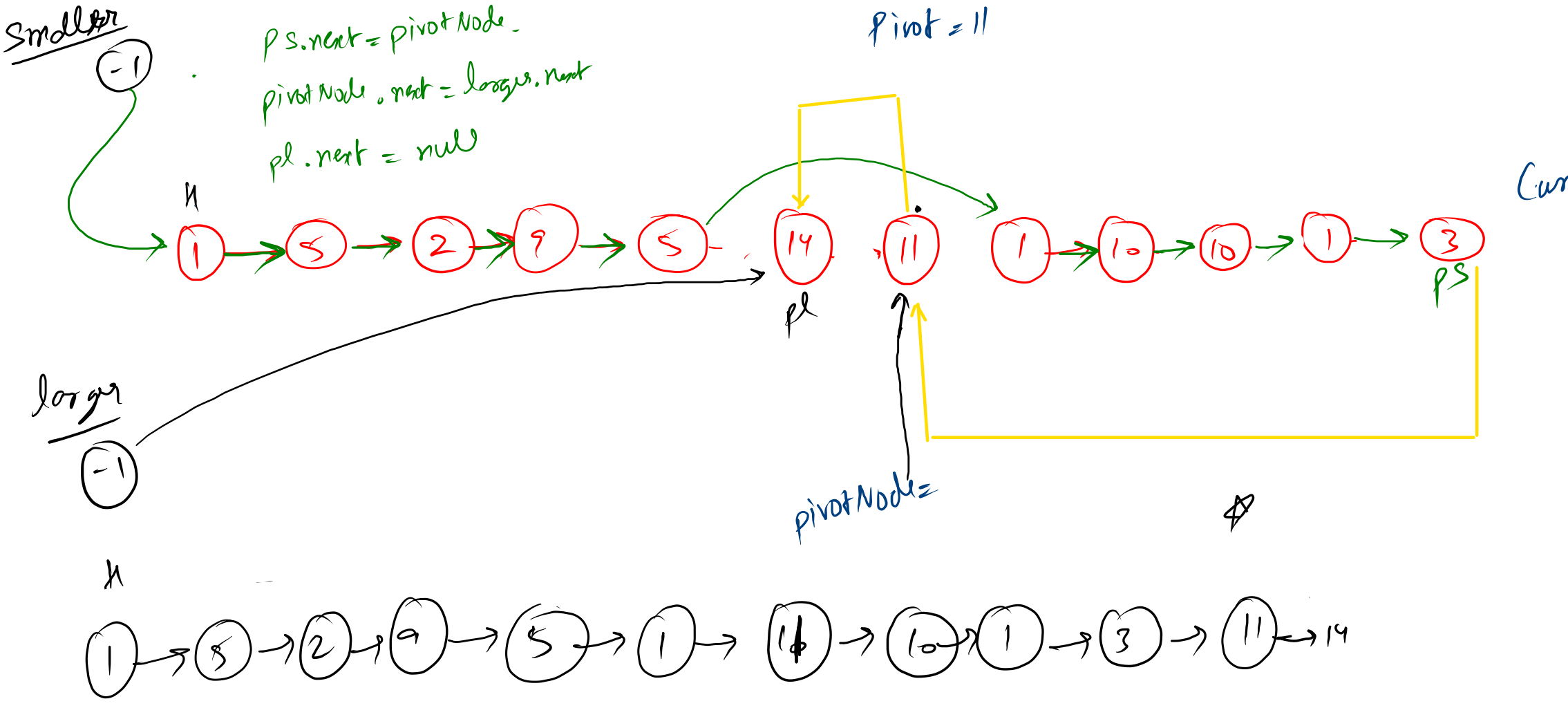



```
ListNode i = head , j = head , k = head;

while(i != null){
    if(i.val == 2){
        i = i.next;
    }else if(i.val == 1){
        swap(i,k);
        i = i.next;
        k = k.next;
    }else{
        swap(i,j);
        if(i.val == 1){
            swap(i,k);
        }
        i = i.next;
        k = k.next;
        j = j.next;
    }
}

return head;
```

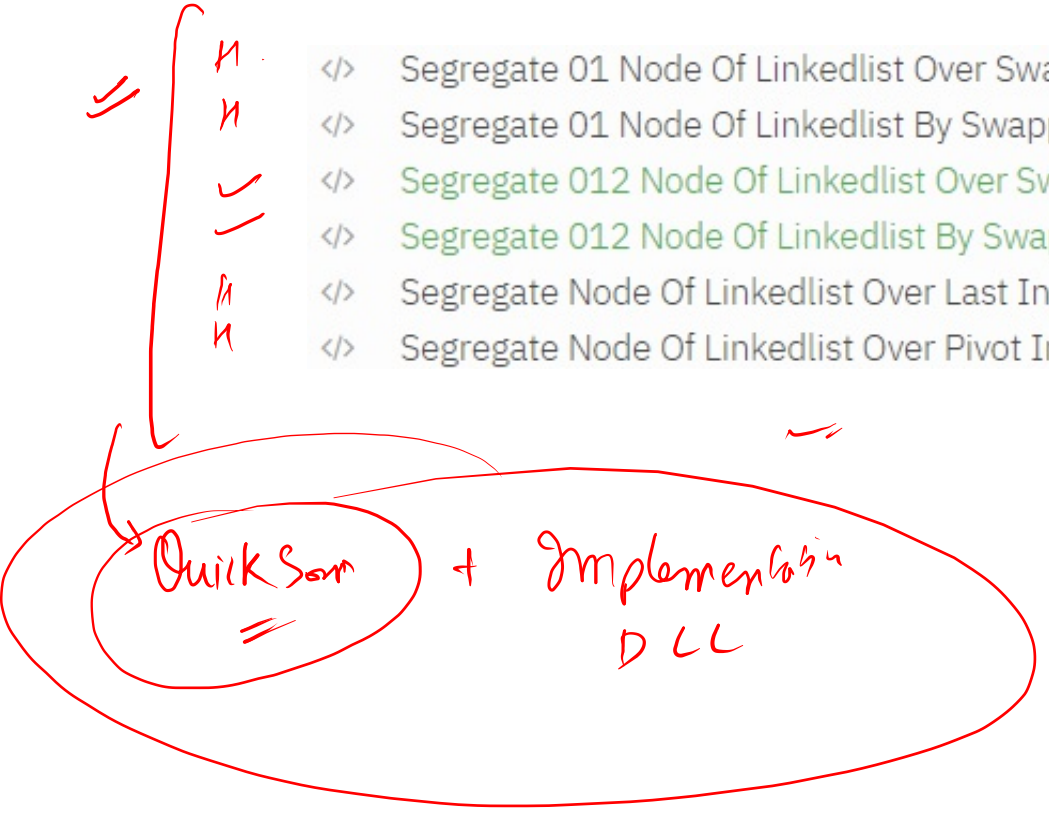

H
1
5
2
9
5
14
11
1
10
1
3
ps
pl
Curr
ps.next = larger.next
pl.next = null
Smaller
-1
larger
-1
small
larger
H
1
2
1
3
5
9
5
14
11
10
3
val <= 3
smaller
val > 3
larger

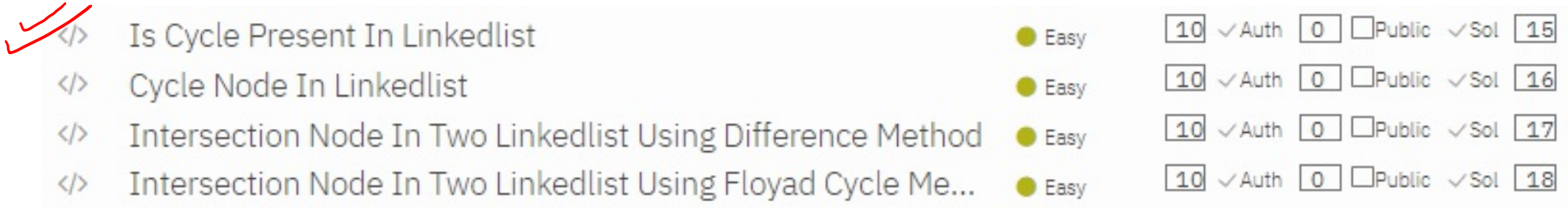

Smaller
-1
Pivot :11
H
1
5
2
9
5
14
3
1
10
10
1
PS
11
Curr
PL
larger
-1

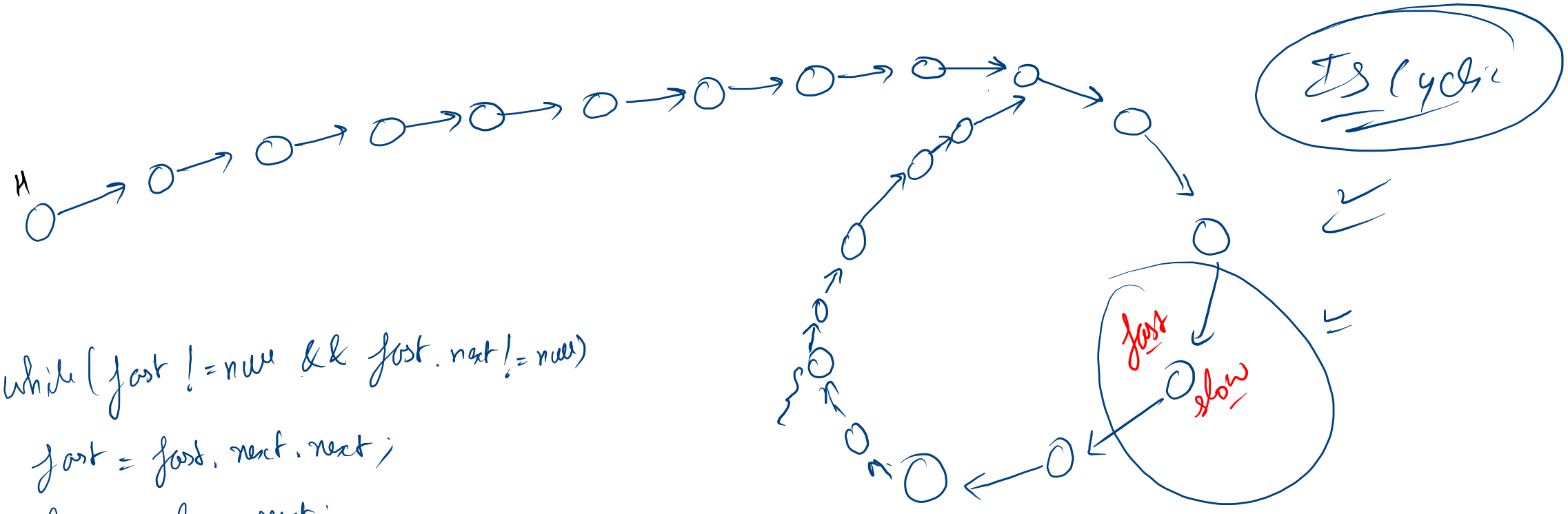

1
5
2
9
5
11
1
10
1
3
14


Smaller
-1
ps.next = pivotNode.
pivotNode.next = larger.next
pl.next = null
pivot = 11
H
1
5
2
9
5
14
11
1
10
10
1
3
ps
pl
Curr
larger
-1
pivotNode=
H
1
5
2
9
5
1
14
10
1
3
11
14

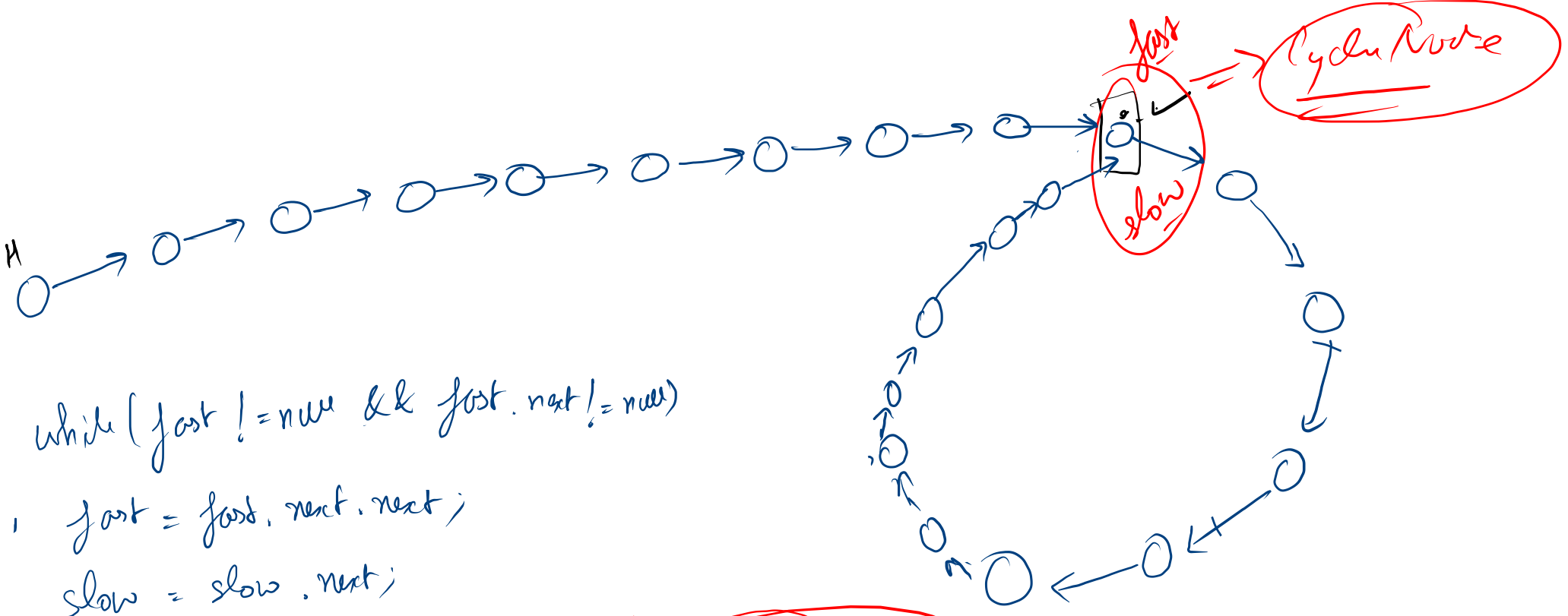
</>	Segregate 01 Node Of Linkedlist Over Swapping Nodes	Easy	10	Auth	0	Public	Sol	21
</>	Segregate 01 Node Of Linkedlist By Swapping Data	Easy	10	Auth	0	Public	Sol	22
</>	Segregate 012 Node Of Linkedlist Over Swapping Nodes	Easy	10	Auth	0	Public	Sol	23
</>	Segregate 012 Node Of Linkedlist By Swapping Data	Easy	10	Auth	0	Public	Sol	24
</>	Segregate Node Of Linkedlist Over Last Index.	Easy	10	Auth	0	Public	Sol	25
</>	Segregate Node Of Linkedlist Over Pivot Index	Medium	10	Auth	0	Public	Sol	26




Is Cycle Present In Linkedlist
Easy
10 Auth 0 Public Sol 15
Cycle Node In Linkedlist
Easy
10 Auth 0 Public Sol 16
Intersection Node In Two Linkedlist Using Difference Method
Easy
10 Auth 0 Public Sol 17
Intersection Node In Two Linkedlist Using Floyad Cycle Me...
Easy
10 Auth 0 Public Sol 18

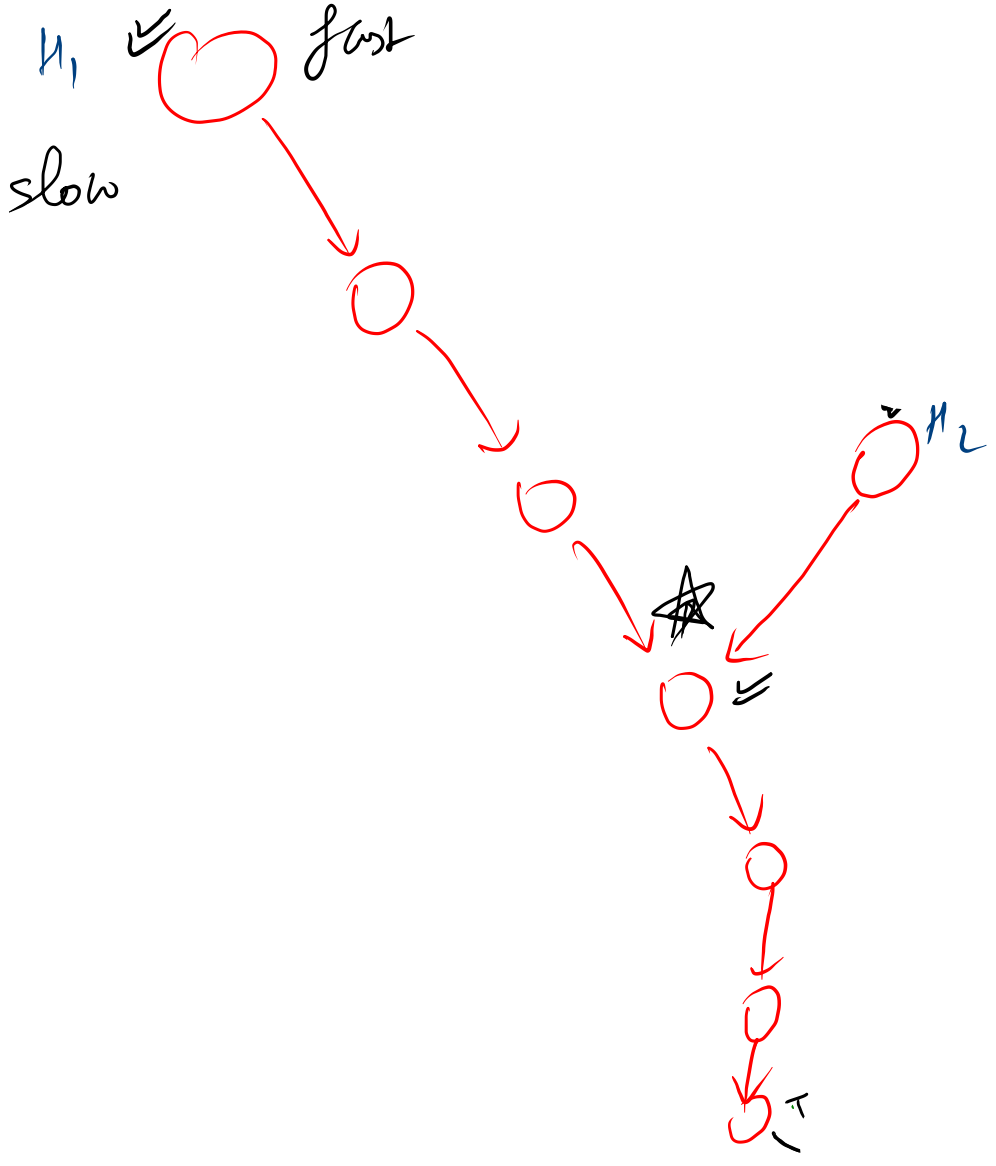


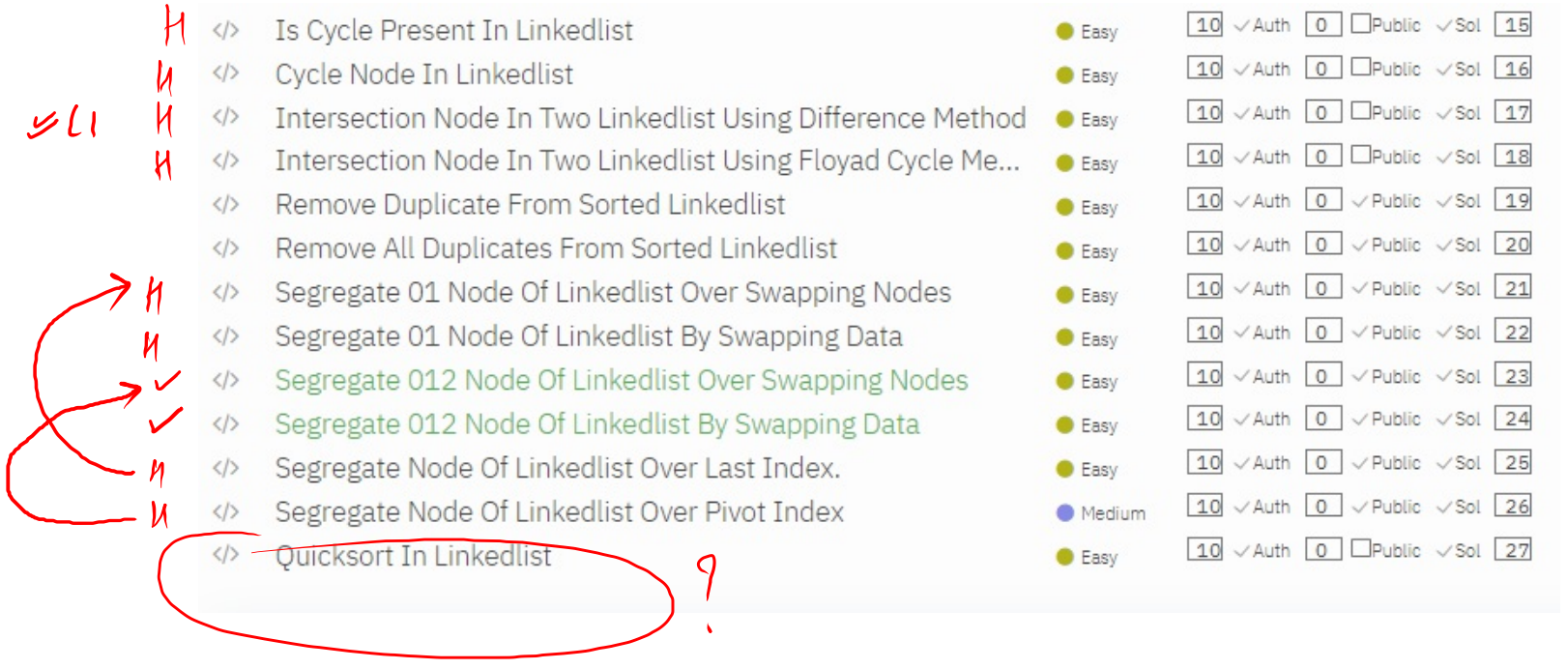

```
while (fast != null && fast.next != null)
    fast = fast.next.next;
    slow = slow.next;
    if (fast == slow) {
        return true;
    }
}
return false;
```




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

if (fast != slow) {
    return null
}
```

fast = head ;


H_1
fast
slow
H_2



	Is Cycle Present In Linkedlist	Easy	10	Auth	0	Public	Sol	15
	Cycle Node In Linkedlist	Easy	10	Auth	0	Public	Sol	16
	Intersection Node In Two Linkedlist Using Difference Method	Easy	10	Auth	0	Public	Sol	17
	Intersection Node In Two Linkedlist Using Floyad Cycle Me...	Easy	10	Auth	0	Public	Sol	18
	Remove Duplicate From Sorted Linkedlist	Easy	10	Auth	0	Public	Sol	19
	Remove All Duplicates From Sorted Linkedlist	Easy	10	Auth	0	Public	Sol	20
	Segregate 01 Node Of Linkedlist Over Swapping Nodes	Easy	10	Auth	0	Public	Sol	21
	Segregate 01 Node Of Linkedlist By Swapping Data	Easy	10	Auth	0	Public	Sol	22
	Segregate 012 Node Of Linkedlist Over Swapping Nodes	Easy	10	Auth	0	Public	Sol	23
	Segregate 012 Node Of Linkedlist By Swapping Data	Easy	10	Auth	0	Public	Sol	24
	Segregate Node Of Linkedlist Over Last Index.	Easy	10	Auth	0	Public	Sol	25
	Segregate Node Of Linkedlist Over Pivot Index	Medium	10	Auth	0	Public	Sol	26
	Quicksort In Linkedlist	Easy	10	Auth	0	Public	Sol	27