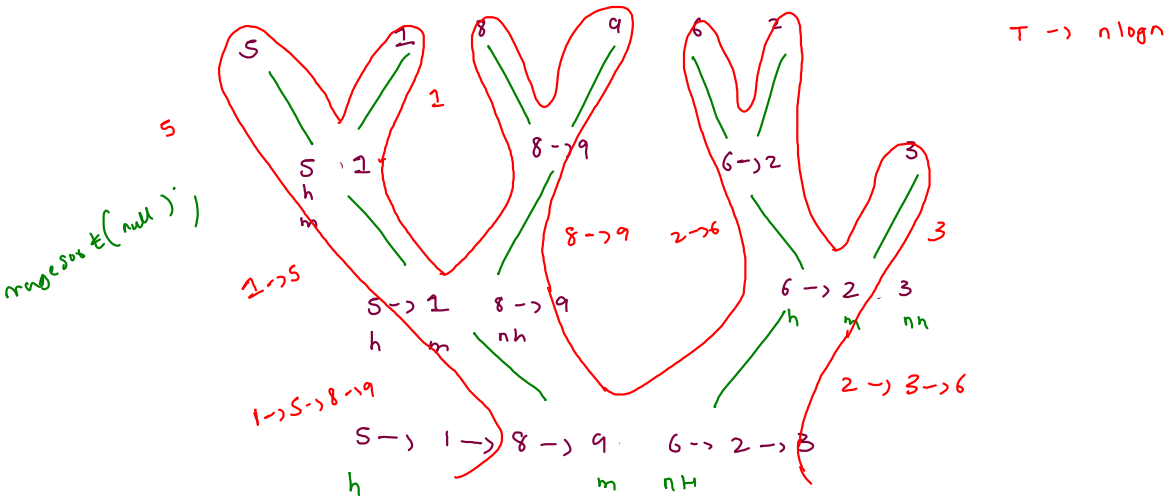
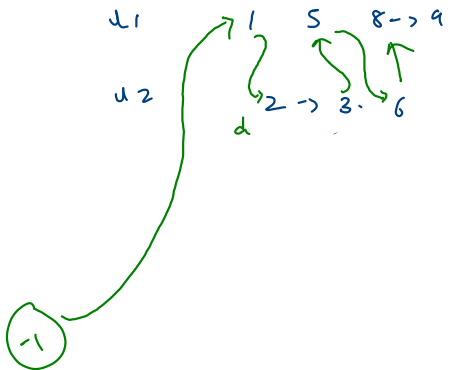

T -> nlogn
5
1
8
9
6
2
3
5
1
8->9
6->2
3
5
1
8->9
6->2
3
5->1
8->9
6->2
3
h
m
nh
1->5->8->9
5->1->8->9->6->2->3
h
m
nH
2->3->6


1
5
8->9
2->3->6
-1

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

    ListNode mid = middleOfLL(head);
    ListNode nH = mid.next;
    mid.next = null;

    ListNode left = mergeSort(head);
    ListNode right = mergeSort(nH);

    ListNode ml = mergeTwoSortedLL(left,right);
    return ml;
}
```

k

$T(n) = 2T\left(\frac{n}{2}\right) + n$ — (1)

$2T\left(\frac{n}{2}\right) = 4T\left(\frac{n}{4}\right) + n$ — (2) $\times 2$

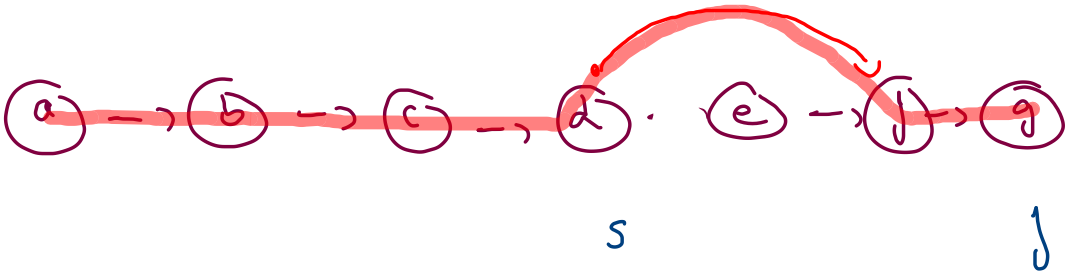
$4T\left(\frac{n}{4}\right) = 8T\left(\frac{n}{8}\right) + n$ — (3) $\times 4$

$\vdots$

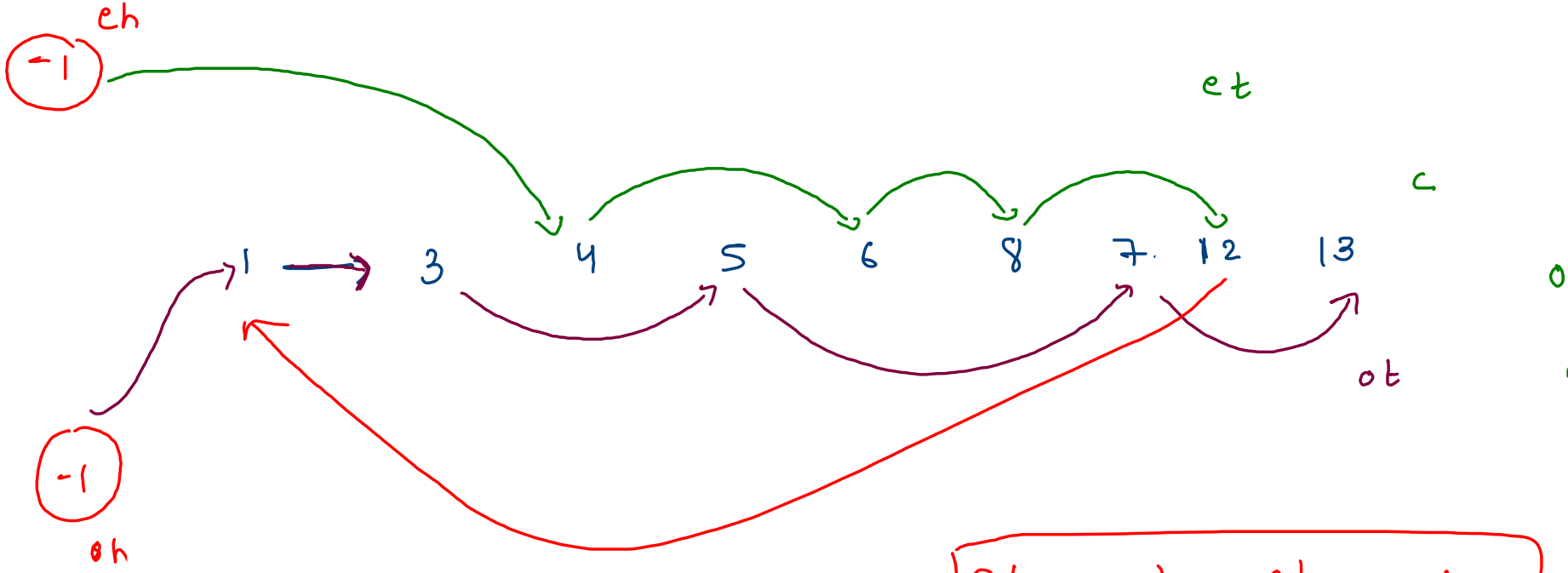
$T(1) = 0 + 0$

$T(n) = nk$

$= n \log n$



$k = 3$



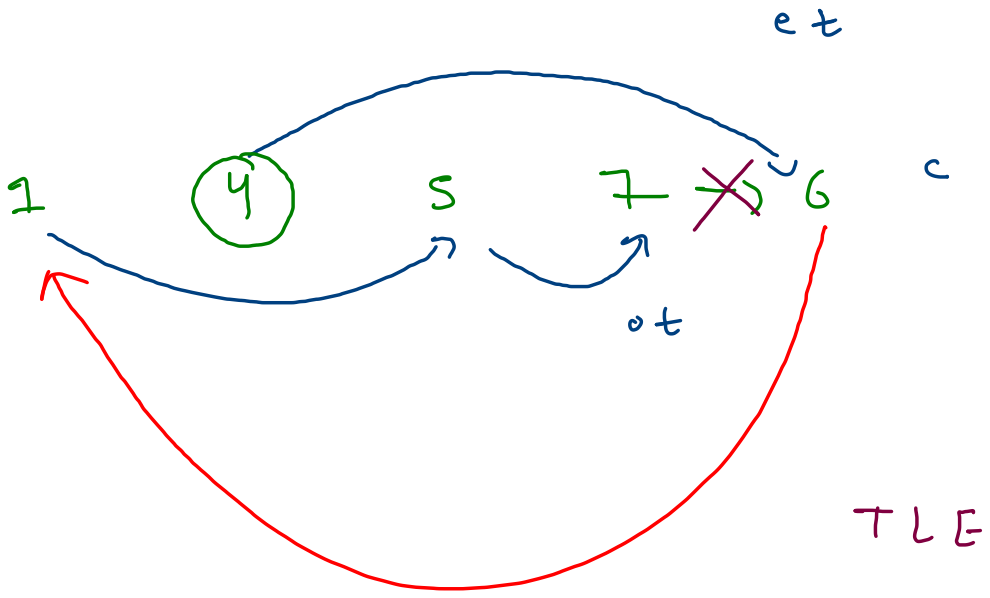
et.next = oh.next

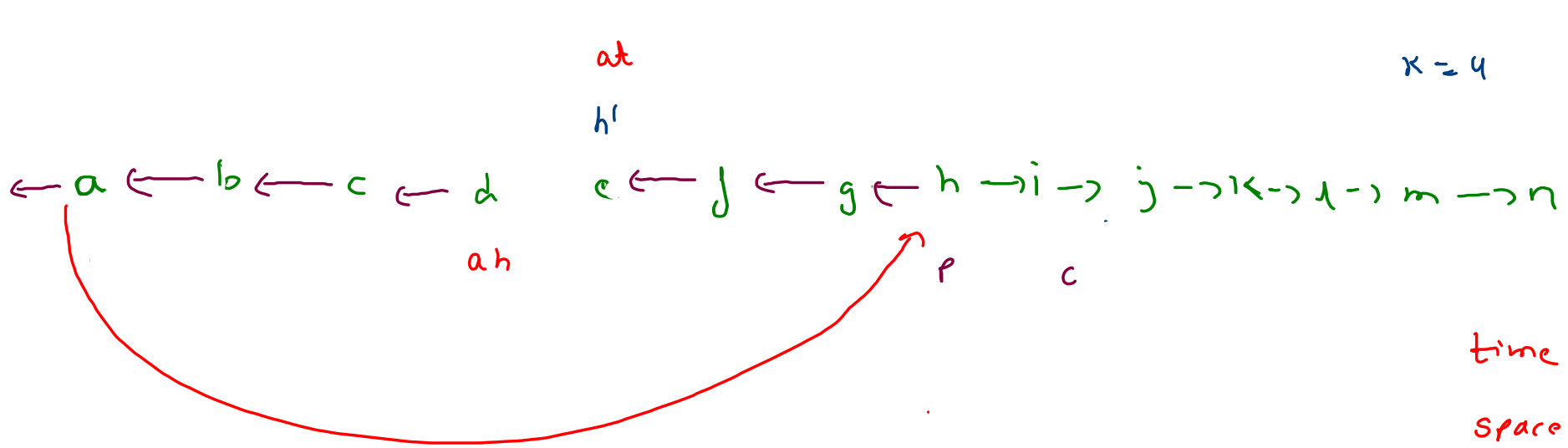
eh.next

O(n)

O(1)

```
while(curr != null) {
    if(curr.val % 2 == 0) {
        et.next = curr;
        et = et.next;
    }
    else {
        ot.next = curr;
        ot = ot.next;
    }
    curr = curr.next;
}

et.next = oh.next;

return eh.next;
```





time -> O(n)

space -> O(1)

at.next = p

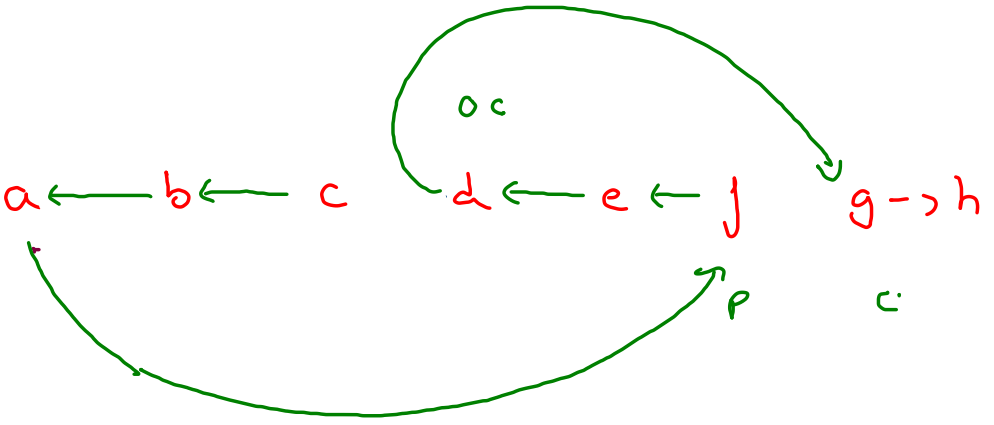
at = h'

nxsj = 6

```java
while(nrsf >= k) {
    //work for k nodes
    ListNode prev = null;
    ListNode oc = curr;

    int temp = k;
    while(temp-- > 0) {
        //preserve
        ListNode next = curr.next;

        //links
        curr.next = prev;

        //move
        prev = curr;
        curr = next;
    }

    if(ansH == null) {
        //this is first group
        ansH = prev;
        ansT = oc;
    }
    else {
        //make connection between ans so far & current group
        ansT.next = prev;
        ansT = oc;
    }

    nrsf -= k;
}


if(nrsf > 0) {
    ansT.next = curr;
}
```

nrsf = 2

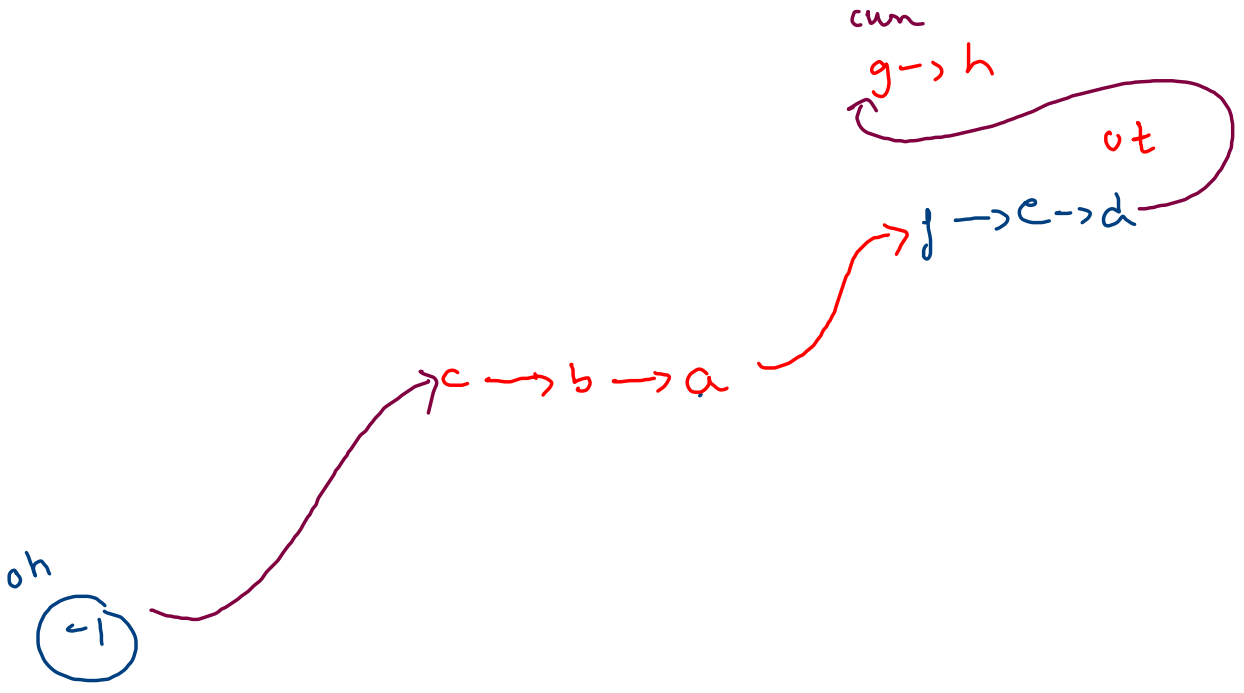
k = 3



aH = c

aT = d

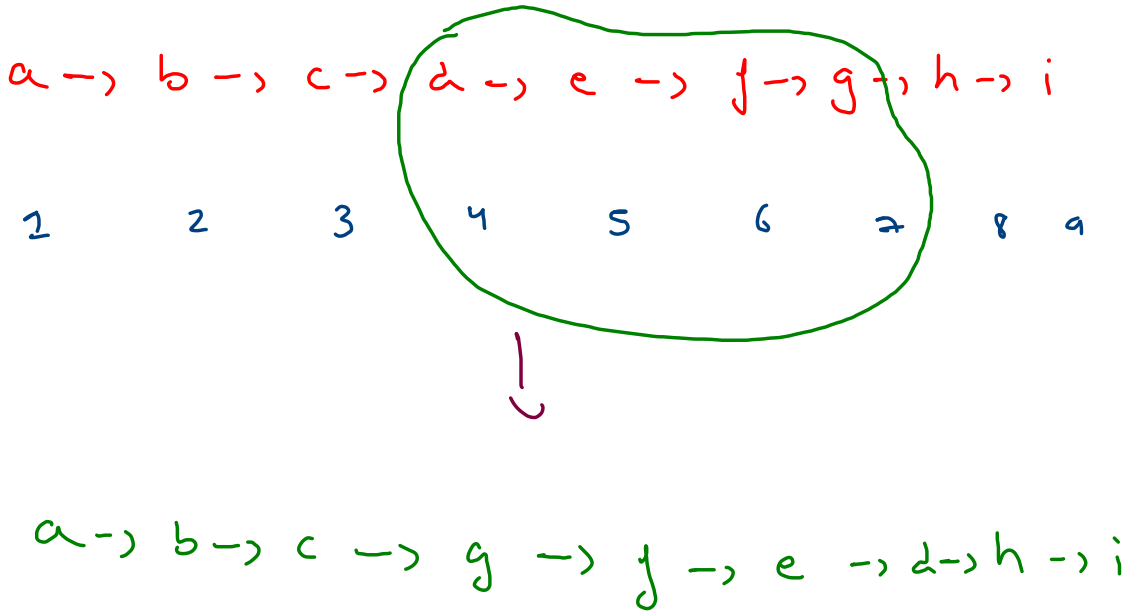
```java
while(nrsf >= k) {
    //work on k nodes
    int temp = k;
    ch = null;
    ct = null;

    while(temp-- > 0) {
        ListNode next = curr.next;

        curr.next = null;

        //addfirst
        addFirst(curr);

        curr = next;
    }

    if(oh == ot) {
        //this is first group
        oh.next = ch;
        ot = ct;
    }
    else {
        ot.next = ch;
        ot = ct;
    }

    nrsf -= k;
}

if(nrsf > 0) {
    ot.next = curr;
}
```



8->8->14->1->10->12->null
3
5

Output Format

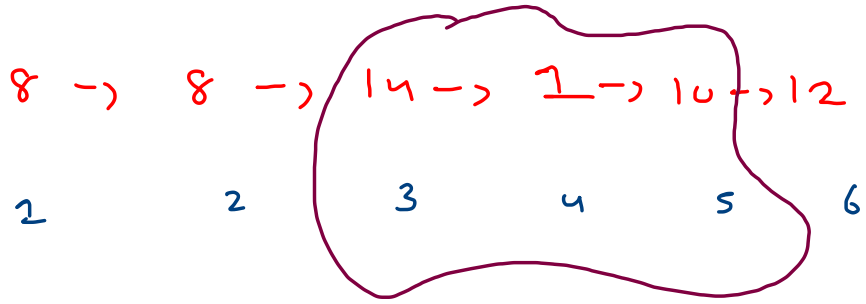
8->8->10->1->14->12->null



8->8->14->1->10->12->null
3
5

Output Format

8->8->10->1->14->12->null

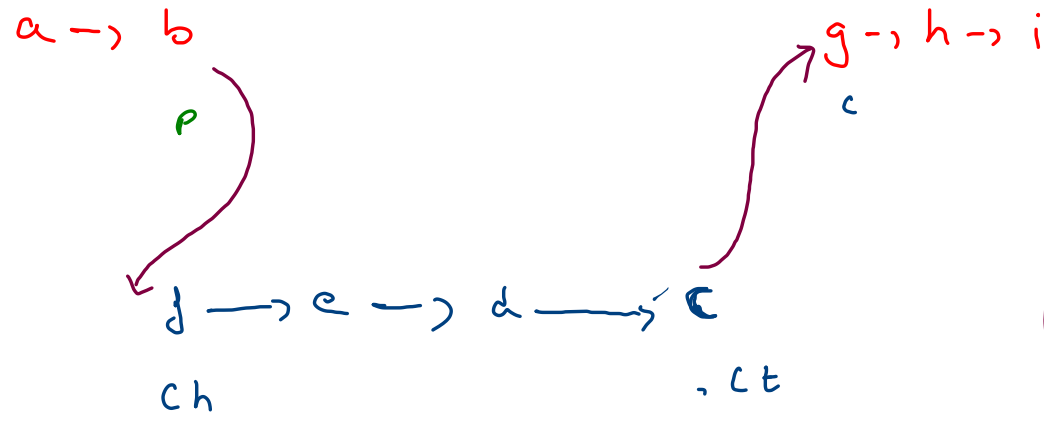


n = 3

m = 5

8 -> 8 -> 10 -> 1 -> 14 -> 12

1 2 3 4 5 6 7 8 9



p.next = ch

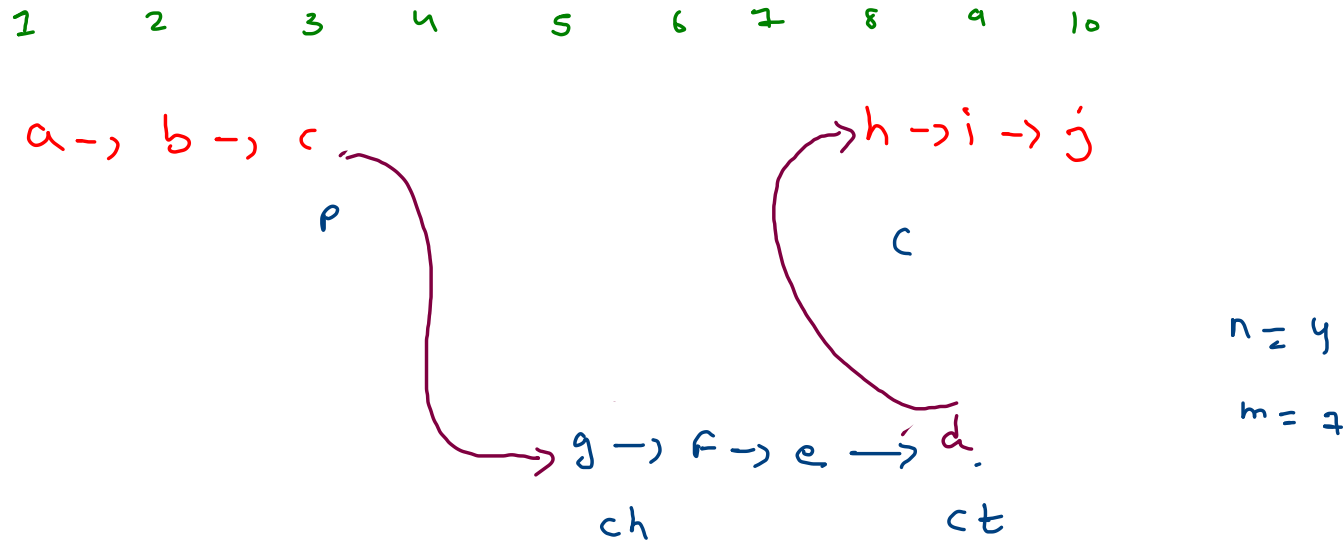
ct.next = c

n = 3

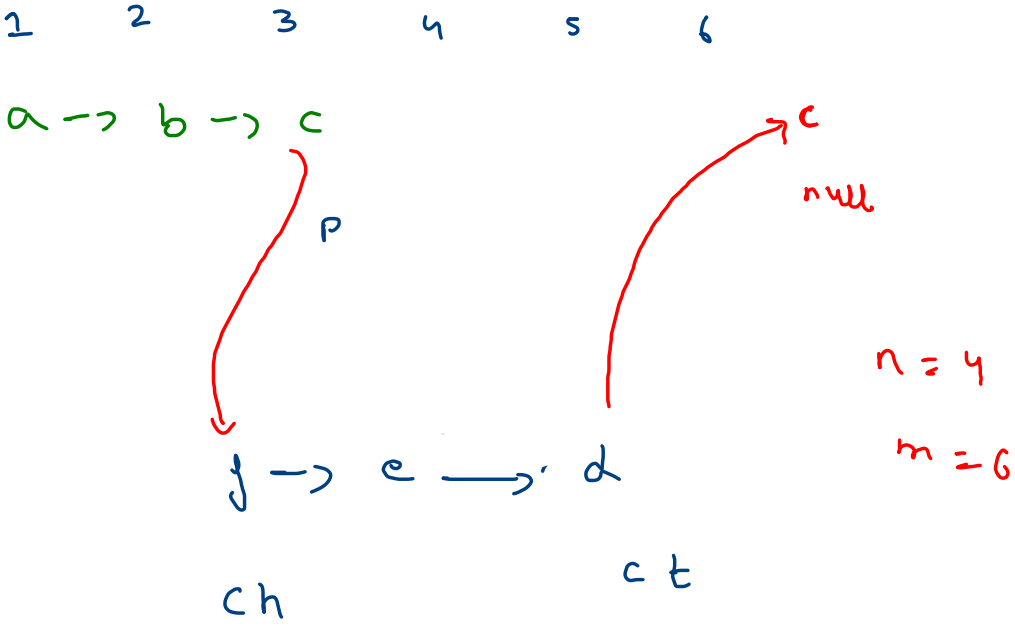
m = 6

i = ~~7~~ 2

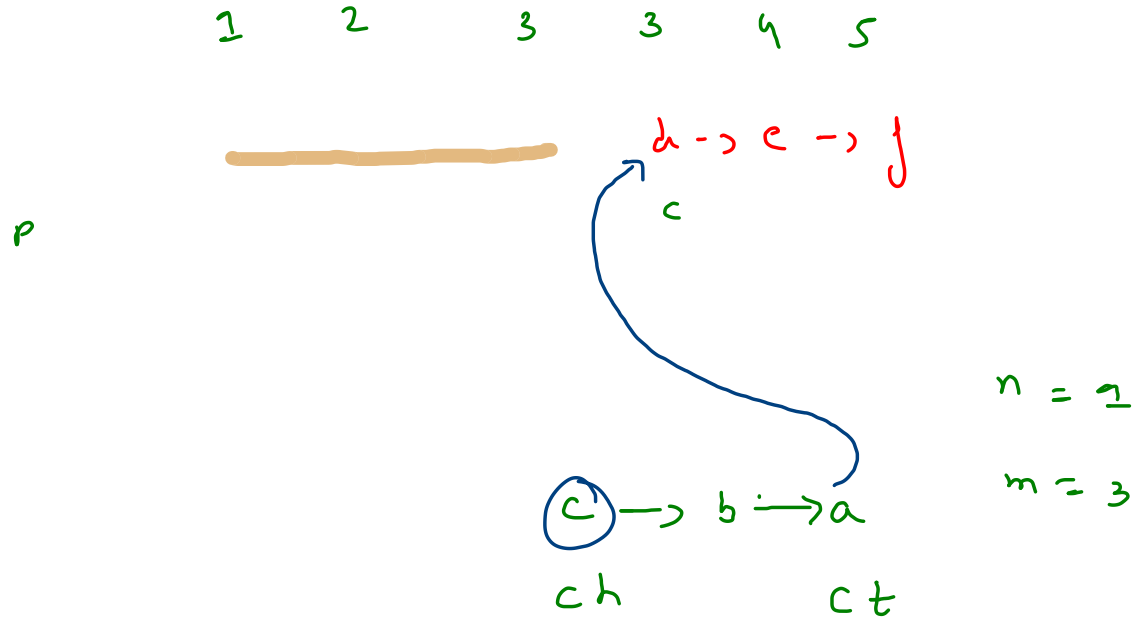
```java
ListNode prev = null;
ListNode curr = head;
int idx = 1;

while(curr != null) {
    if(idx < n) {
        //pre working area
        prev = curr;
        curr = curr.next;
    }
    else if(idx >= n && idx <= m) {
        //working area -> reverse
        ListNode next = curr.next;
        curr.next = null;
        addFirst(curr);

        curr = next;
    }
    else if(idx > m) {
        prev.next = ch;
        ct.next = curr;
        break;
    }
    idx++;

}

return head;
```



```java
ListNode prev = null;
ListNode curr = head;
int idx = 1;

while(curr != null) {
    if(idx < n) {
        //pre working area
        prev = curr;
        curr = curr.next;
    }
    else if(idx >= n && idx <= m) {
        //working area -> reverse
        ListNode next = curr.next;
        curr.next = null;
        addFirst(curr);

        curr = next;
    }
    else if(idx > m) {
        prev.next = ch;
        ct.next = curr;
        break;
    }
    idx++;

}

return head;
```



```
while(idx <= m+1) {
    if(idx < n) {
        //pre working area
        prev = curr;
        curr = curr.next;
    }
    else if(idx >= n && idx <= m) {
        //working area -> reverse
        ListNode next = curr.next;
        curr.next = null;
        addFirst(curr);

        curr = next;
    }
    else if(idx > m) {
        prev.next = ch;
        ct.next = curr;
        break;
    }
    idx++;

}
```



```
while(idx <= m+1) {
    if(idx < n) {
        //pre working area
        prev = curr;
        curr = curr.next;
    }
    else if(idx >= n && idx <= m) {
        //working area -> reverse
        ListNode next = curr.next;
        curr.next = null;
        addFirst(curr);

        curr = next;
    }
    else if(idx > m) {
        if(prev == null) {
            //when n == 1
            ct.next = curr;
            return ch;
        }

        prev.next = ch;
        ct.next = curr;
    }
    idx++;
}
```