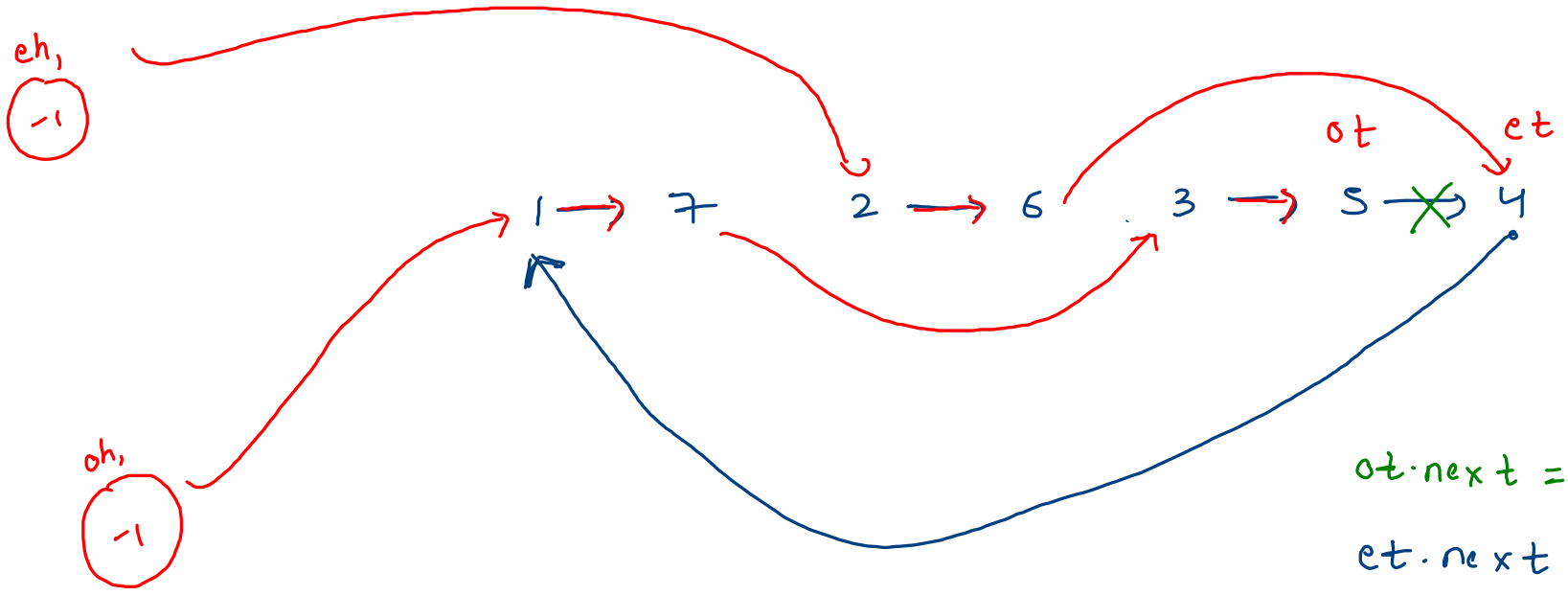
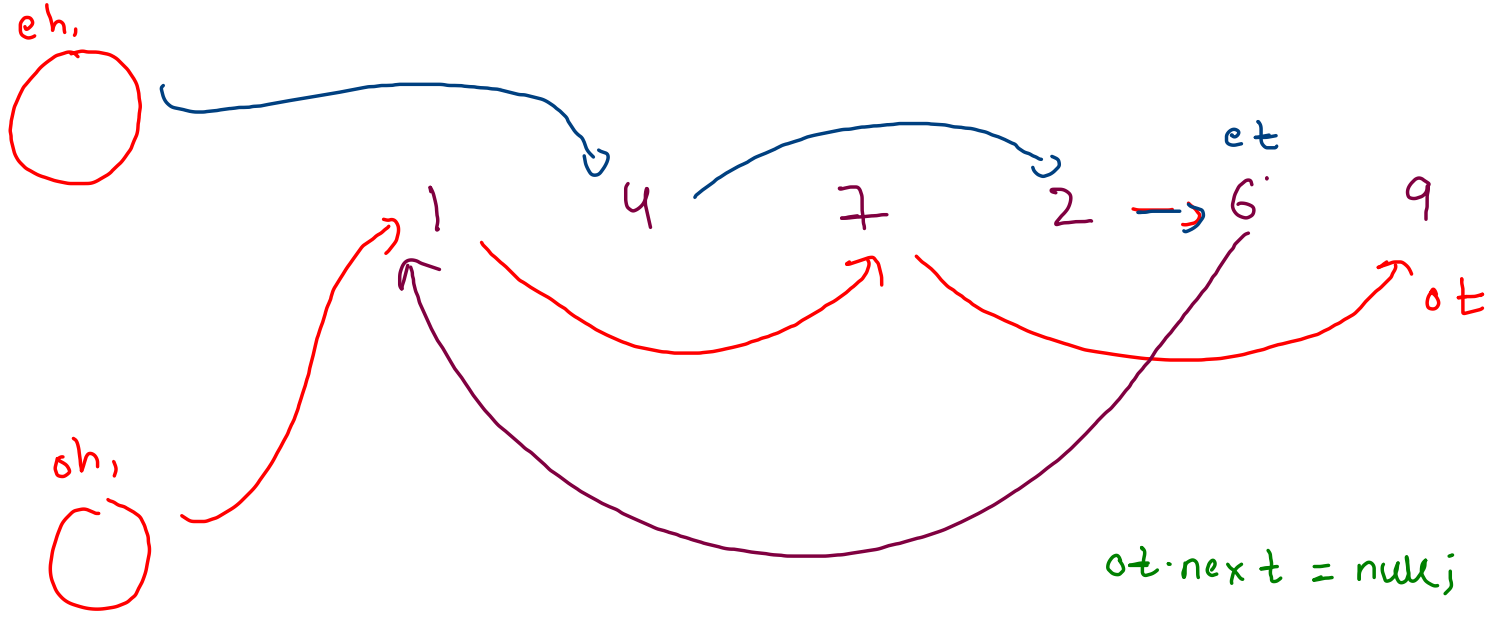
Segregate Even And Odd Nodes In A Linkedlist

1->7->2->6->3->5->4->null




```
ot.next = null;
et.next = oh.next
return eh.next;
```




```
ot.next = null;
et.next = oh.next
return eh.next;
```

25. Reverse Nodes in k-Group

$k = 3$

old: a → b → c → d → e → f → g → h

T: O(n)

k rev ll: c → b → a → f → e → d → g → h

S: O(1)

t

g -> h

ah at

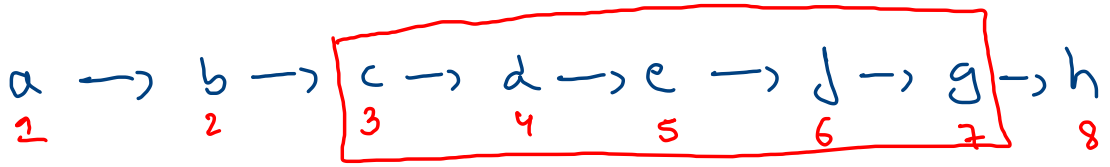
c -> b -> a -> j -> e -> d -> g -> h

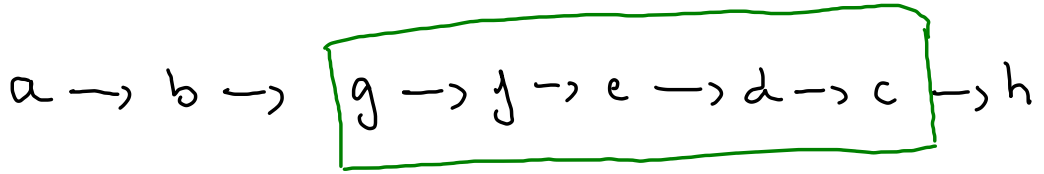
at.next = ch

at = et

92. Reverse Linked List II

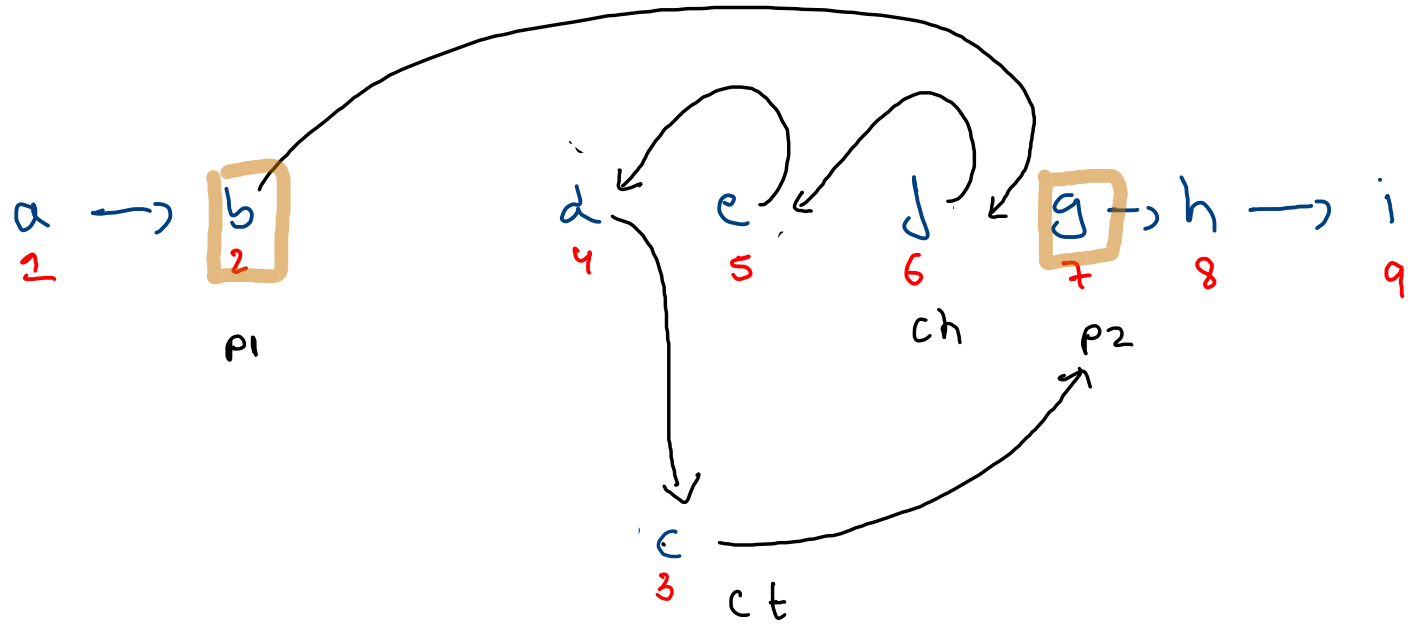
(reverse in range)





left = 3

right = 7

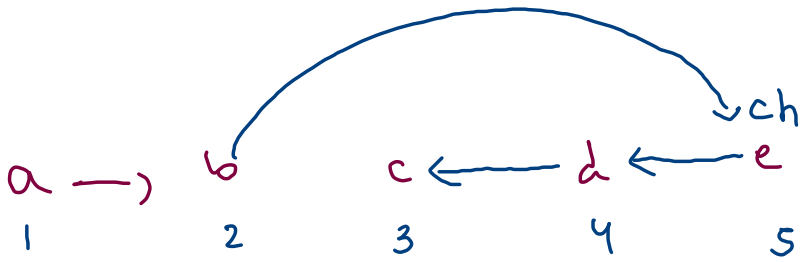


d = 3

r = 6

p1.next = ch

ct.next = p2



i = 3

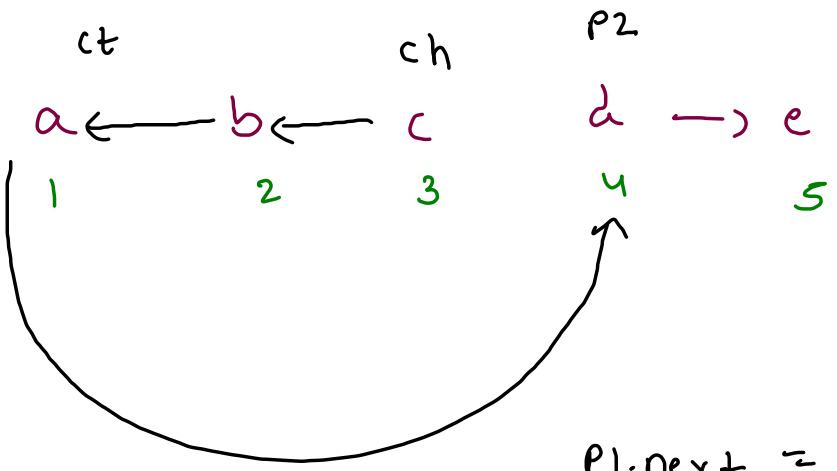
r = 5

p2 = null

p1.next = ch

ct.next = p2

p1 = null



l = 1

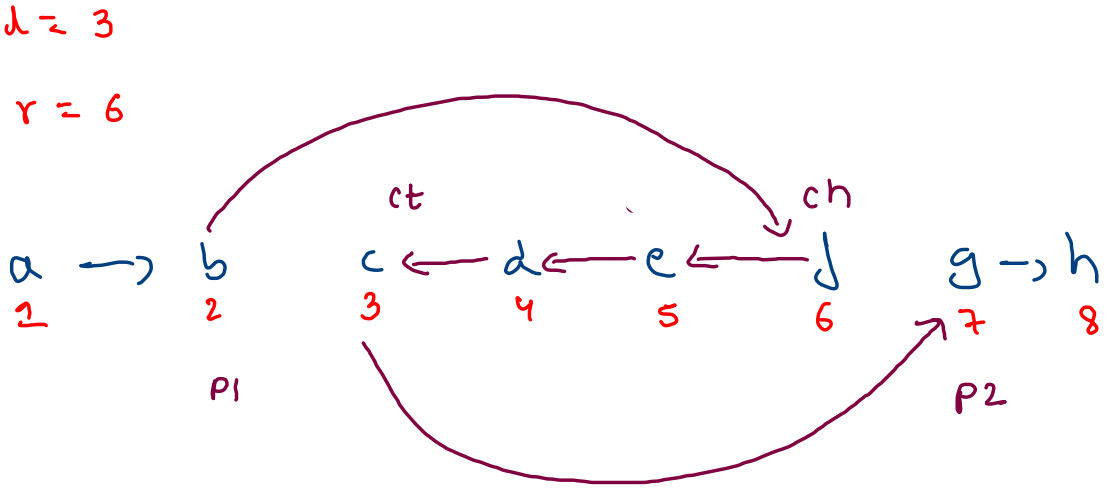
r = 3

p1.next = ch

ct.next = p2

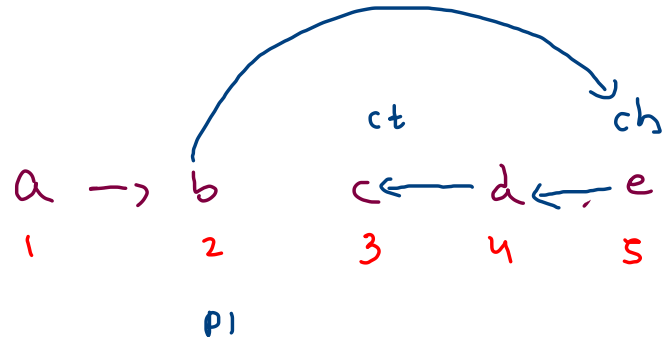
return ch;

```
while(temp != null) {
    //pre
    if(pos < left) {
        p1 = temp;
        temp = temp.next;
    }

    //work
    else if(pos >= left && pos <= right) {
        ListNode n = temp.next;

        //removefirst
        temp.next = null;

        //addfirst
        if(ch == null) {
            ch = ct = temp;
        }
        else {
            temp.next = ch;
            ch = temp;
        }

        temp = n;
    }

    //post
    else {
        p2 = temp;
        break;
    }

    pos++;
}
```




```
if(p1 != null) {
    p1.next = ch;
    ct.next = p2;

    return head;
}
else {
    ct.next = p2;
    return ch;
}
```

```
while(temp != null) {
    //pre
    if(pos < left) {
        p1 = temp;
        temp = temp.next;
    }

    //work
    else if(pos >= left && pos <= right) {
        ListNode n = temp.next;

        //removefirst
        temp.next = null;

        //addfirst
        if(ch == null) {
            ch = ct = temp;
        }
        else {
            temp.next = ch;
            ch = temp;
        }

        temp = n;
    }

    //post
    else {
        p2 = temp;
        break;
    }

    pos++;
}
```



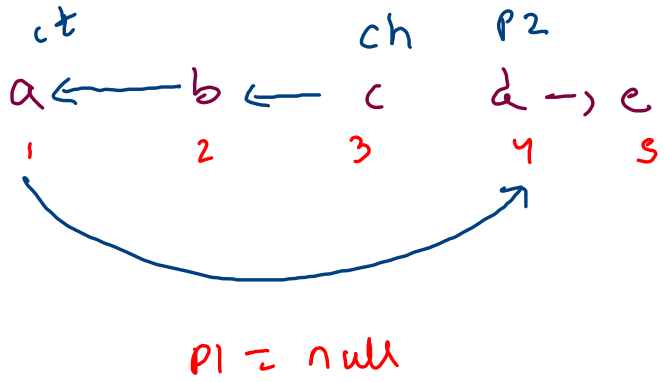
d = 3

r = 5

p2 = null

```
if(p1 != null) {
    p1.next = ch;
    ct.next = p2;

    return head;
}
else {
    ct.next = p2;
    return ch;
}
```

```
while(temp != null) {
    //pre
    if(pos < left) {
        p1 = temp;
        temp = temp.next;
    }

    //work
    else if(pos >= left && pos <= right) {
        ListNode n = temp.next;

        //removefirst
        temp.next = null;

        //addfirst
        if(ch == null) {
            ch = ct = temp;
        }
        else {
            temp.next = ch;
            ch = temp;
        }

        temp = n;
    }

    //post
    else {
        p2 = temp;
        break;
    }

    pos++;
}
```

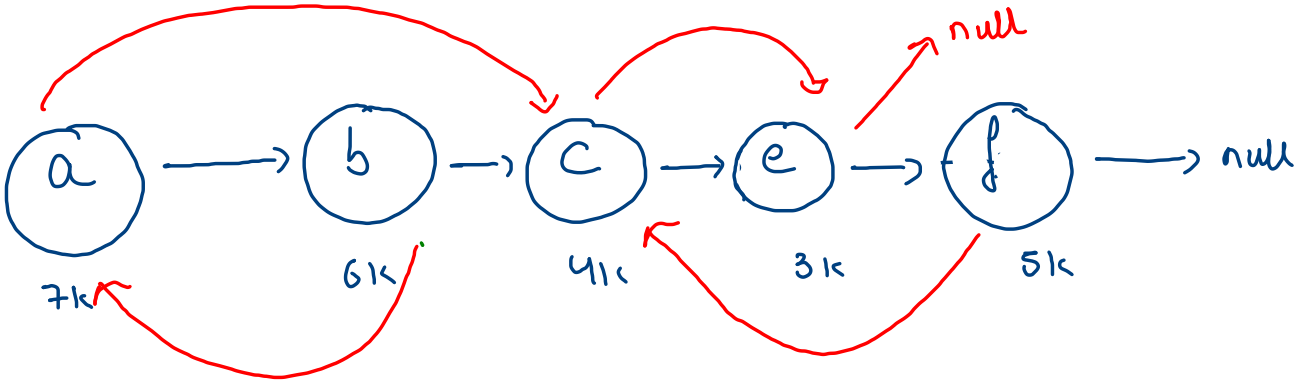


l = 1

r = 3

```
if(p1 != null) {
    p1.next = ch;
    ct.next = p2;

    return head;
}
else {
    ct.next = p2;
    return ch;
}
```

Copy Linkedlist With Random Pointers



Odd node vs new node

7k → 101k

61k → 191k

41k → 151k

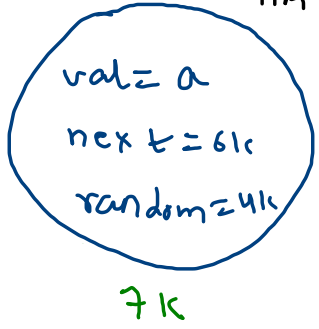
31k → 211k

51k → 161k

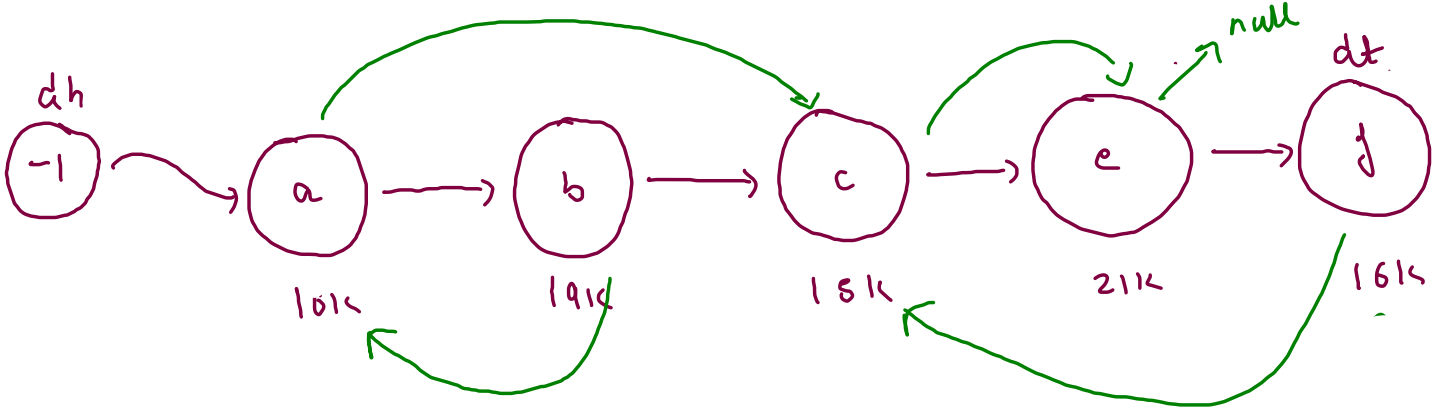
T : o(n)

S : o(n)

HM

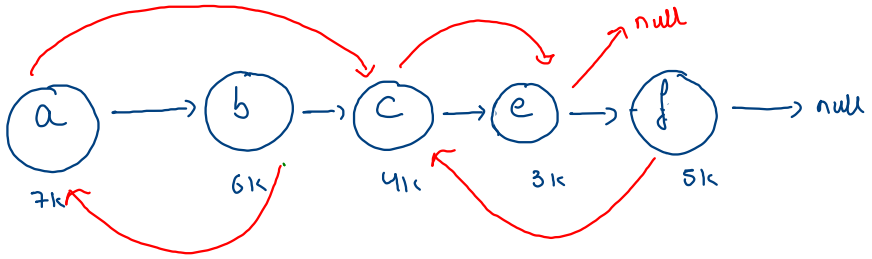


p2.random = map.get(p1.random);

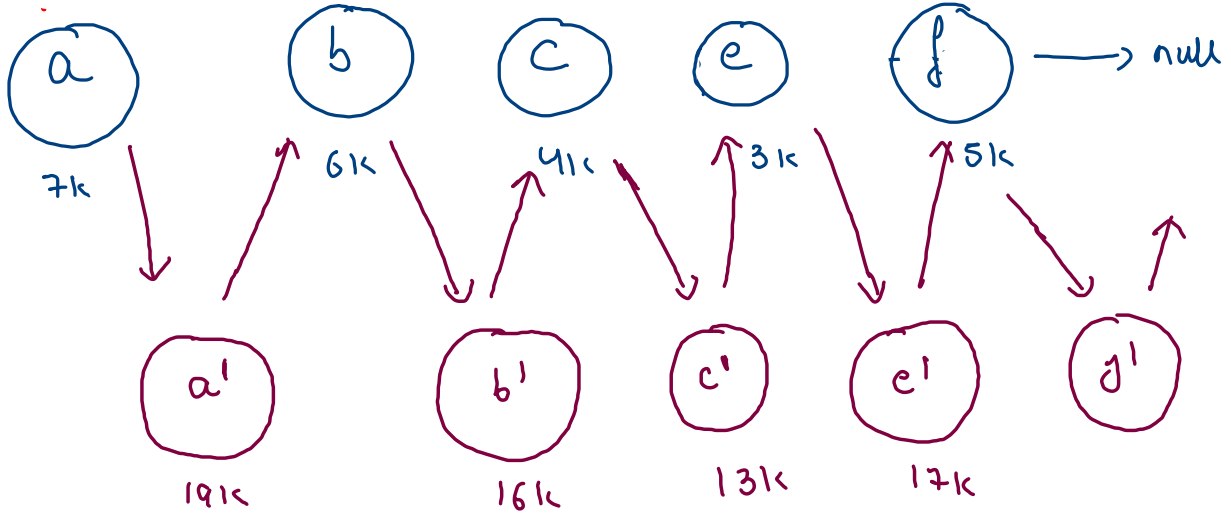


o(n) : T

o(n) : S



1)



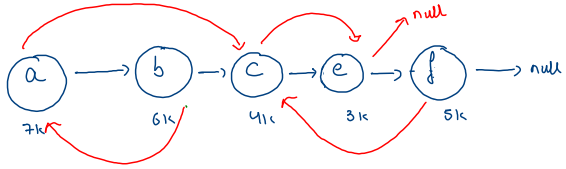
O(n) : T

O(1) : S

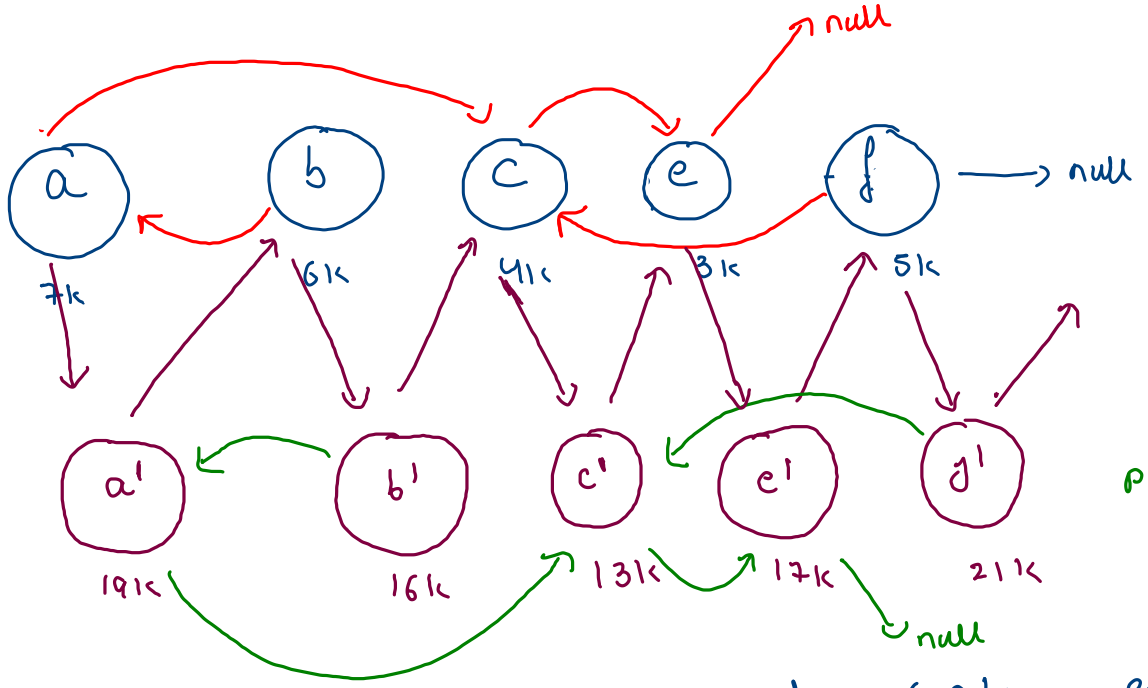
c.next = nn

nn.next = n

insert new nodes blw old nodes.

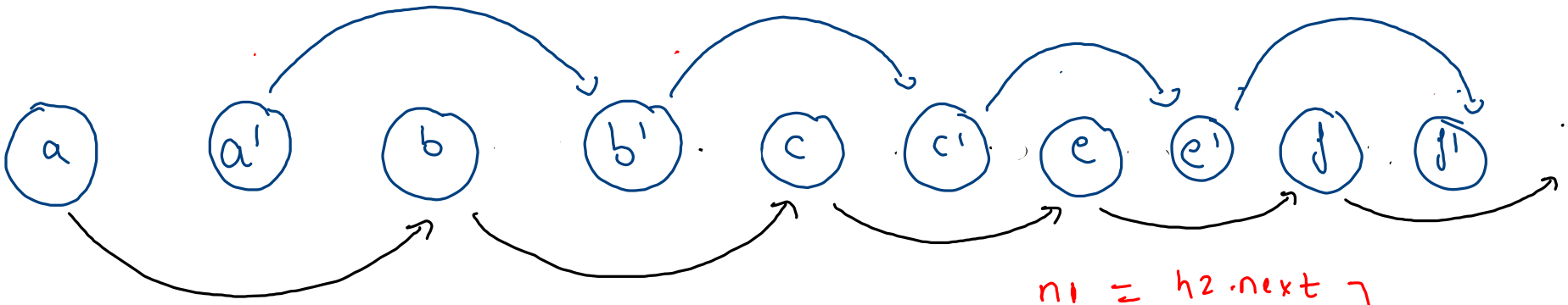


(2)



p2.random = p1.random.next

set random ptr.




```
n1 = h2.next
n2 = n1.next      (n1 != null)

h1.next = n1
h2.next = n2

h1 = n1;  h2 = n2;
```

Segregate old & new list