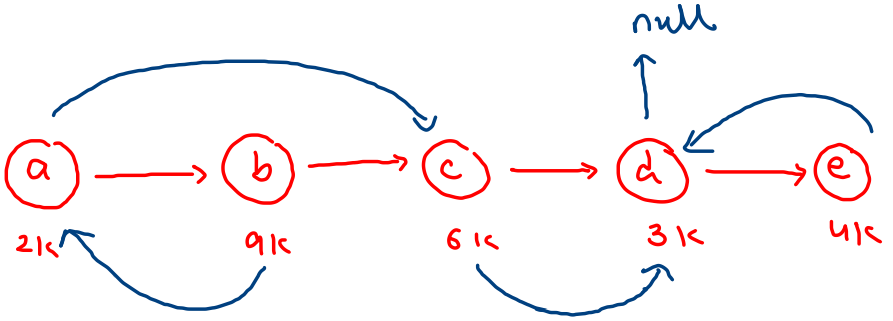
138. Copy List with Random Pointer



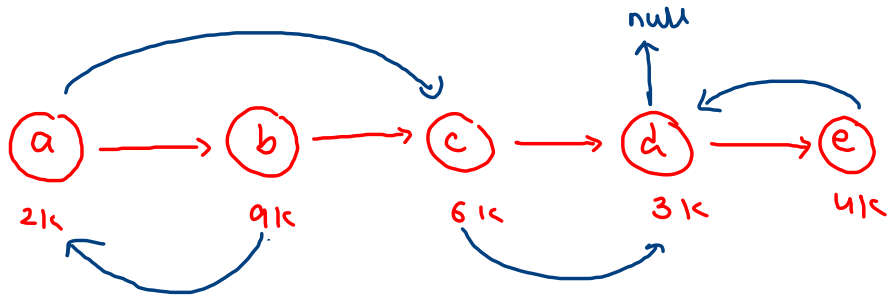
red : next

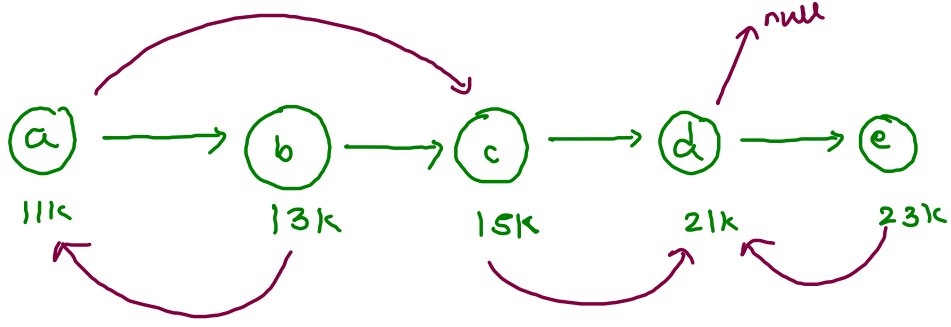
blue : random

```
Node {
  int val;
  Node next;
  Node random;
}
```

T: O(n)

S: O(n)





21k → 111k

91k → 131k

6k → 15k

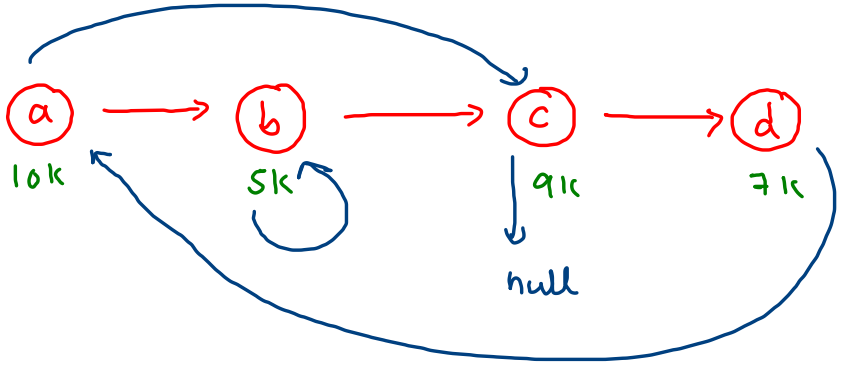
3k → 21k

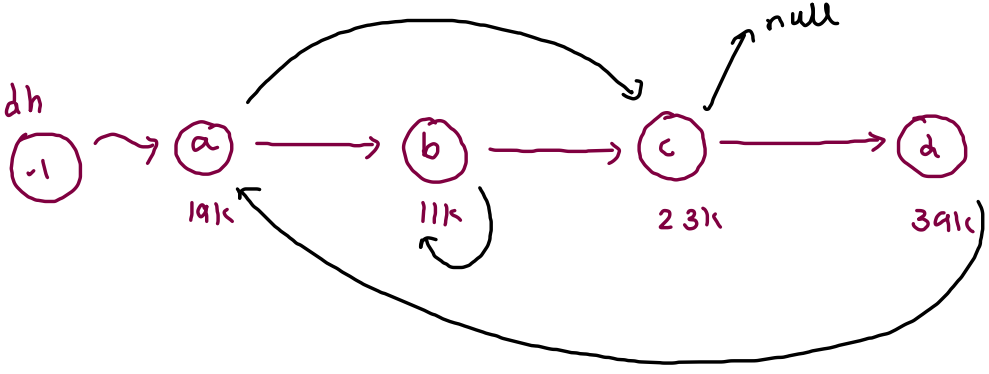
4k → 23k

hashmap

old address vs new address

t2.random = map.get(t1.random);





(i) copy value and next, and maintain a HM having key as old node and value as new node

10k → 191k

5k → 111k

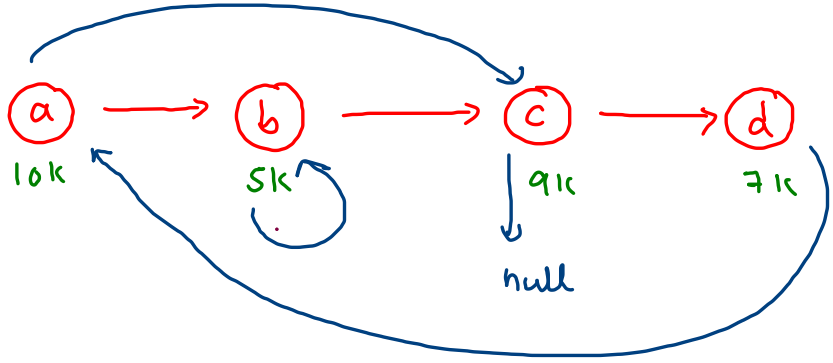
9k → 231k

7k → 391k

(ii) now set random ptr of copied LL.

t2.random = map.get(t1.random)

without hashmap ?



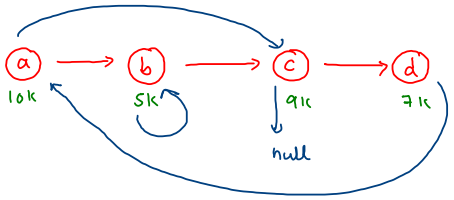
curr

(a) → (a') → (b) → (b') → (c) ....

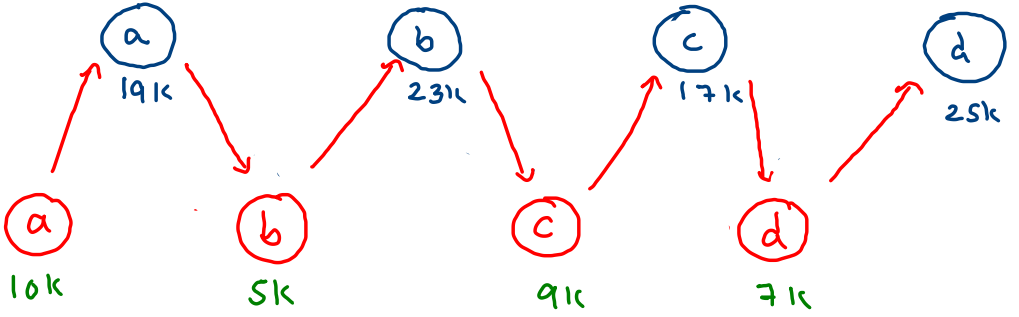
(i) insert copied nodes between original nodes.

(ii) set random ptr curr.next.random = curr.random.next.

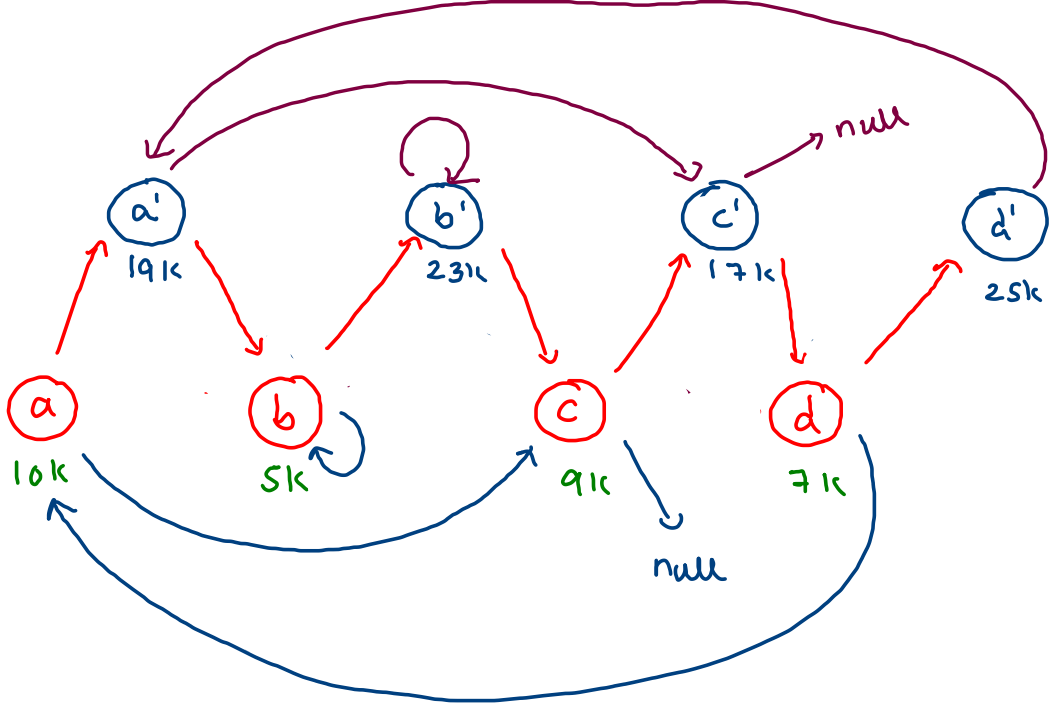
(iii) segregate original and copied LL.


a
10k
b
5k
c
91k
null
d
7 1k

Step 1.

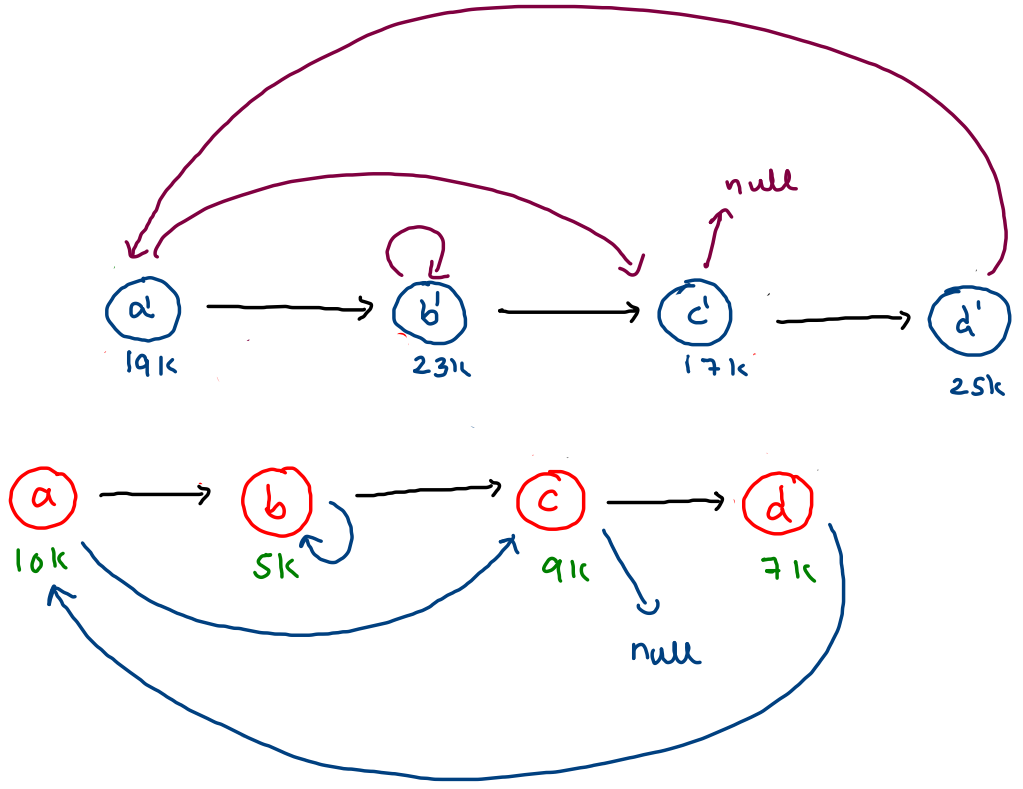

a
19 1k
b
23 1k
c
17 1k
d
25k
a
10k
b
5k
c
91k
d
7 1k

Step 2.

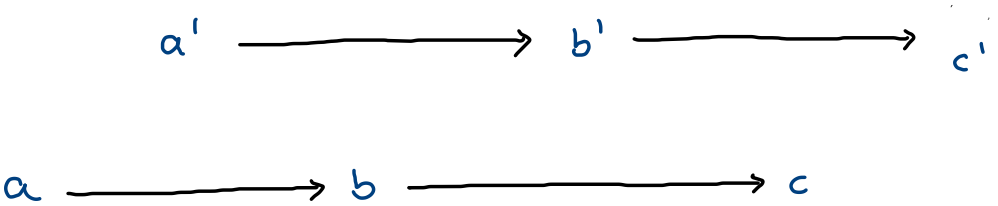


curr.next.random = curr.random.next;

Step 3

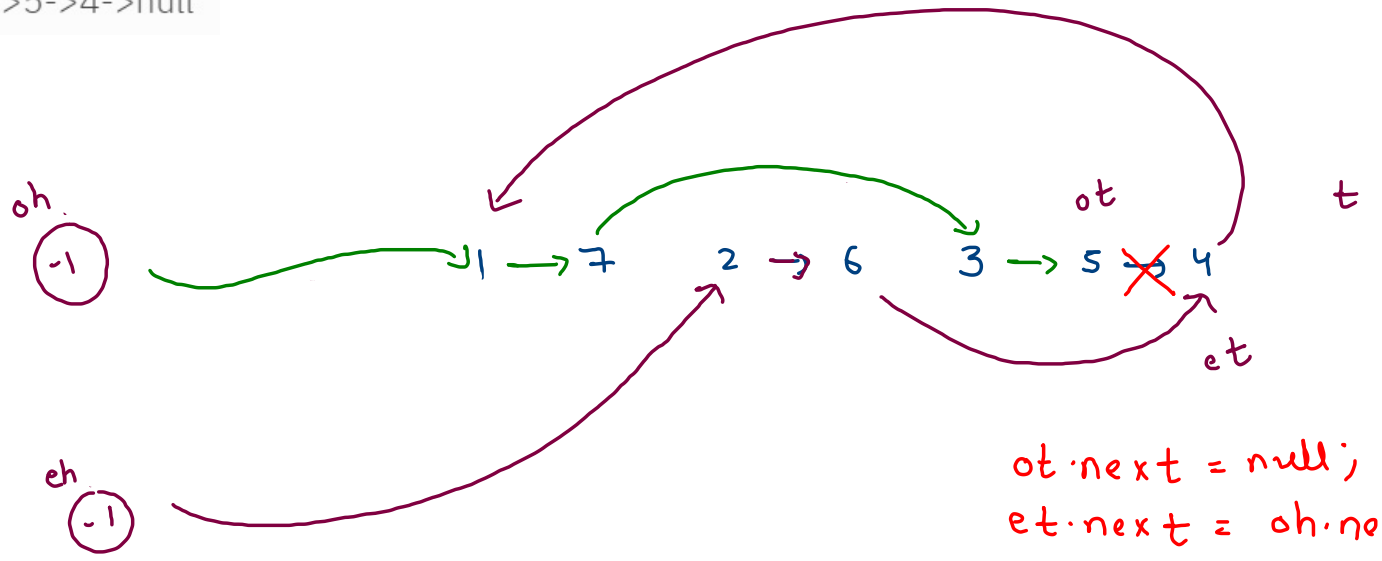


```java
//segregate the original and copied LL
Node c1 = head;
Node ans = c1.next;
Node c2 = c1.next;

while(c1 != null && c2 != null) {
    Node n1 = c2.next;
    Node n2 = (n1 == null) ? null : n1.next;

    c1.next = n1;
    c2.next = n2;

    c1 = n1;
    c2 = n2;
}
```



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;
```

Remove Nth Node From End Of Linkedlist

k = 3