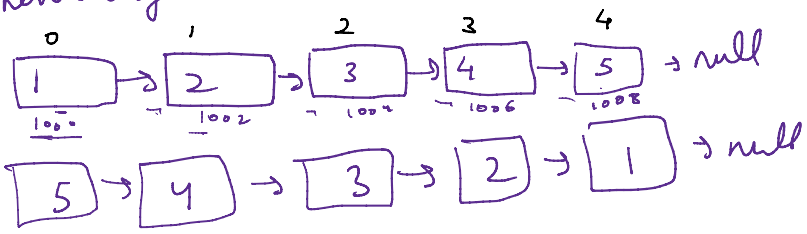
27-Dec-2021

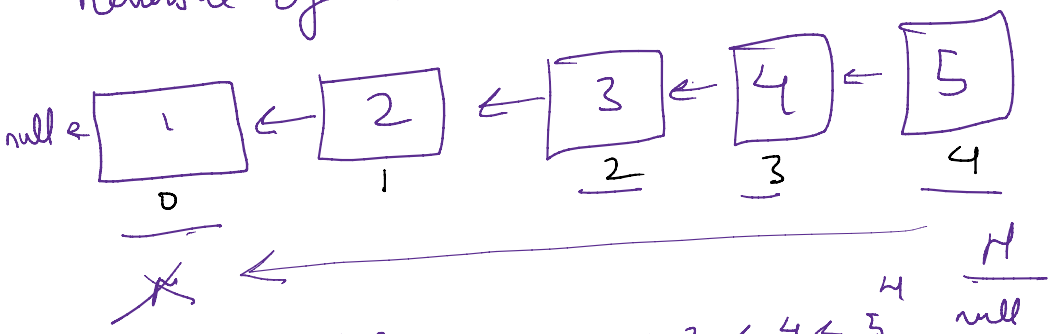
27 December 2021 21:03

Reversal of LL with Data:-




```
reversalWithData() {                    T.C. => n^2
    i = 0;
    j = size - 1;
    while (i < j) {
    Node in = getAt(i);
    Node jn = getAt(j);
    swap(in, jn);
      i++;
      j--;
  }
}
```

Reversal of Linked List with pointers

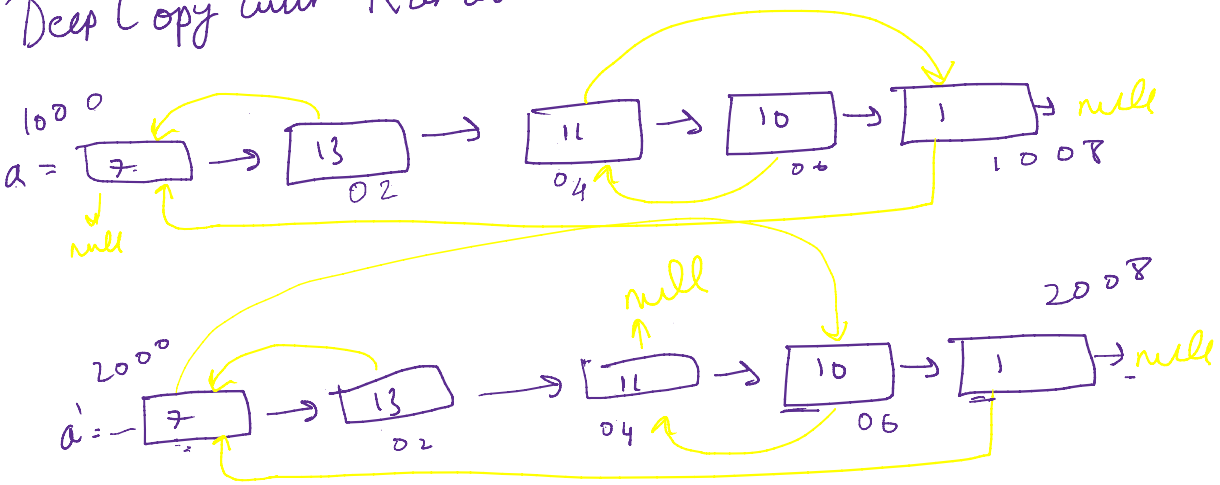



```
reversalWithPointers() {
  c = head;
  p = null;
  while (c.next != null) {
    n = c.next;                    T.C. => O(n).
  c.next = p
    p = c;
    c = n;
  }
    c.next = p;
  head = c;
}
```

Reversal With Pointers Recursion => Homework

Reversal With Pointer Recursion => Homework

Deep Copy with Random Pointers :- TC => O(n), S.C => O(n)



Using Map :=)

a : 7 -> 13 -> 11 -> 10 -> 1 -> null

oN = 13 11

nN = 11

nN.random = map.get(oN.random);

}

map = old K Node | new V Node

old K Node	new V Node
7	7
13	13
11	11
10	10
1	1

Steps :-

=> Make a new LL without setting Random Pointers

=> Make a hashMap with old & new LL

=> Setup the Random Pointers.

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

n

-1 → 1 → 2 → 3 → 4 → 5 → null

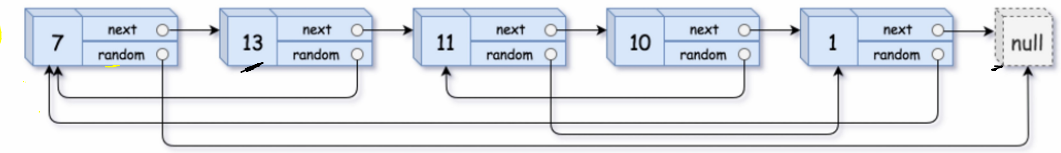
res = res.next;

K	V
1	1
2	2
3	3
4	4
5	5

```
Node n=head;
Node newLL=new Node(-1);
Node res=newLL;
while(n!=null){
    Node node=new Node(n.val);
    newLL.next=node;
    newLL=newLL.next;
    n=n.next;
}
```

```
while(n!=null){
    map.put(n,newLL);
    n=n.next;
    newLL=newLL.next;
}
```

old =



new → 7 → 13 → 11 → 10 → 1 → null

K = old Nodes

V = newNodes