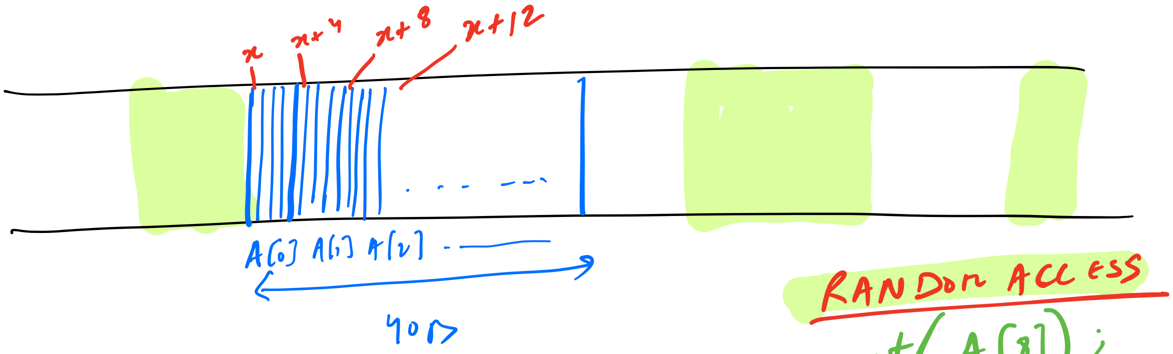
Array.

int A[10]; → 40B

int → 4B



RANDOM ACCESS

print(A[8]); → O(1)

$A[0] \rightarrow \boxed{x}$

$A[1] \rightarrow x+4$

$A[2] \rightarrow x+8$

⋮

$\boxed{A[i] \rightarrow x + 4 \times i}$

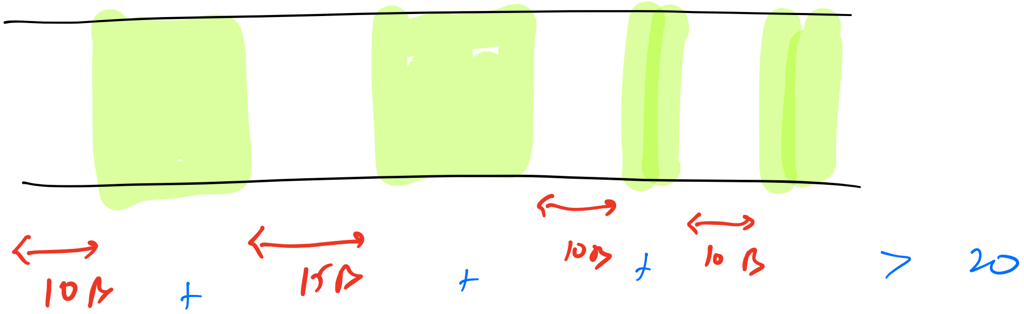
index.

BASE Add. of A[]

size of data type

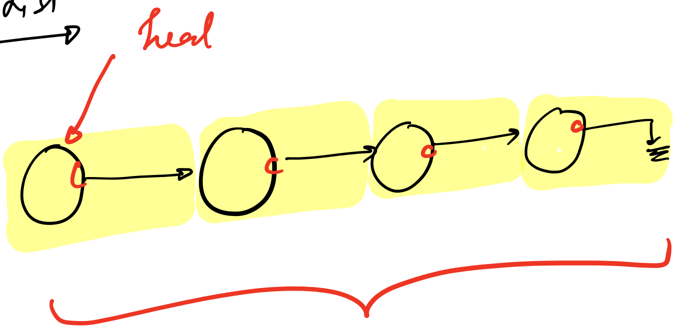
- → Size cannot be increased → : Dynamic Array
- → Insert in b/w is difficult → X
- → Deletion from b/w ——— → X

int A[5]; → 20B ✗

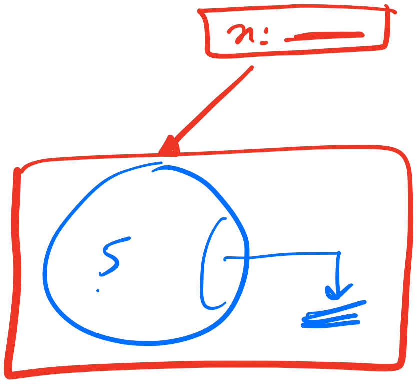


→ Memory fragmentation is not helping → ✗ with proper memory utilization.

Linked List



```
class Node {
        data;
    Node next; → 4B
        Node(int val) {
            data = val;
            next = NULL
        }
}
```

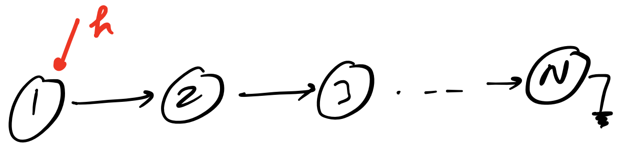


Node x = new Node(5);

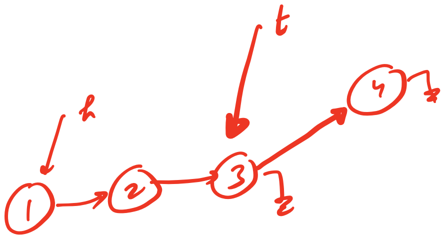
print(x.data);

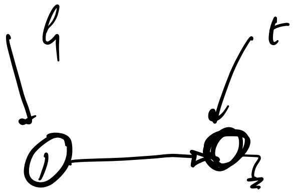
Q Create a LL.

Given N: $N \geq 1$

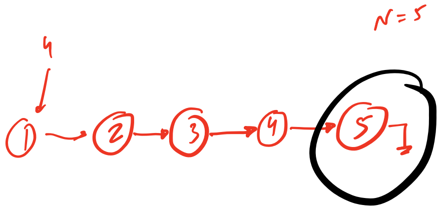


```
Node h = new Node(1);
Node t = h;
f(i=2; i<=N; i++) {
    t.next = new Node(i);
        t = t.next;
}
ret h;
```

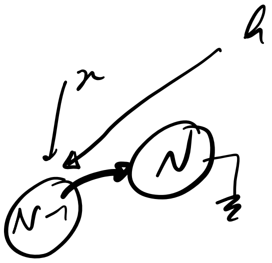




II



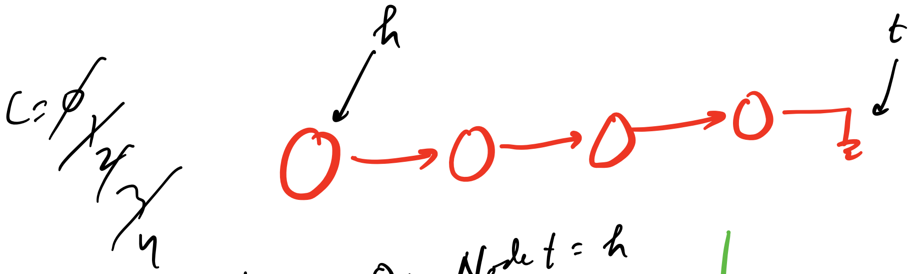
```
Node h = new Node(N);
f( i=N-1; i>=1; i--) {
    Node x = new Node(i);
    x.next = h;
    h = x;
}
ret h;
```



$$\boxed{TC = O(N)}$$

for I & II

Q Given a LL. Calc. the size of the LL. [No. of nodes]



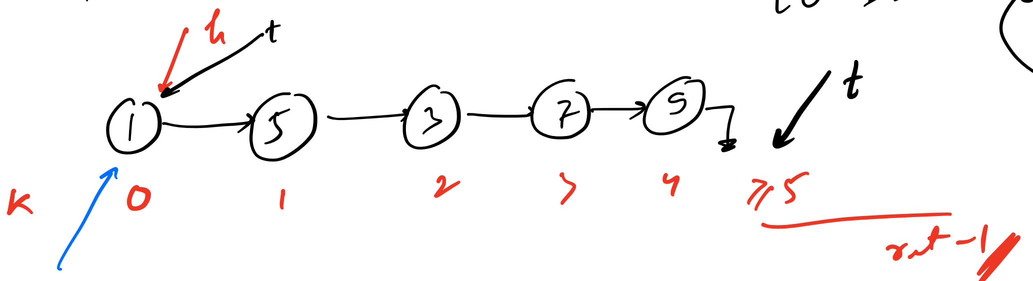
```
int c=0; Node t=h
while( t != NULL) {
        c++;
        t = t.next;

}
ret c;
```

$\boxed{TC = O(N)}$

$\boxed{SC = O(1)}$

Q Given a LL. Find the element at K^{th} pos [0-based]

invalid ret -1



```
Node t = h;
f( i=1; i<=K && t!=NULL; i++) {
        t = t.next;

}
if(t == NULL) ret -1;
 ret t.data;
```

```
Node t = h
c = 0;
while( c < k && t != NULL) {
        c++;
        t = t.next;
}
if( t == NULL) ret -1;
    ret t.data;
```

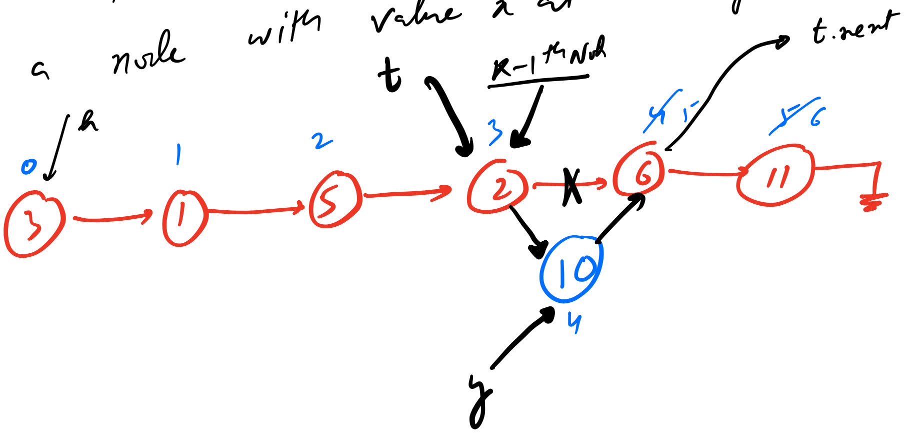
$TC = O(min(N, K))$

$SC = O(1)$

$N \to 10^5$ | 5
$K \to 5$

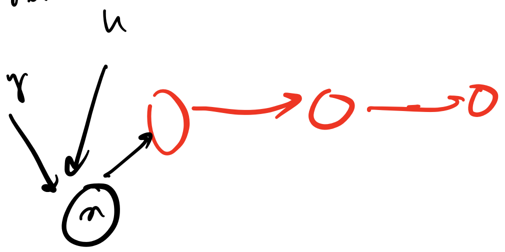
$N \to 5$ | 5
$K \to 10^5$

Q Given a LL, K & x.
Insert a node with value x at K^{th} pos.



K = 4
X = 10

```
Node t = getKthNode(h, K-1);
 Node y = new Node(x);
  y.next = t.next;
  t.next = y;
  ret h;
```



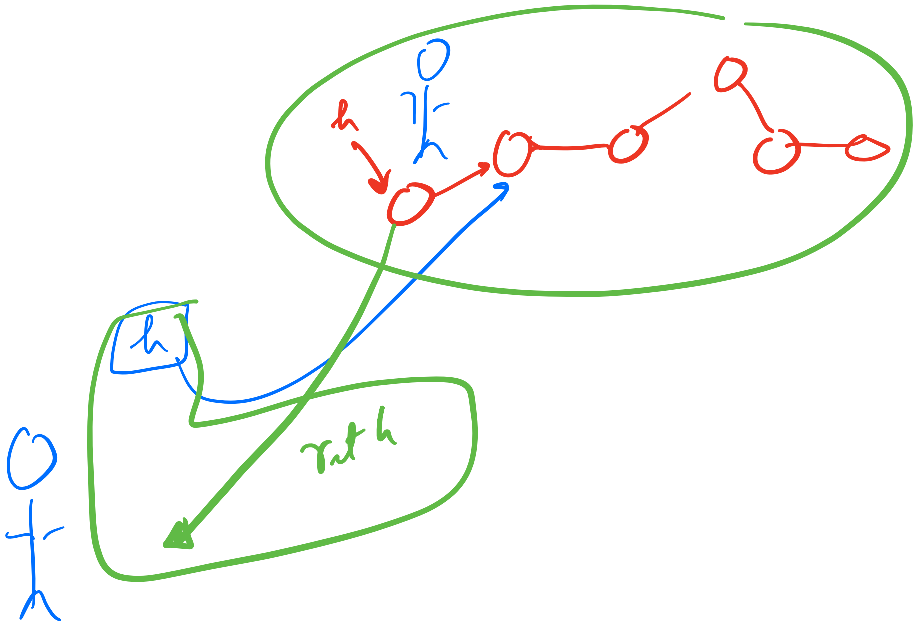
CORNER CASES

1) if (t == NULL)
K > N
→ NOT possible to INSERT

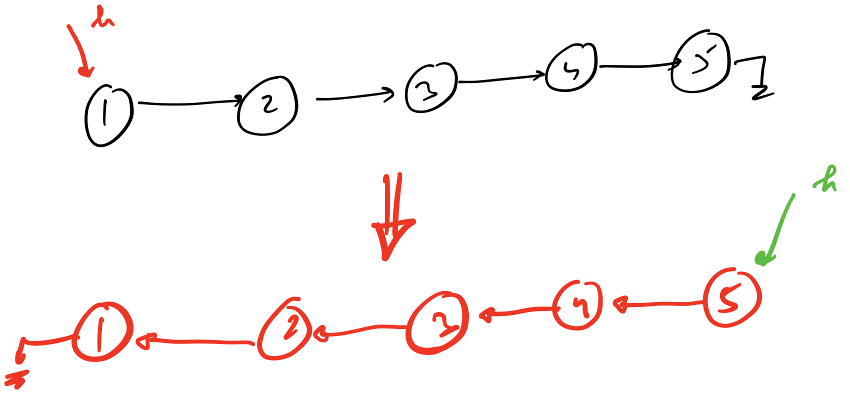
ret h;

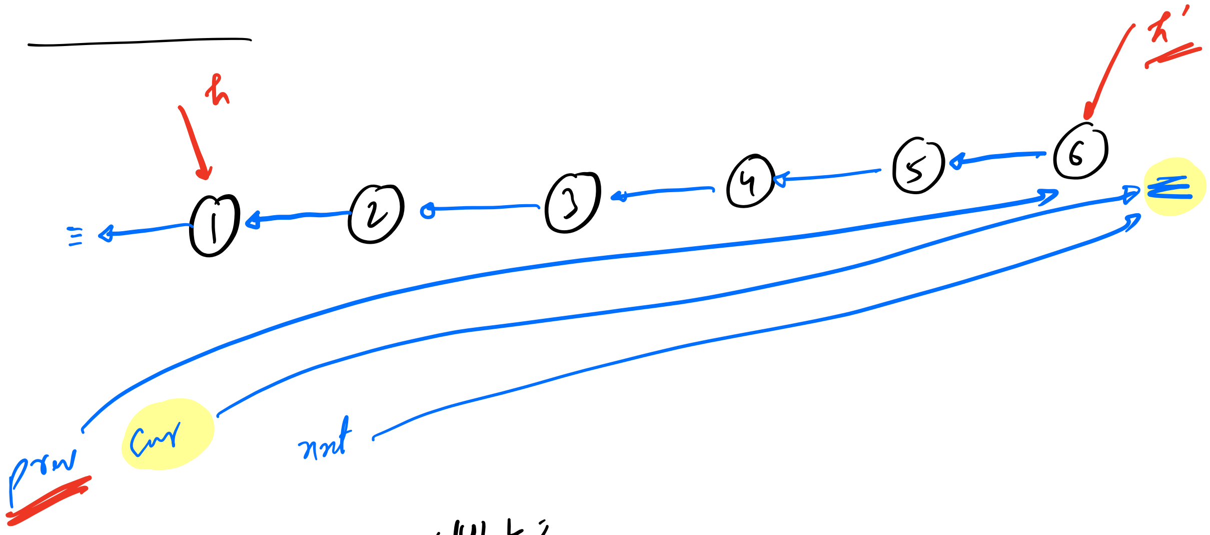
2) K = 0

```
Node y = new Node(x);
  y.next = head;
   h = y;
  ret h;
```



Q Given a LL. Reverse it !





```
Node prev = NULL;
Node curr = h;
while( curr != NULL) {
    Node nxt = curr.next;
    curr.next = prev;
    prev = curr;
    curr = nxt;
}
ret prev;
```

$\boxed{TC = O(N)}$

$\boxed{SC = O(1)}$

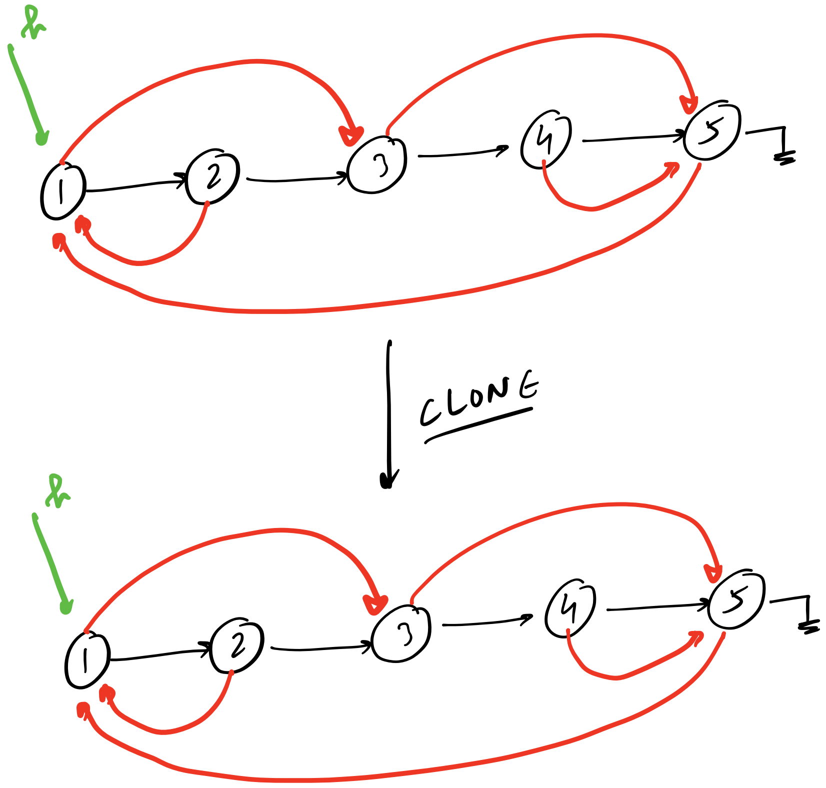
Q Given a L.L, where nodes have 2 ptrs

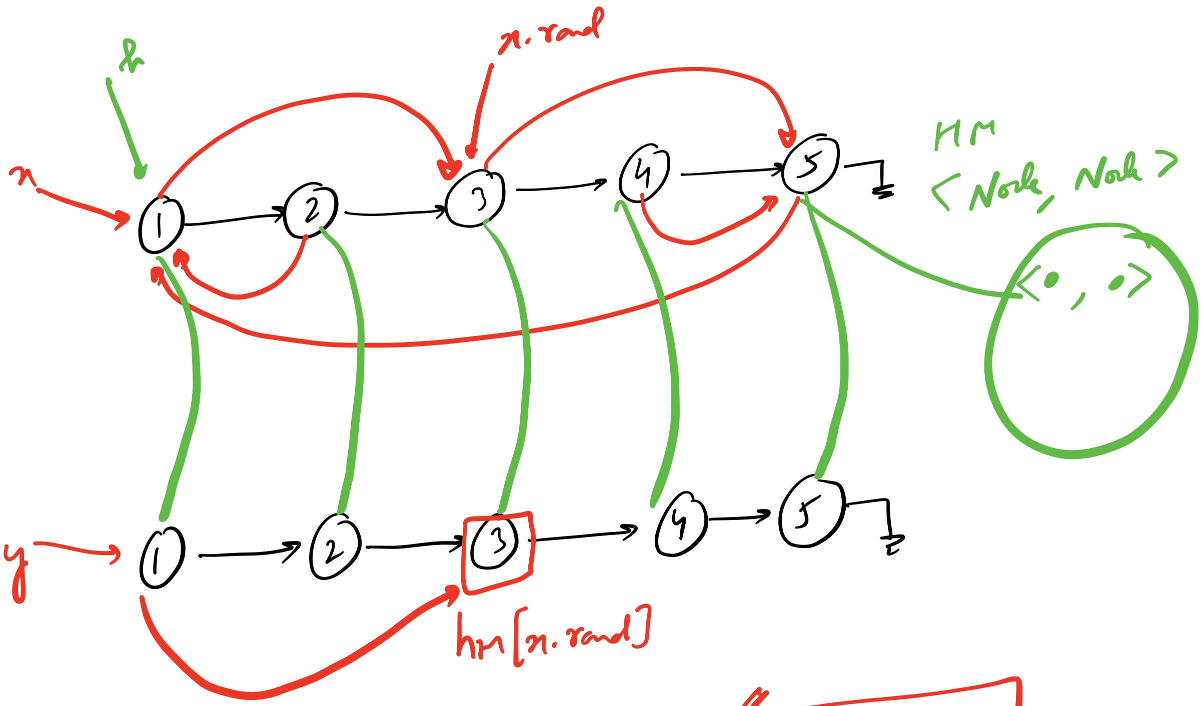
Clone this L.L.

→ next

→ rand

point to any random node



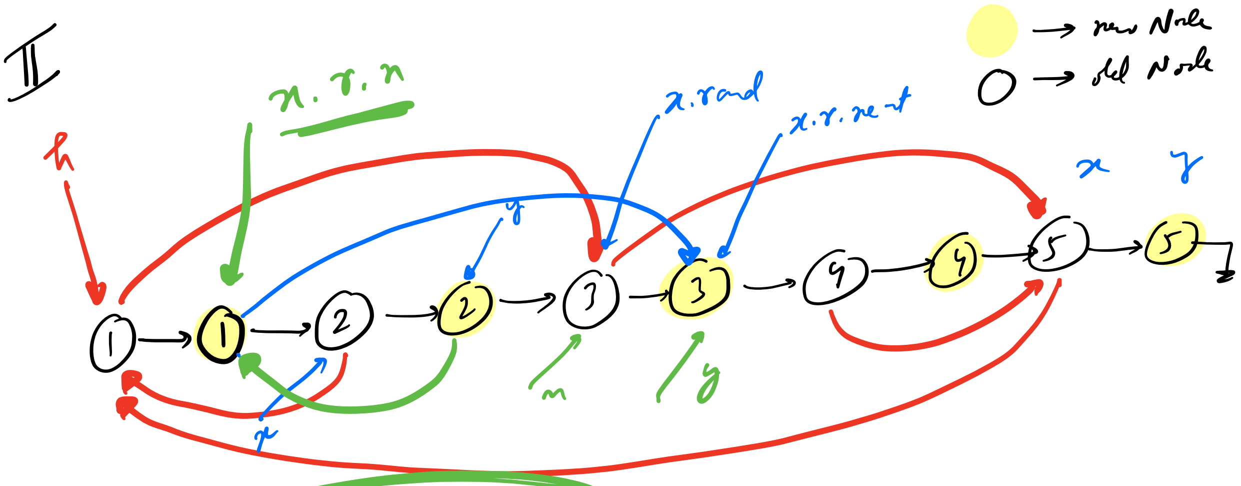


$$\boxed{TC = O(N)}$$

$$\boxed{SC = O(N)}$$

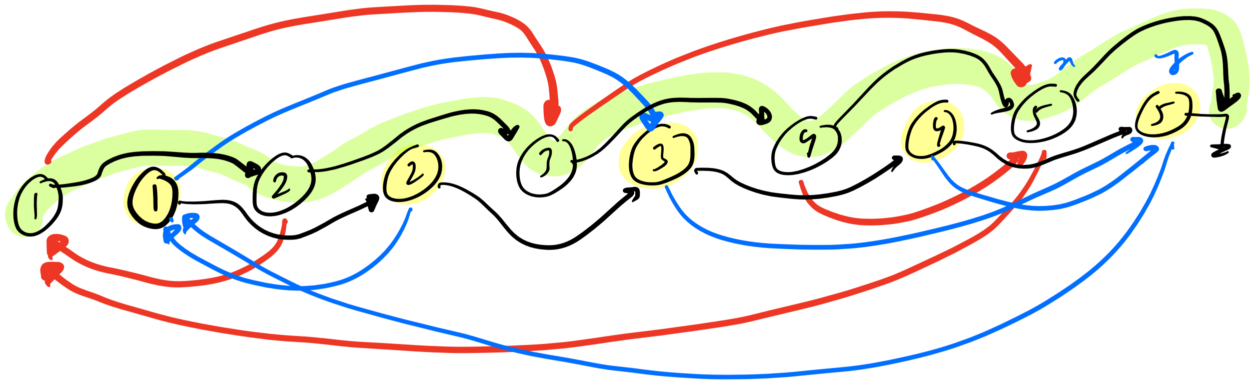
```
x = h
y = h'
loop
    y.rand = hm[x.rand]
    x = x.next
    y = y.—
```

II



loop

```
y.rand = x.rand.next;
x = x.next.next;
y = y.next.next;
```



```
x.next = y.next; ←
y.next = y.next.next;
x = x.next
y = y.next
```

STOPPING COND.

TC : O(N)

SC : O(1)