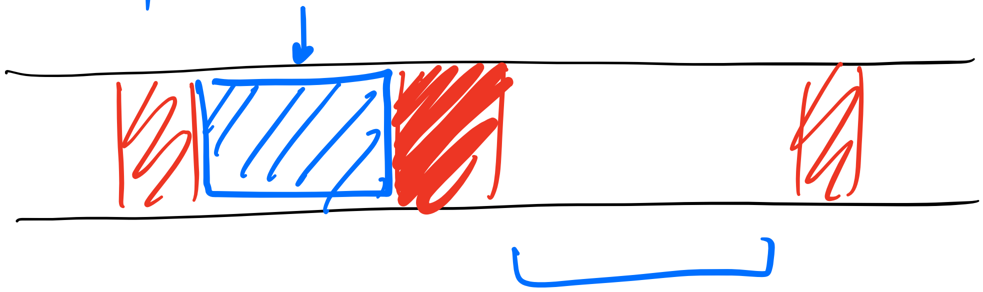
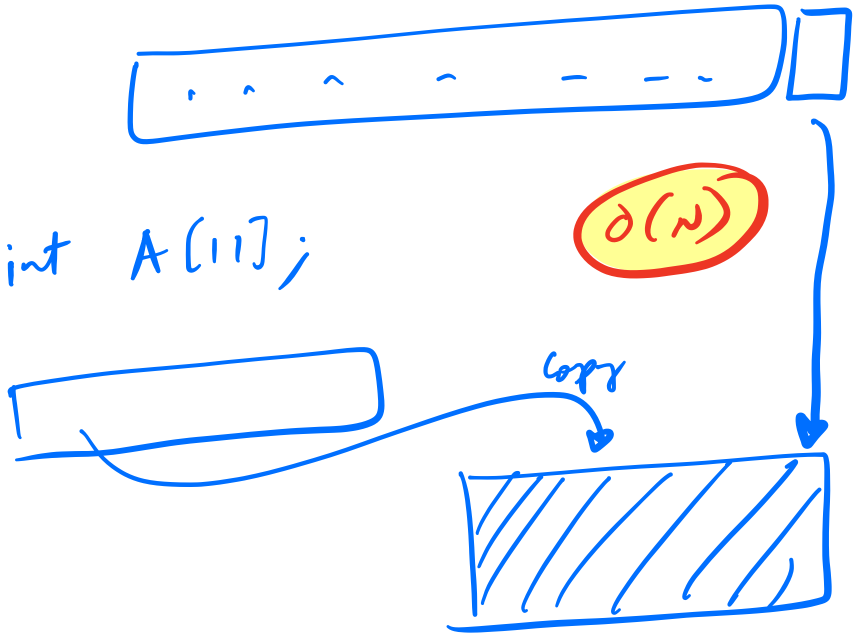
Linked List

I

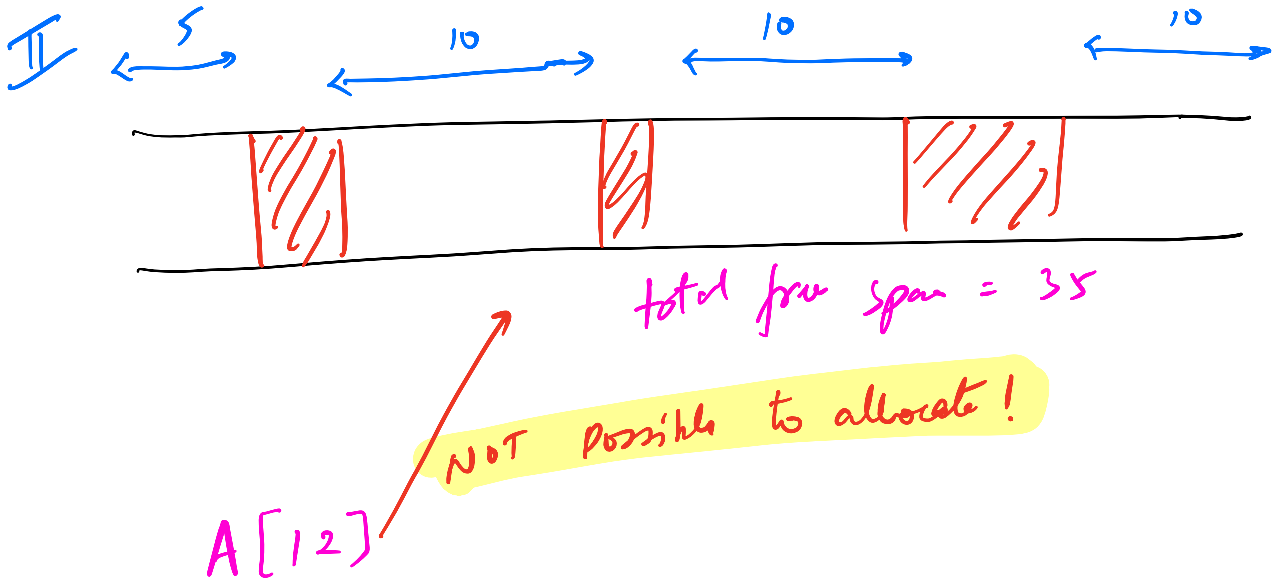
Array :

int A[10];



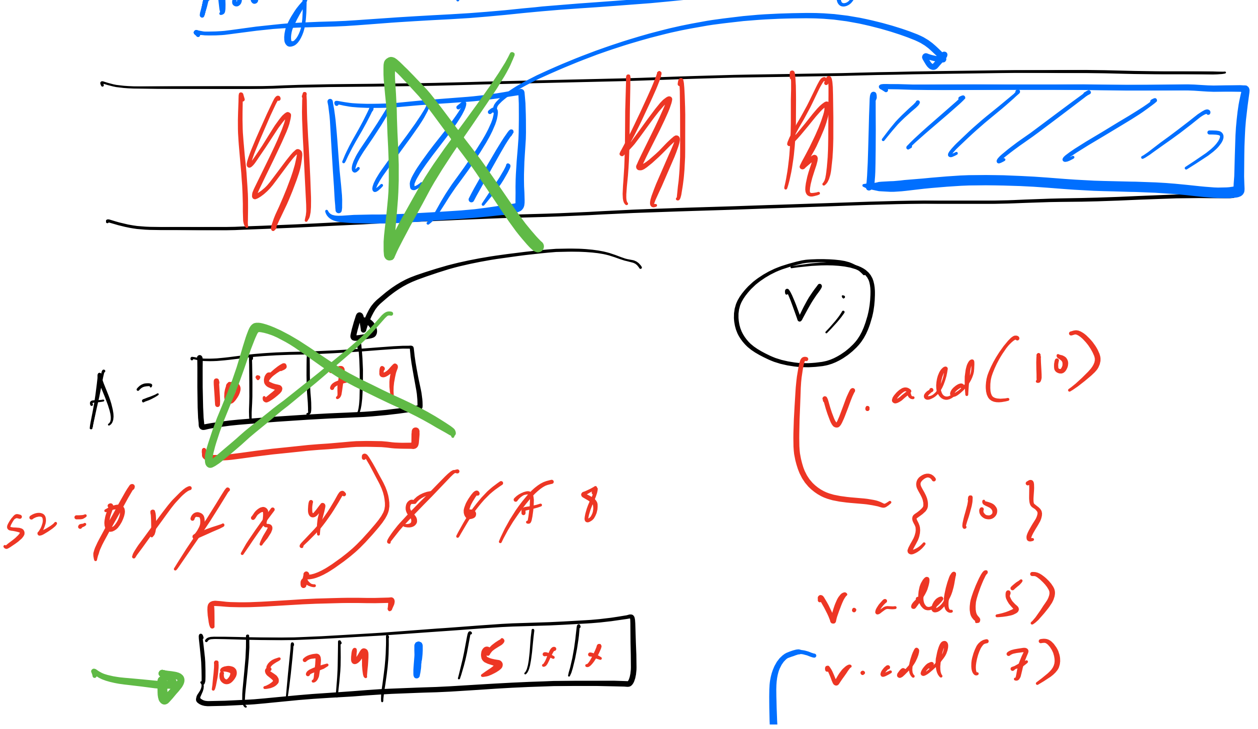


Increasing the size of the Array is INFEASIBLE!



We have adequate amt. of free space but still we might not be able to create arrays!

→ Array list / vector : Dynamic Arrays


A =
10 | 5 | 7 | 4
sz = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ 8
10 | 5 | 7 | 4 | 1 | 5 | x | x
V

v.add(10)

{10}

v.add(5)

v.add(7)

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

v.add(4)

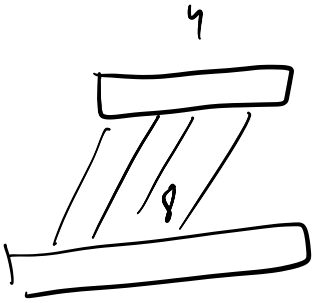
v.add(1)

v.add(5)

⋮

2

{1, 5, 10, 8, ...}

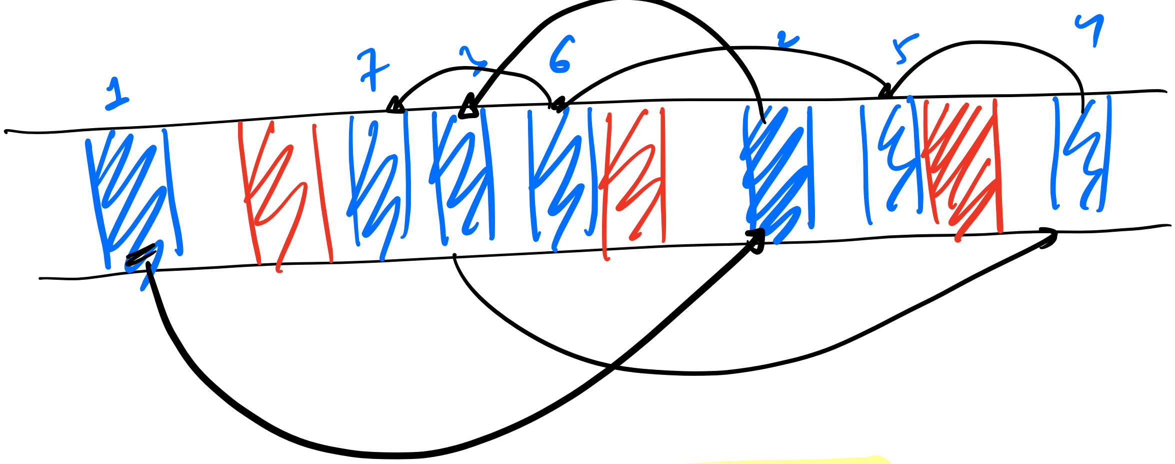


4 → O(1)

5 → 4

6
|
8 → O(1)

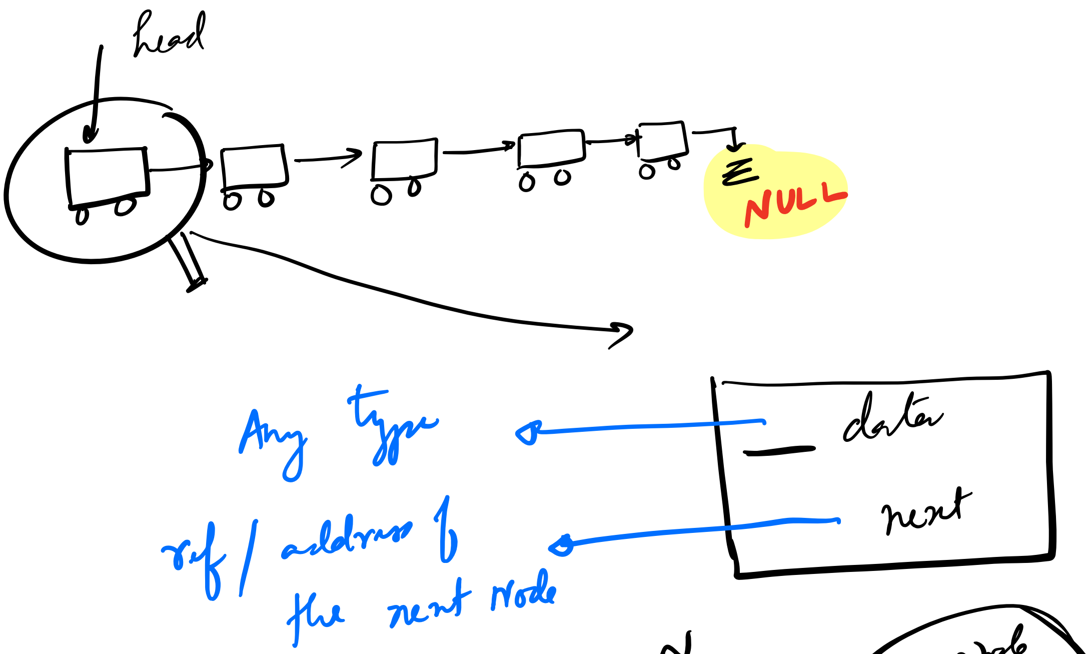
Linked List

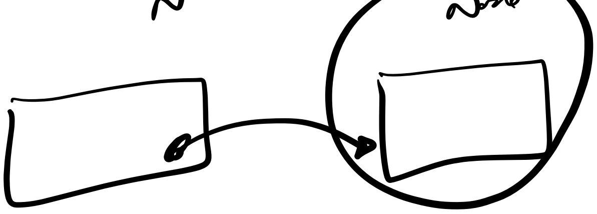


✓ MEMORY UTILIZATION

✓ INC / DEC SIZE

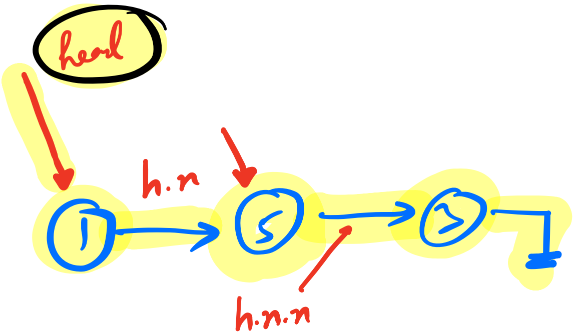
→ NO RANDOM INDEX ACCESS





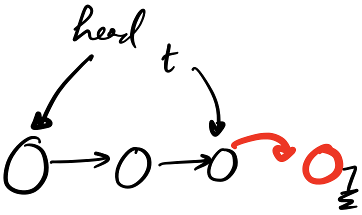
```
class Node {
    int data;
    Node next;
    Node(x) {
      this.data = x;
      this.next = NULL;
    }
}
```

```
Node head = new Node(1);
head.next = new Node(5);
head.next.next = new Node(3);
```



Q Create a f^n for inserting a node at the back of the Linked List!

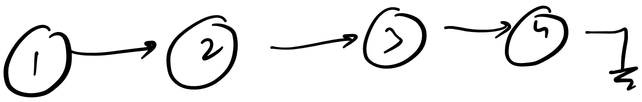
```
Node insert(Node head, int val){
    // insert node with data = val at the end of LL
    // ret the ref. of 1st node (head).
    if(head == NULL){
        head = new Node(val);
        ret head;
    }
    Node t = head;
    while( t.next != NULL){
        t = t.next;
    }
    t.next = new Node(val);
    ret head;
}
```



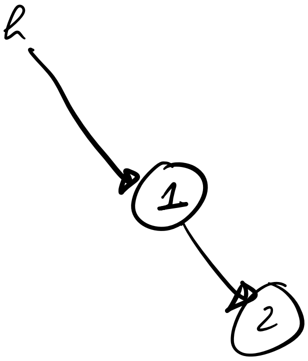
TC : O(N)

Q Create a LL with values from [1 - N].

N = 4



```
Node head = NULL;
f( i = 1 ; i <= N; i++) {
    head = insert( head, i );
}
ret head;
```

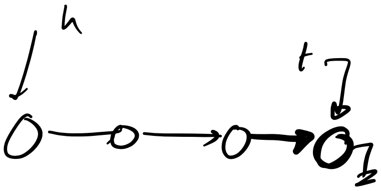


$\# \text{ ops} = 1 + 2 + 3 + \text{---} + N$

$\boxed{TC = O(N^2)}$

II Maintain head & tail

```
Node head = NULL ; Node tail = NULL;
f( i = 1; i <= N; i++) {
    if( i == 1) {
        head = new Node(i);
        tail = head;
    }
    else {
        tail.next = new Node(i);
        tail = tail.next;
    }
```



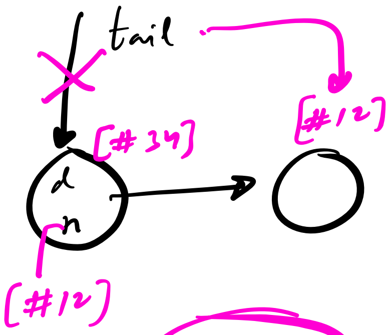
$\boxed{TC = O(N)}$

```
}
ret head;
```

tail = #34

tail = tail.next

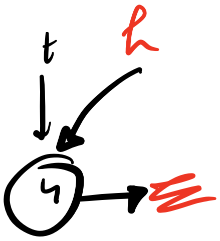
tail = #12

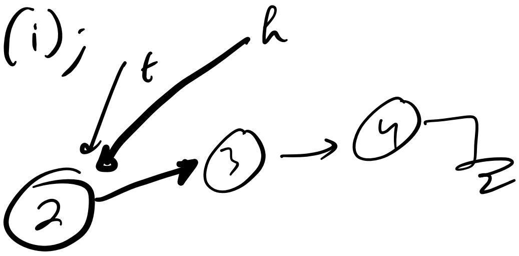


t.next

III NOTE: ADDING IN FRONT!

```
Node h = NULL;        N=4
f( i=N; i>=1; i--) {
    Node t = new Node(i);
      t.next = h;
      h = t;
}
ret h;
```

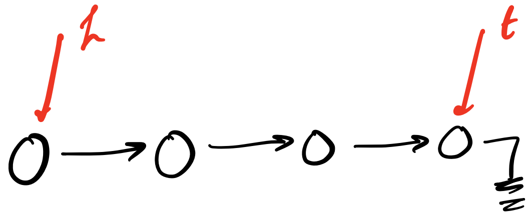


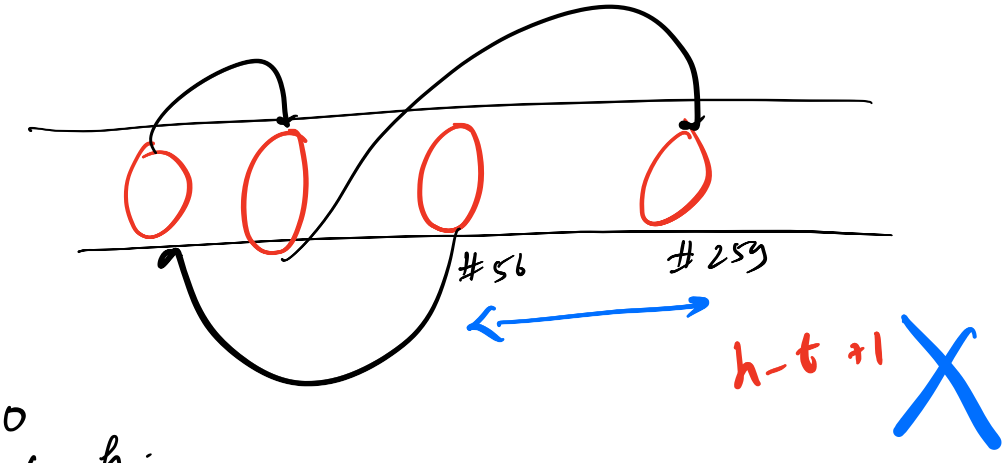


$\boxed{TC = O(N)}$

Q Given a LL. find the size of it!

→ # of Nodes!

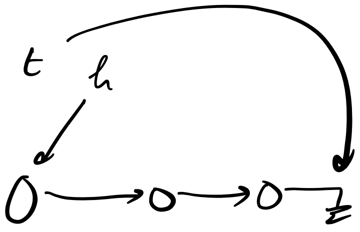




```
cnt = 0
Node t = h;
while( t != NULL){
    cnt++;
    t = t.next;

}
ret cnt;
```

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

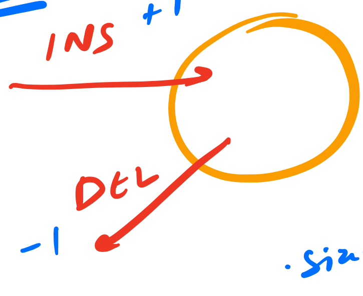


cnt
~~0~~ ~~1~~ ~~2~~ 3 ✓

Sz variable

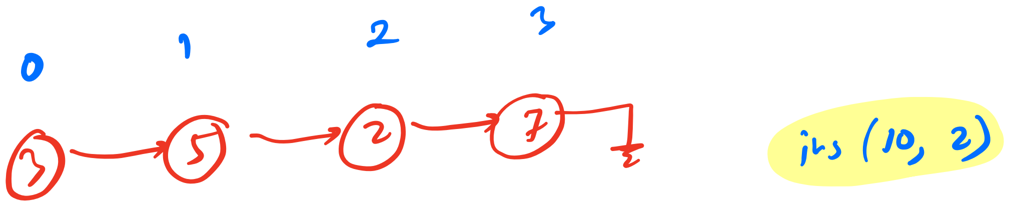
~~0~~ +1 −1

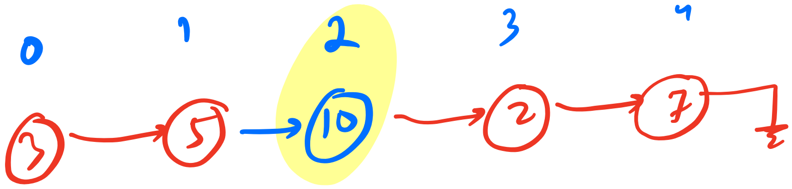
Thought



.size() → O(1)

Q Given a LL & K, val
Insert a new node (val) at K^th pos!



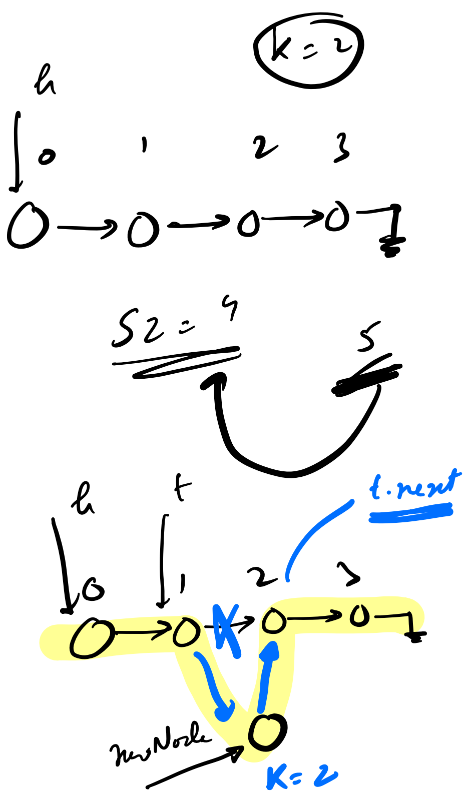


K >= 0

```
Node insertAt(head, K, val){
// ret the head!
    if(K==0){
        Node t= new Node(val);
        t.next= head;
        head= t;
        ret head;
    }
    if( K > size(head)){
            ret head;
    }
    Node t = head;
    for(i=1; i<=K-1; i++){
        t= t.next;

    }
```

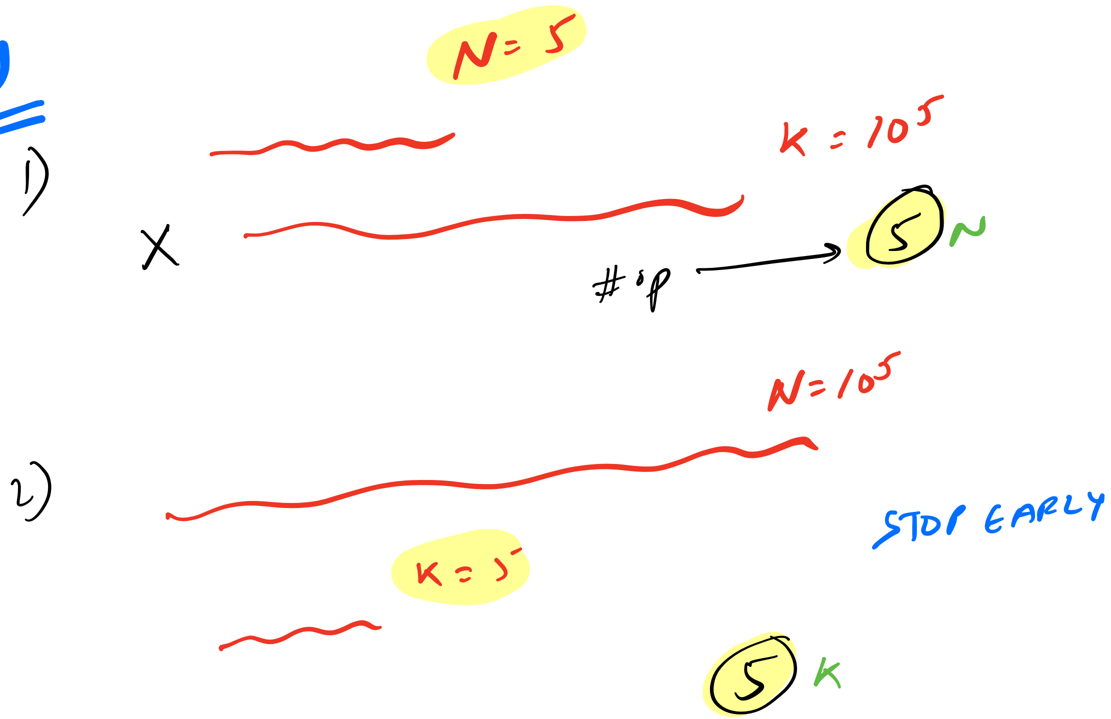
N



```
Node newNode = new Node(val);
newNode.next = t.next;
t.next = newNode;
ret h;
}
```

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

HW



$$\boxed{TC = O(Min(N, K))}$$

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