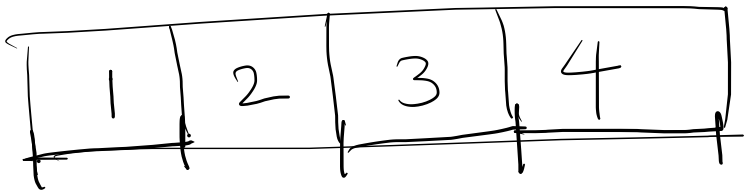
LINKED LIST

Linear data structure

elements → sequentially

ARRAY

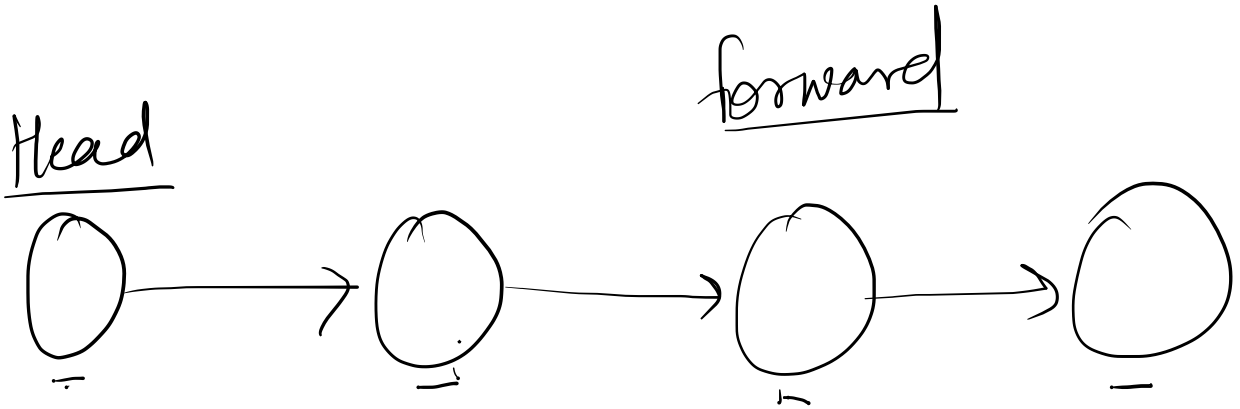


Contiguous memory allocation

Linked list

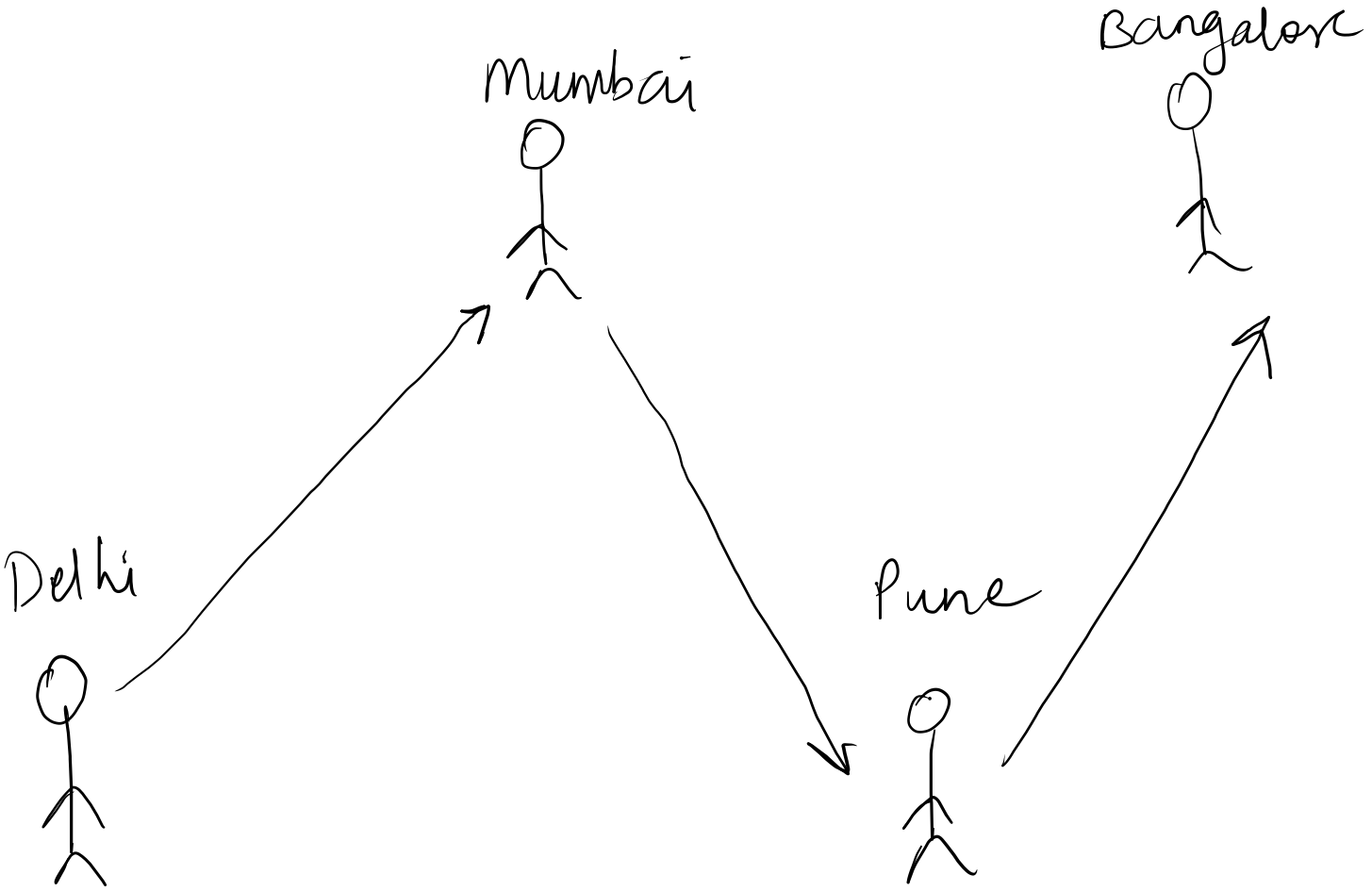
nodes are connected

non-contiguous memory allocation

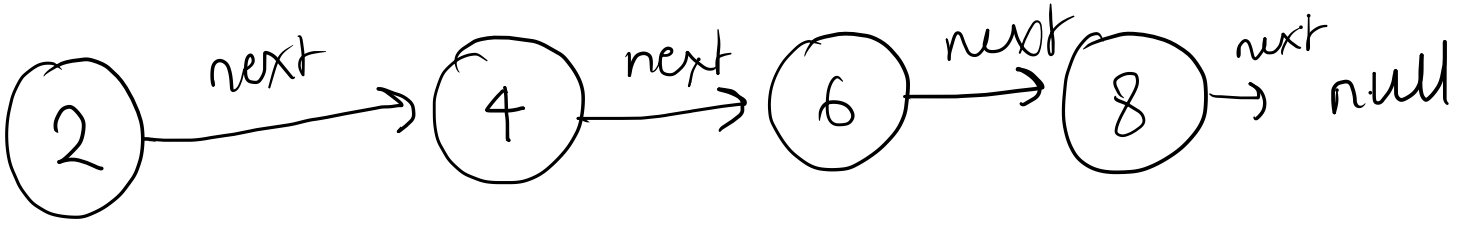


Singly linked linked

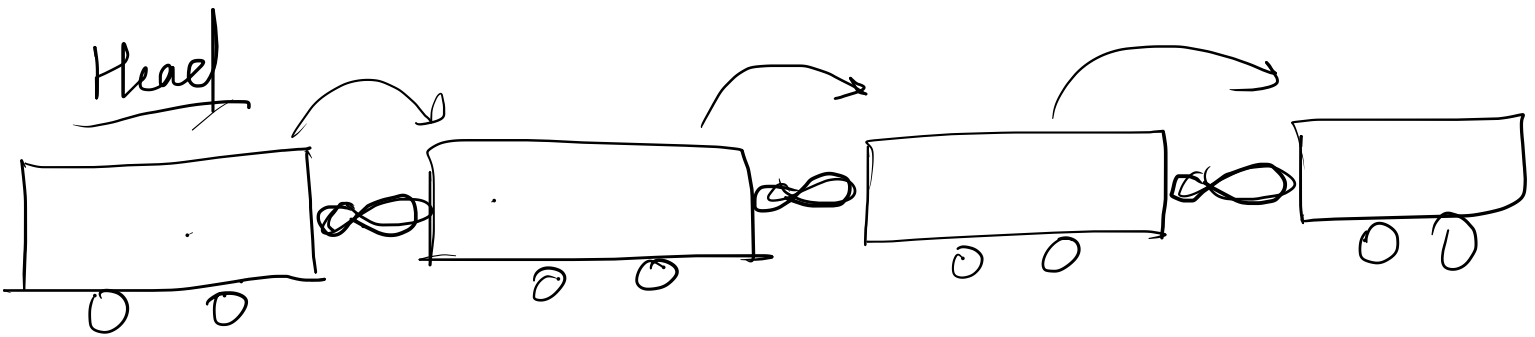
↳ Unidirectional


Bangalore
Mumbai
Delhi
Pune

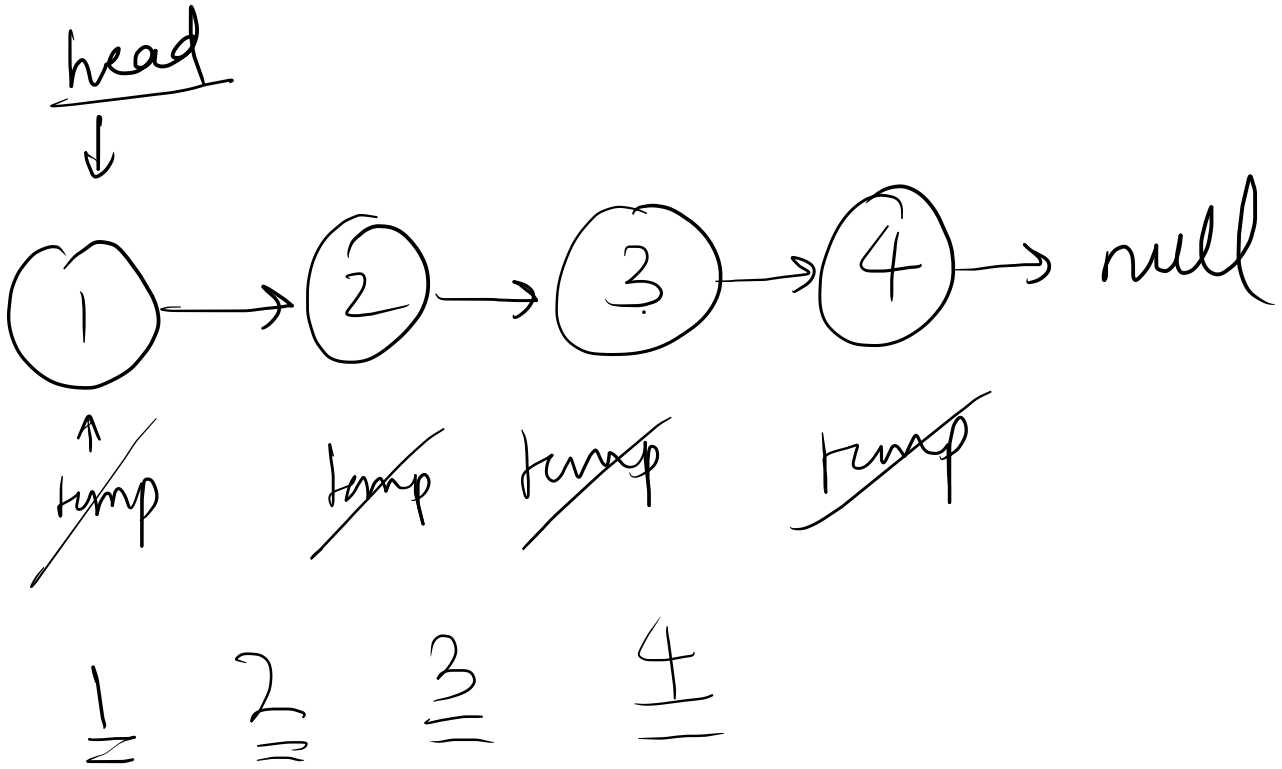
Node → data
Node → address of next



lastNode → next → null

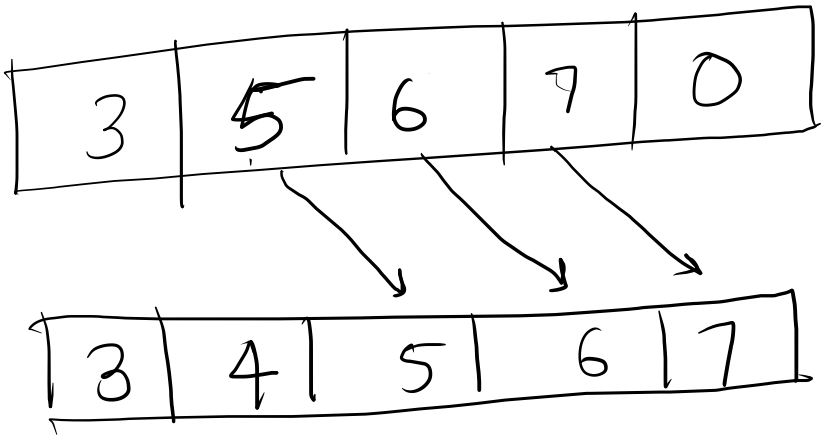


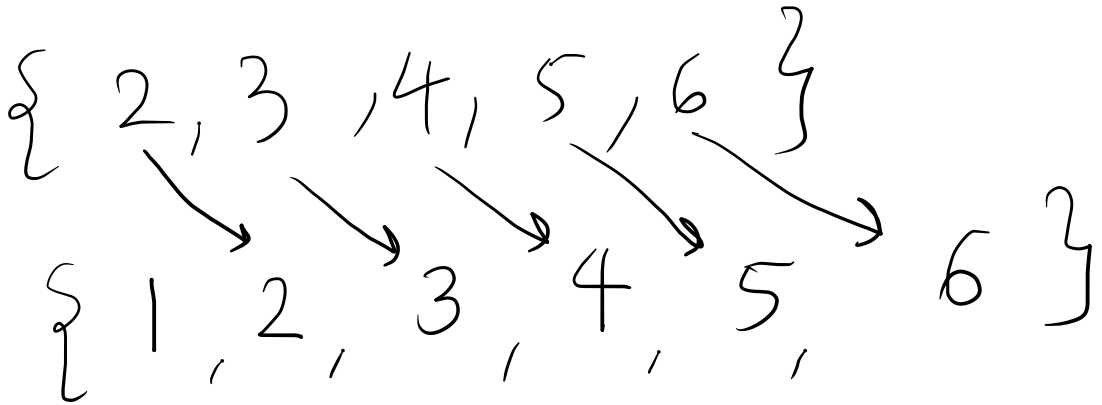
Nodes are connected



1 2 3 4

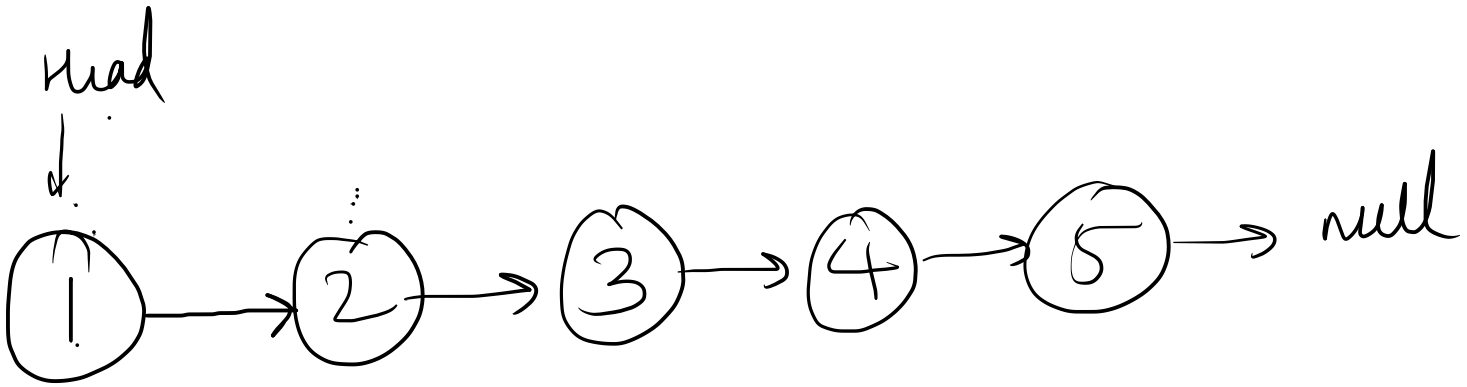
Advantage of linked list





Insertion in Array

O(n)



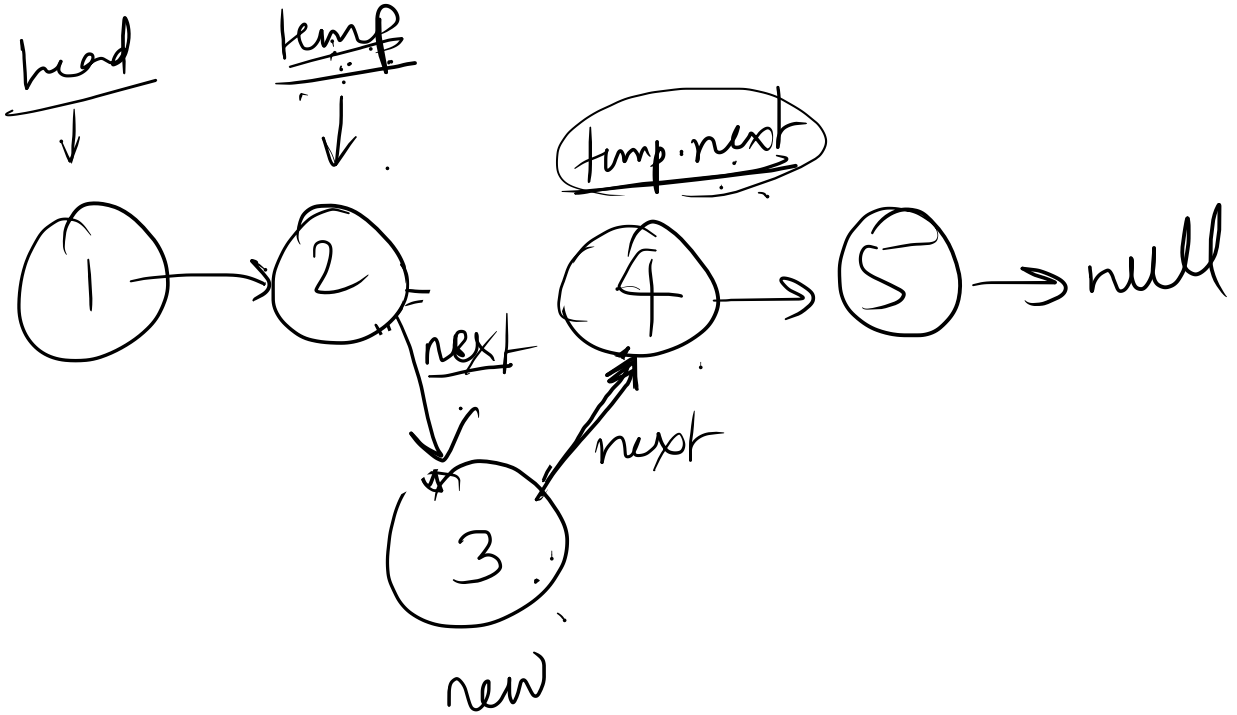
O(1)

new.next = head

head = new

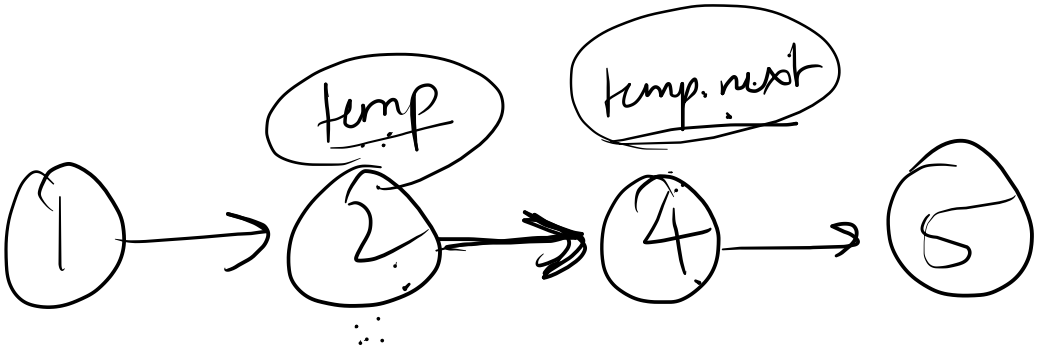
node.next ✓

node.prev ✗



new.next = temp.next

temp.next = new



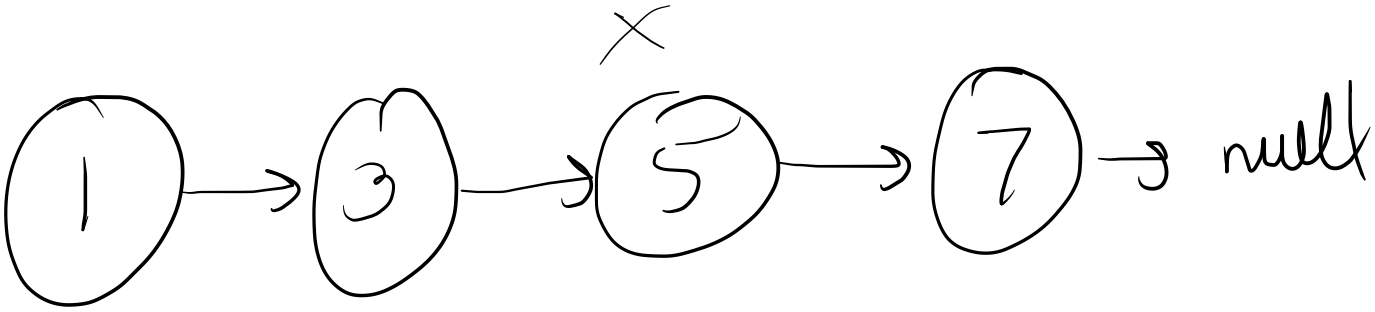
3

new

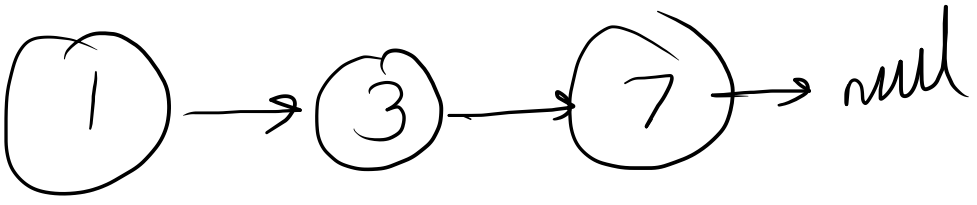
O(1)

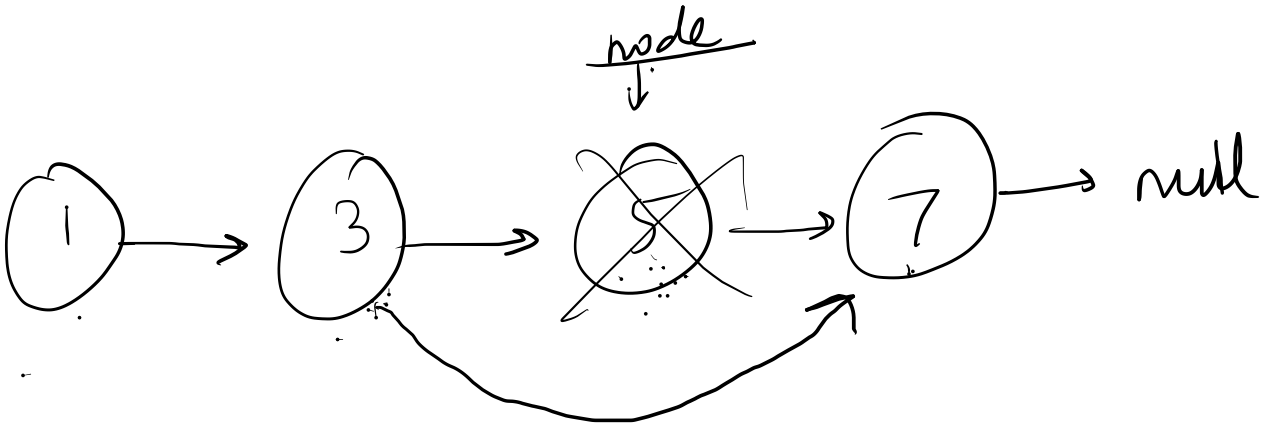
new.next → temp.next

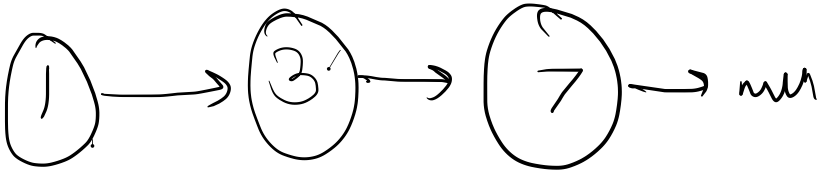
temp.next → new

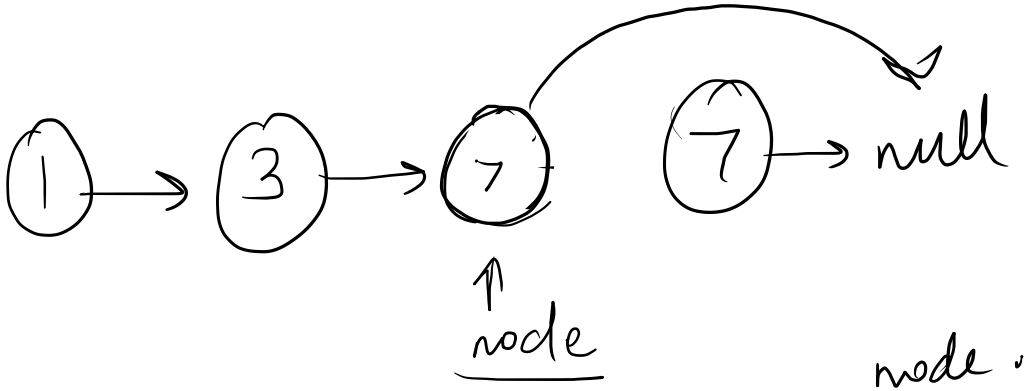


delete node 5



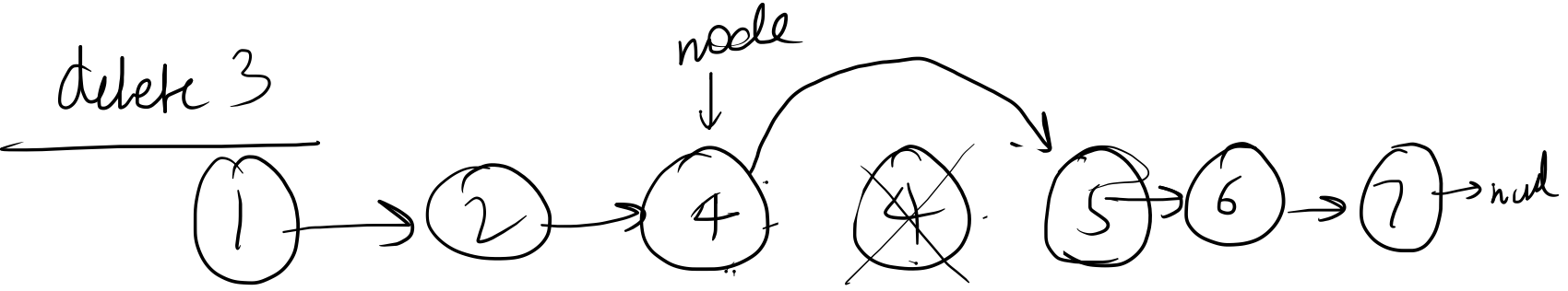

node
1
3
5
7
null


1
3
7

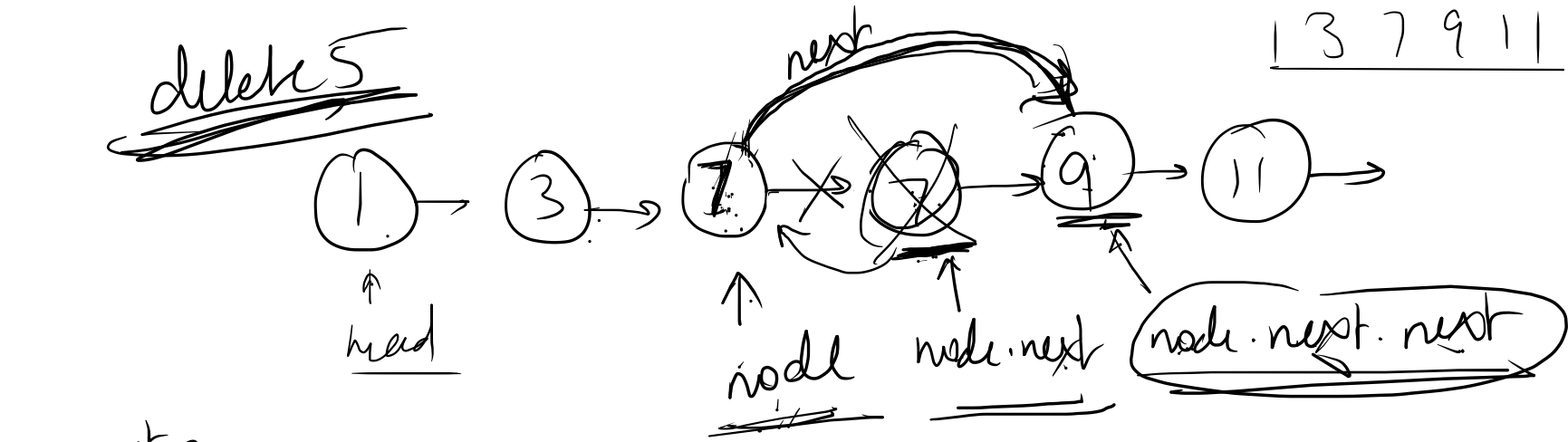


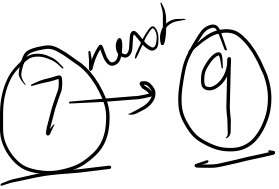
node.value = node.next

delete 3



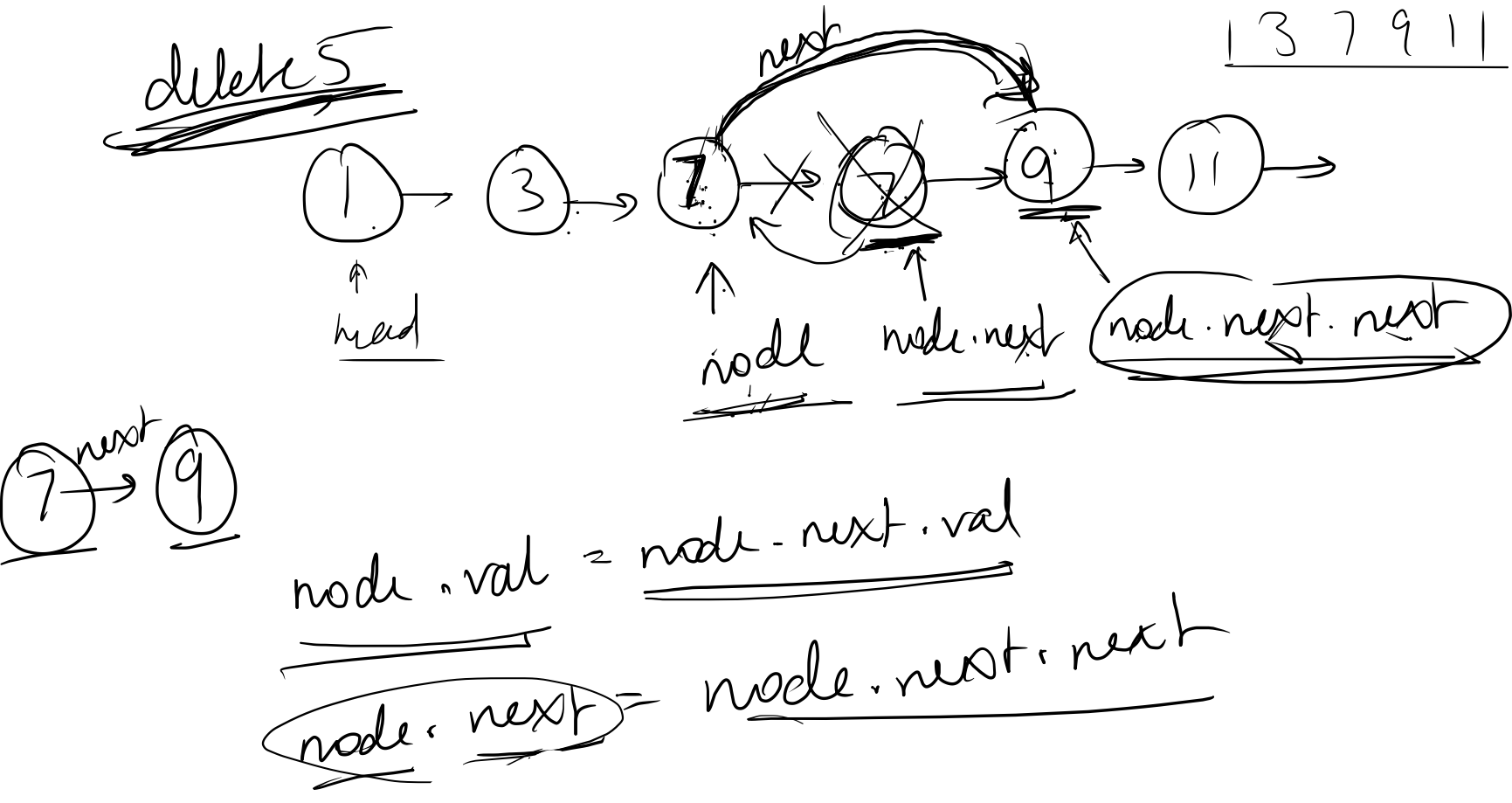
① Replace ③ with ④





node.val = node.next.val

node.next = node.next.next

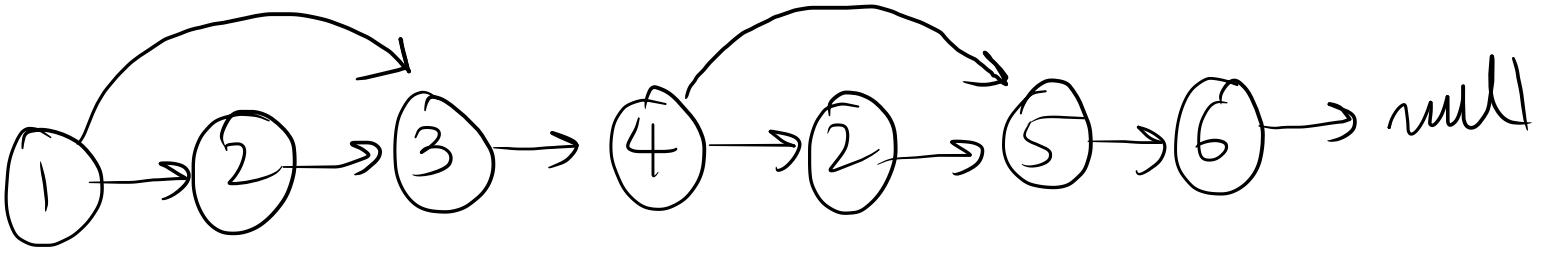

delete 5
1 3 7 9 11
next
1
3
7
9
11
head
node
node.next
node.next.next
7
next
9
node.val = node.next.val
node.next = node.next.next

constant time

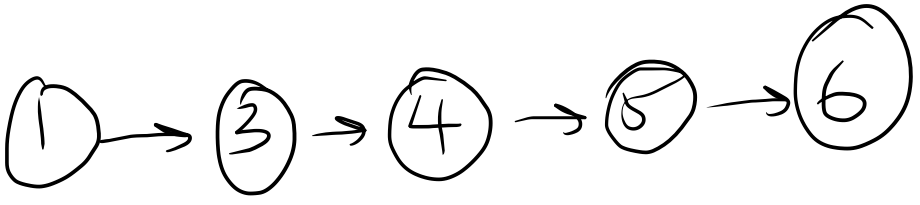
T.C = O(1)

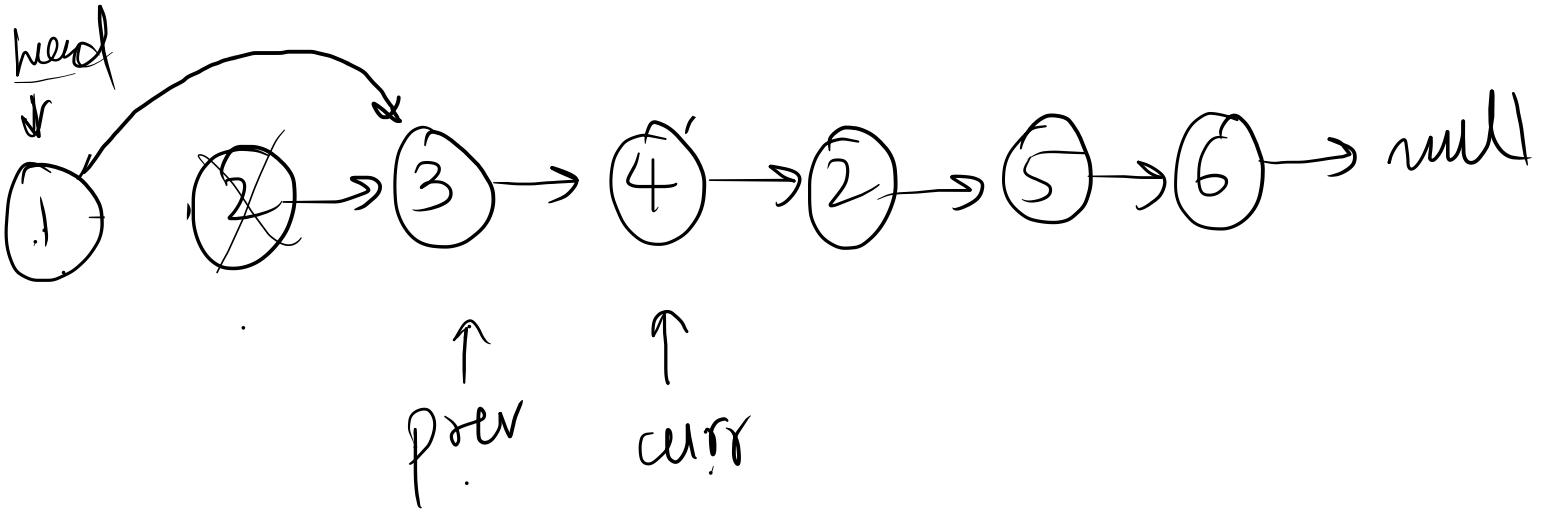
Auxiary space: O(1)

2 mins Break



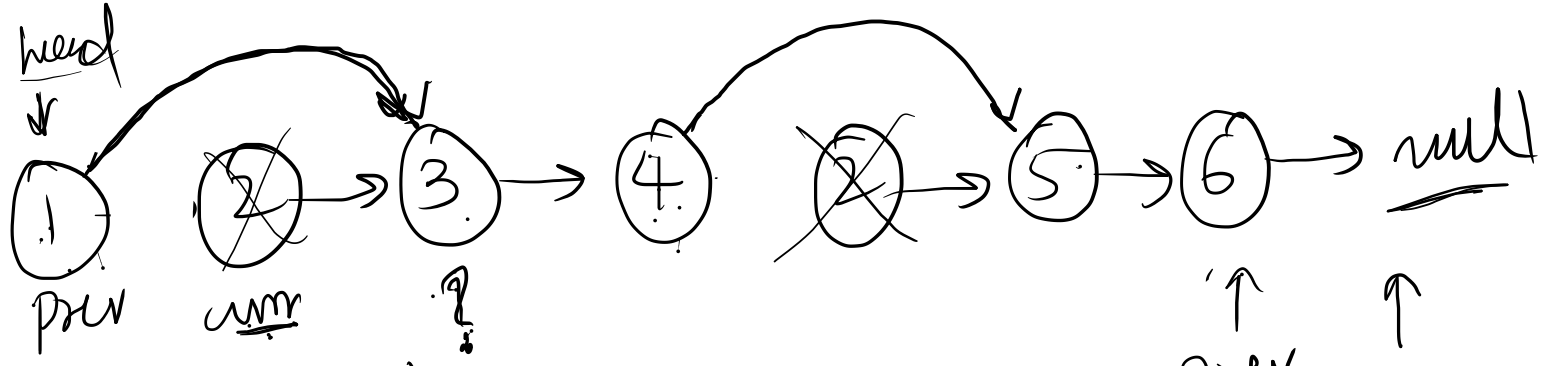
Remove 2



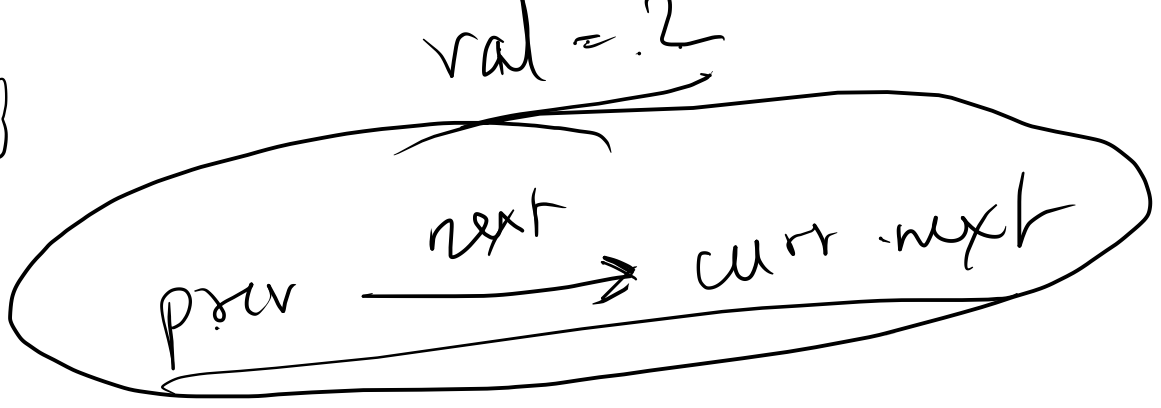


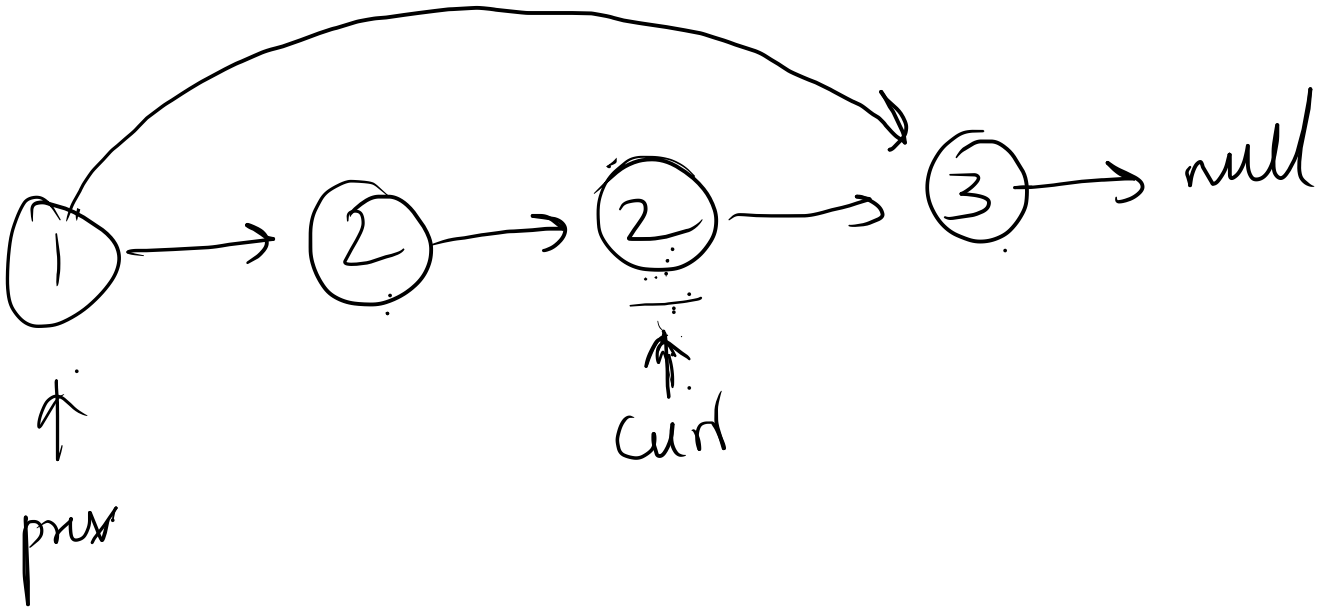
val = 2

prev → next

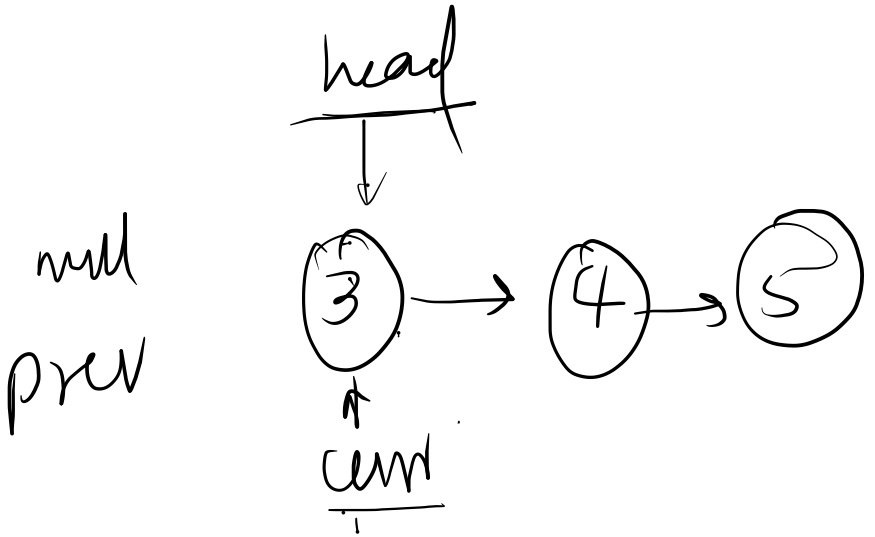


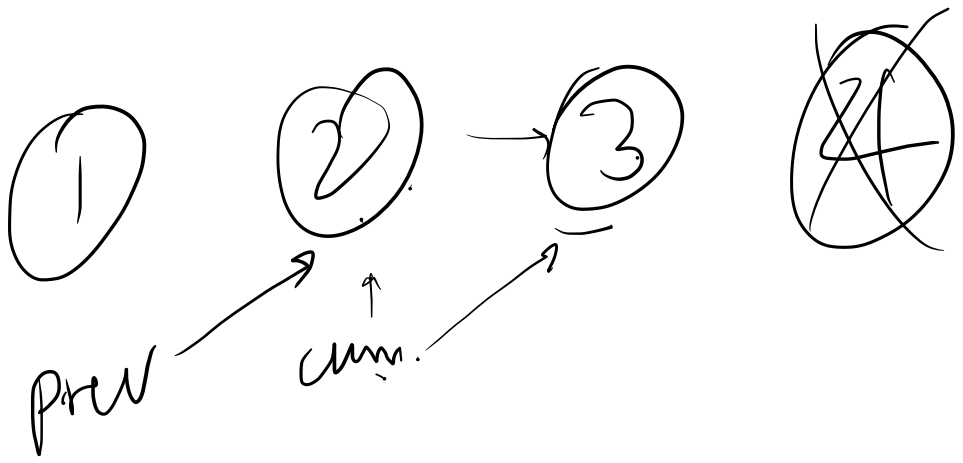

```
while( curr != null)
{

}
```

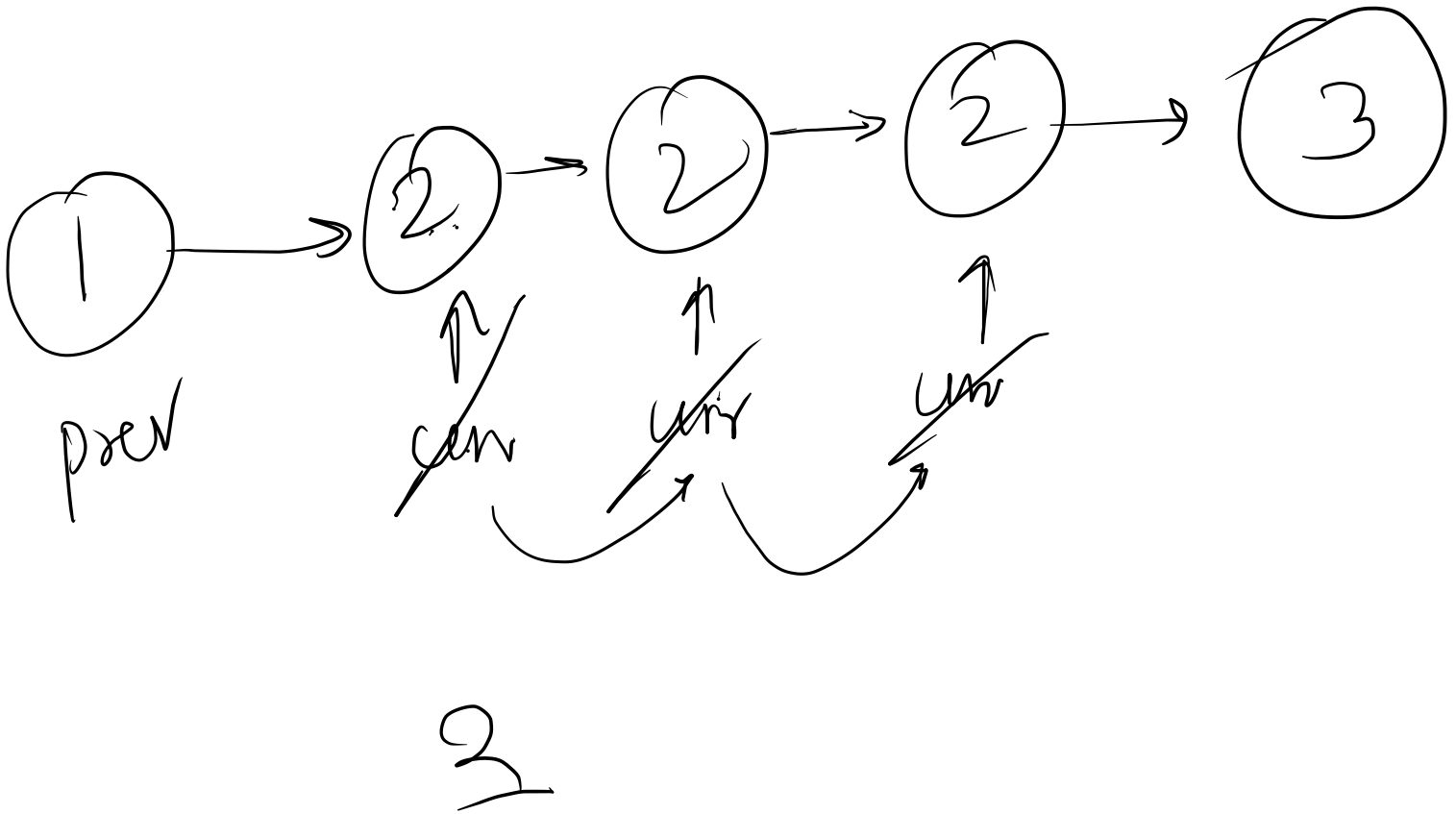




val = 2

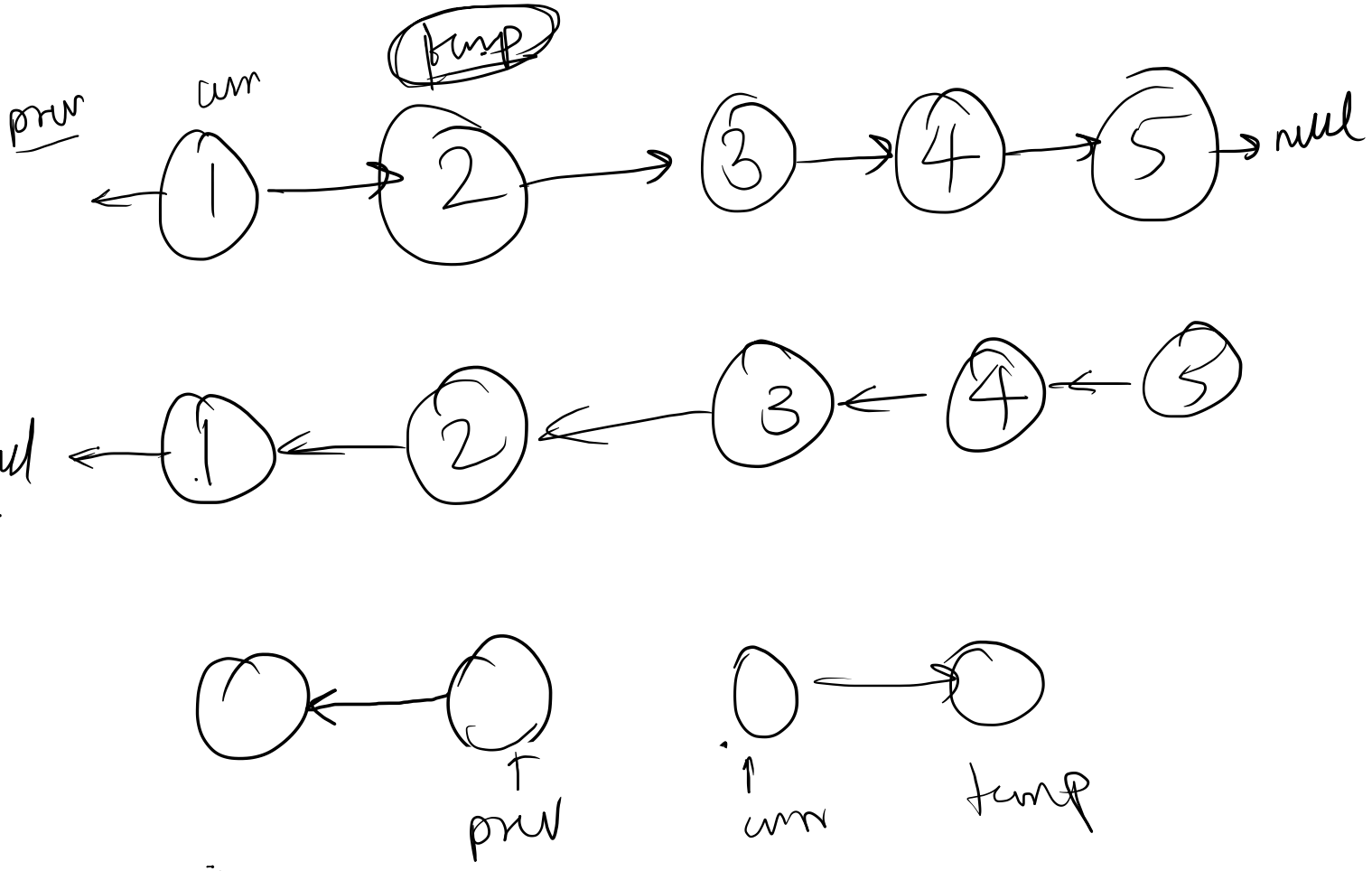

head
null
prev
3
4
5
curr


1
2
3
4
prev
curr

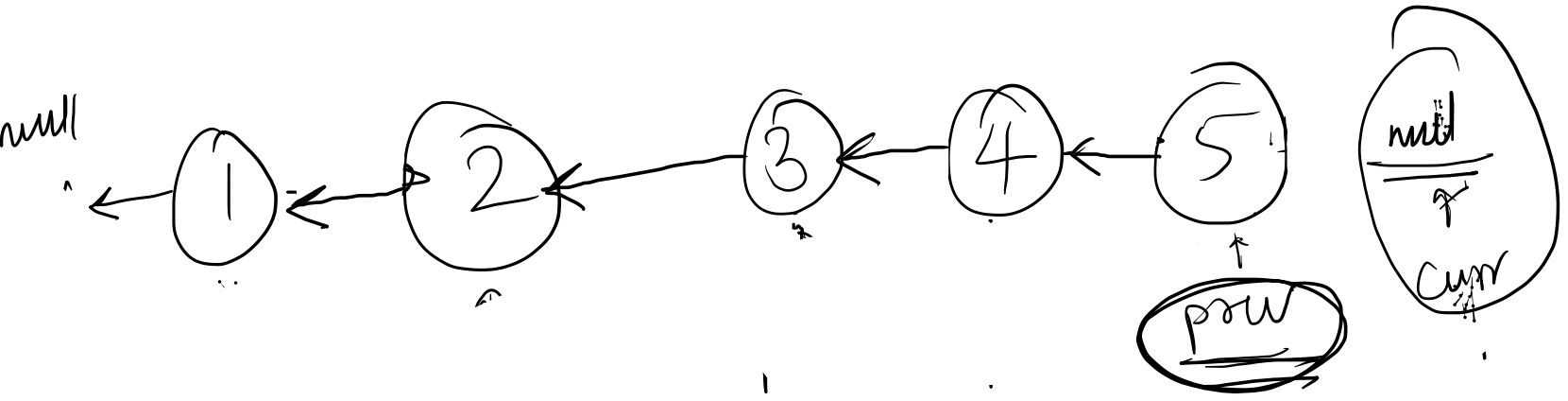

1
2
2
2
3
prev
curr
curr
curr

$TC = O(n)$

Auxiliary space $= O(1)$


temp
prev
curr
1
2
3
4
5
null
null
1
2
3
4
5
prev
curr
temp

temp = curr.next



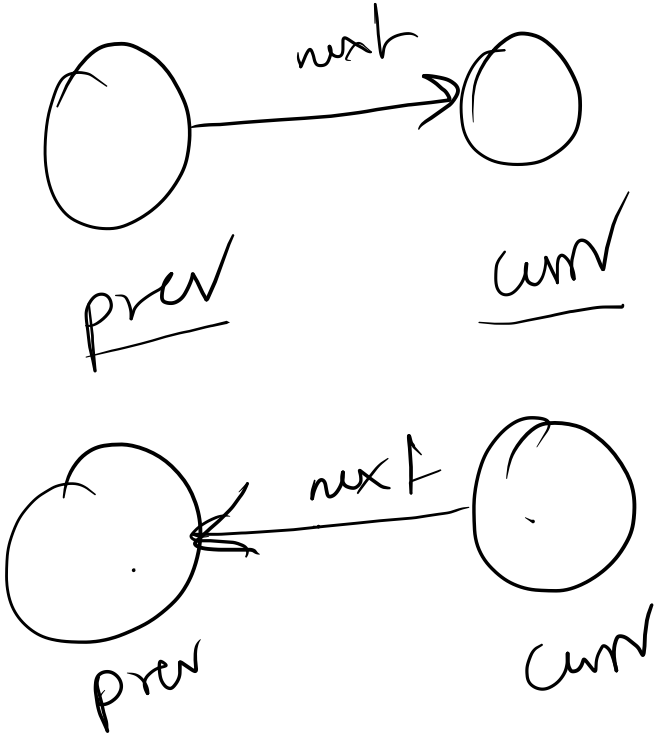
curr.next = prev

while(curr != null)
{

}

TC = O(n)

Aux. space = O(1)


next
prev
curr
next
prev
curr