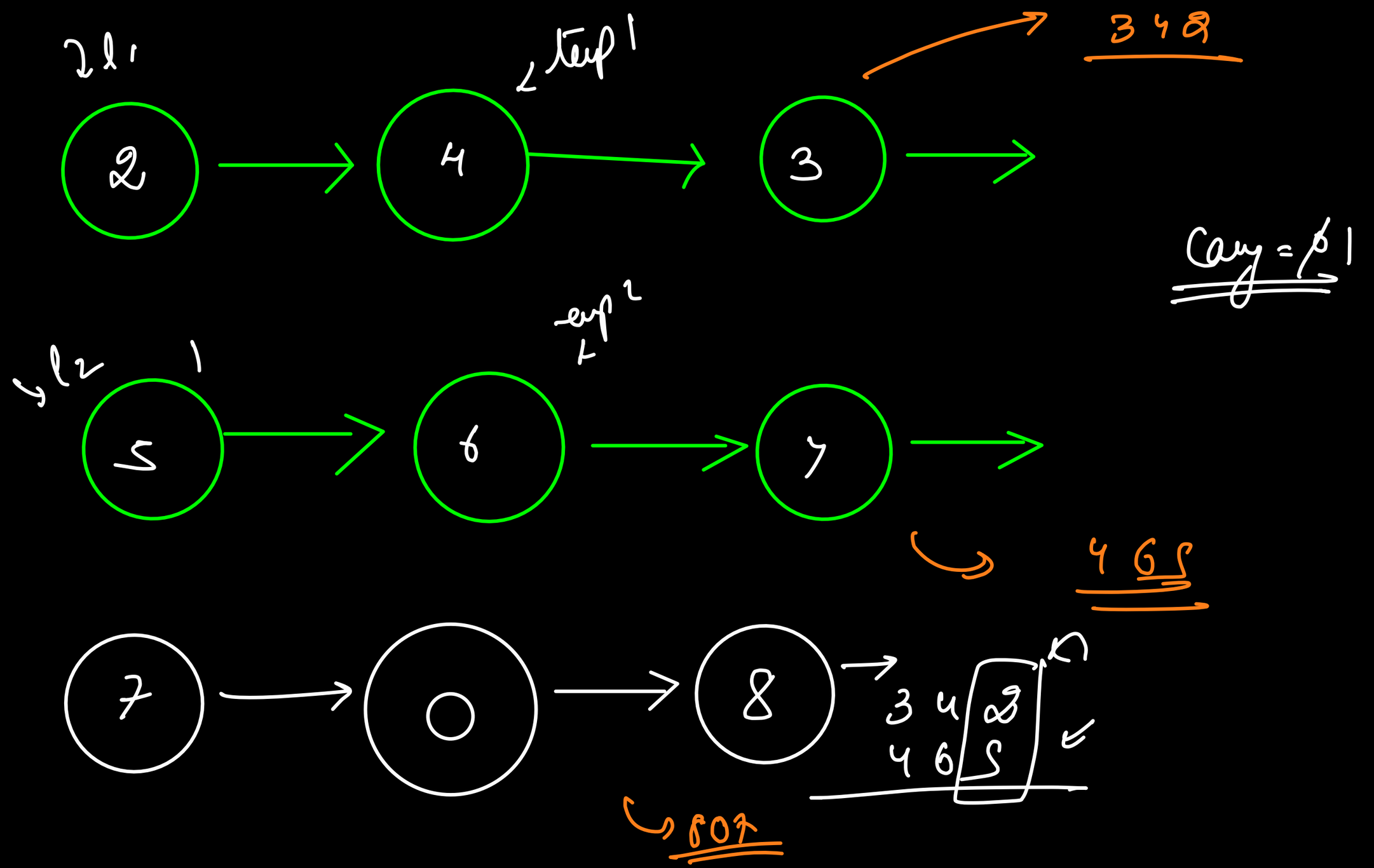
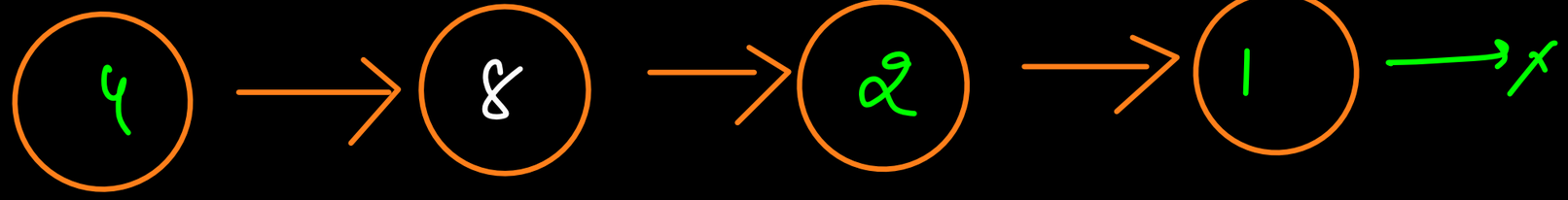
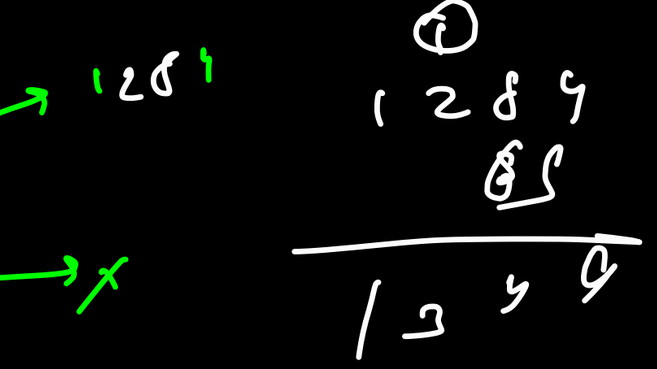
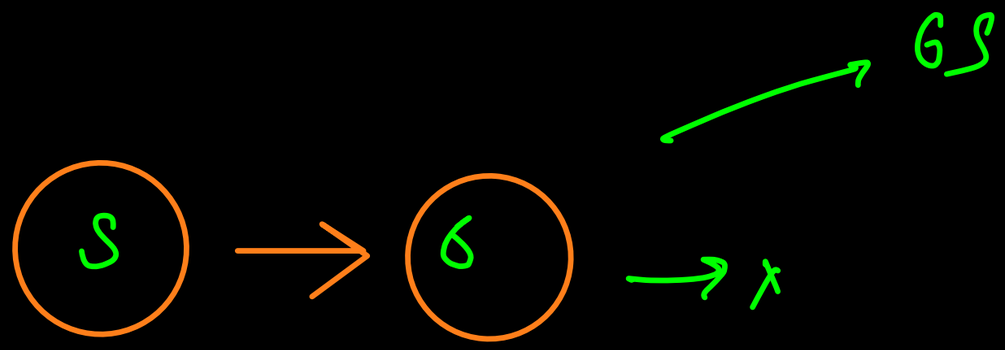
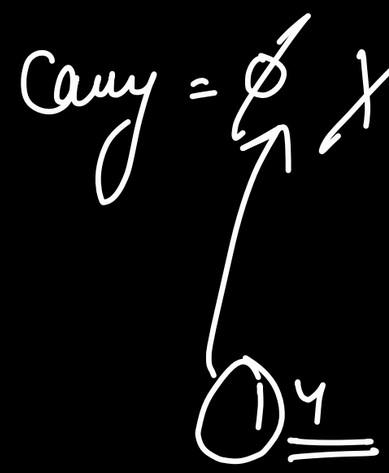
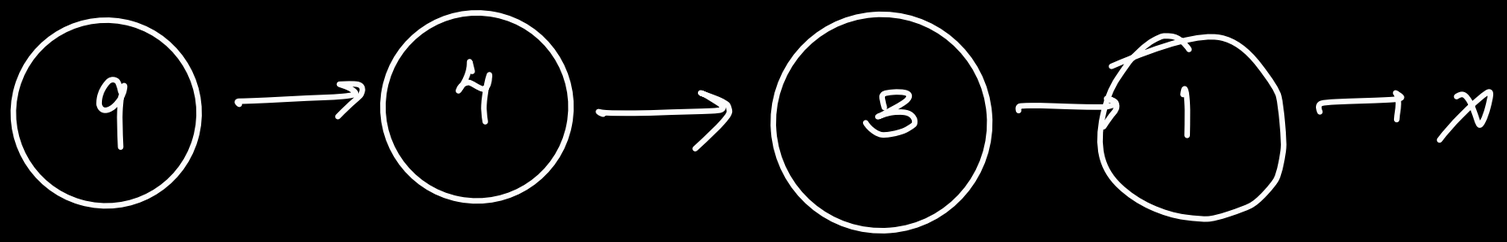


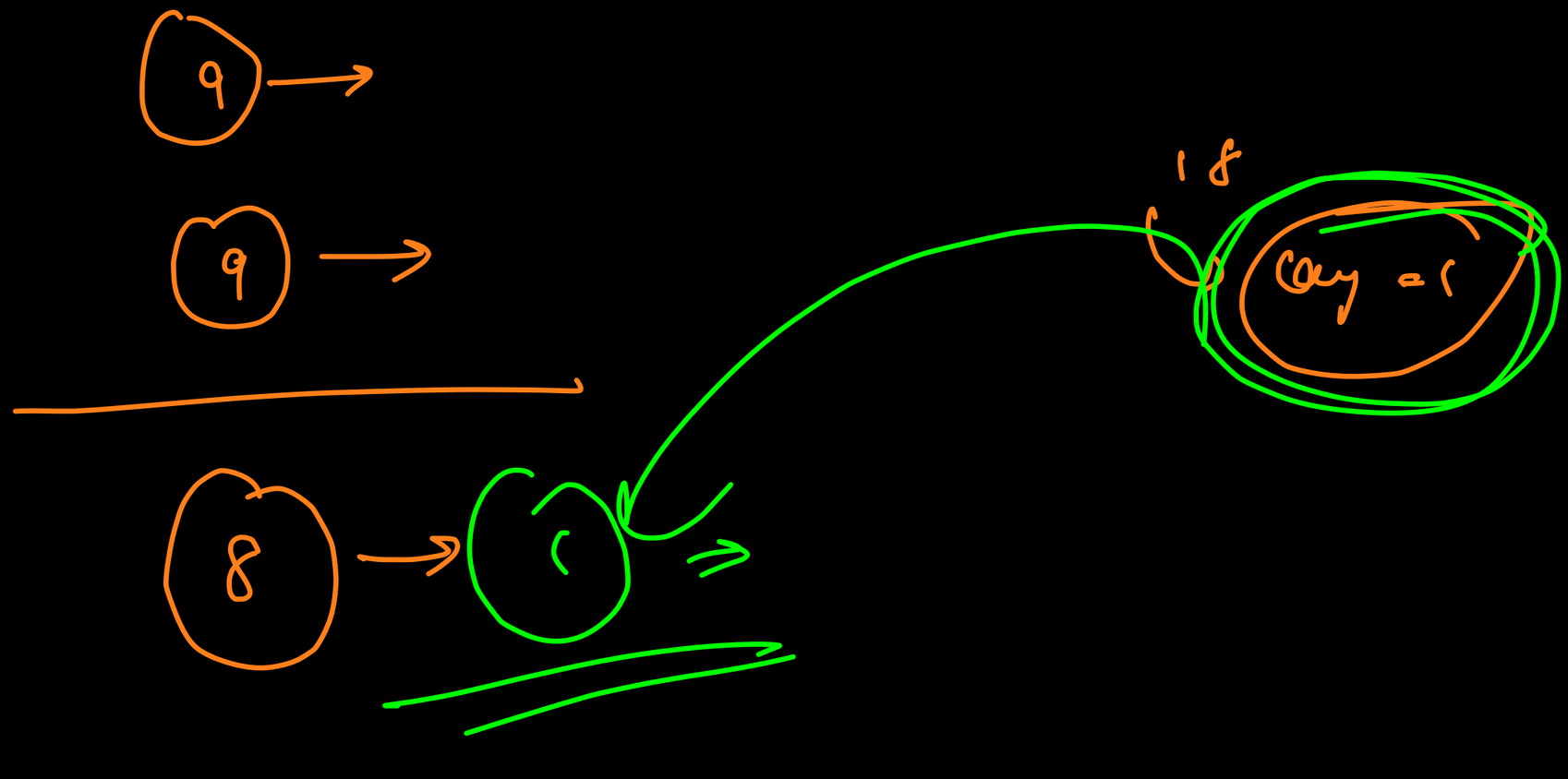




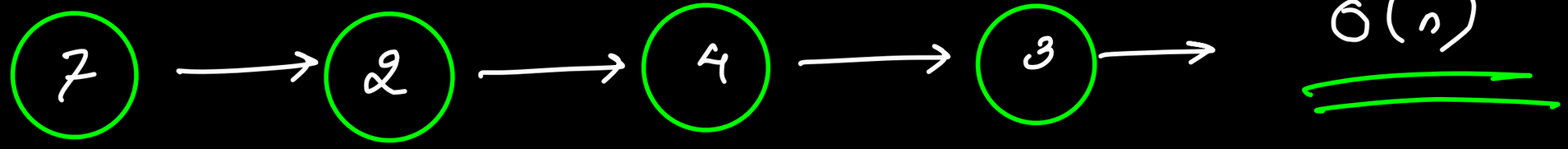








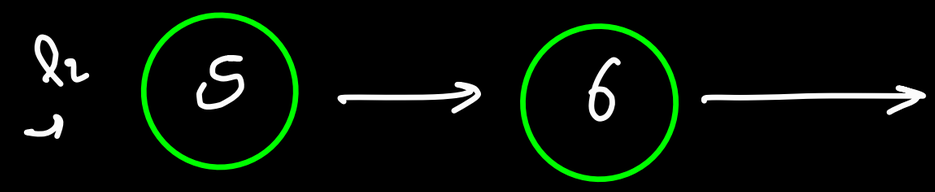
2 ll:



O(n)

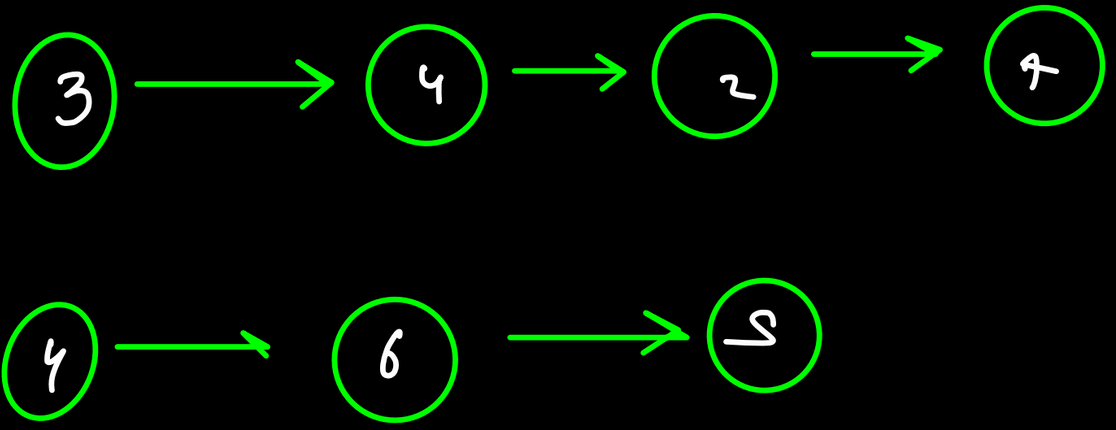
iterative
Recursive

l2 →

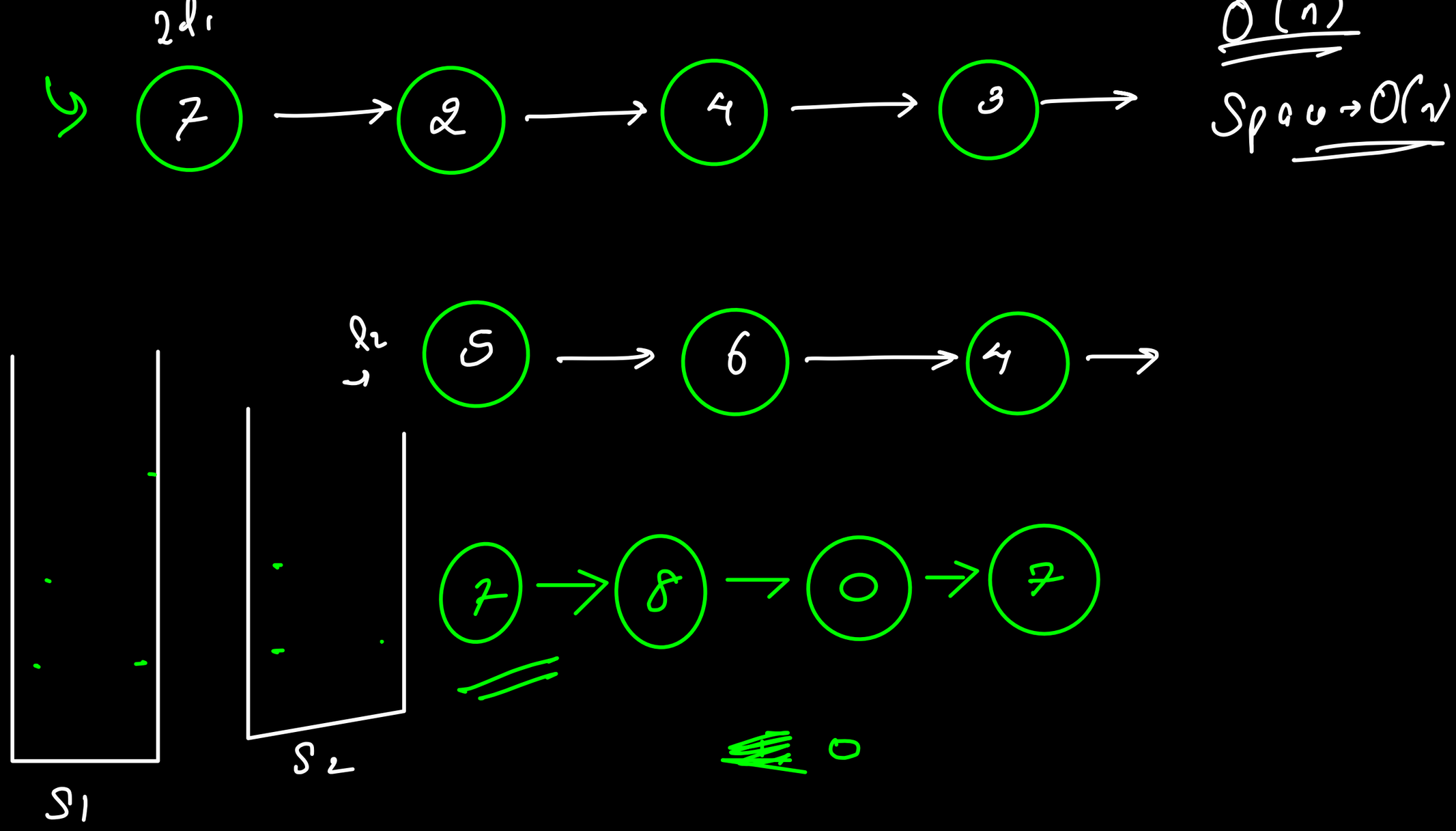


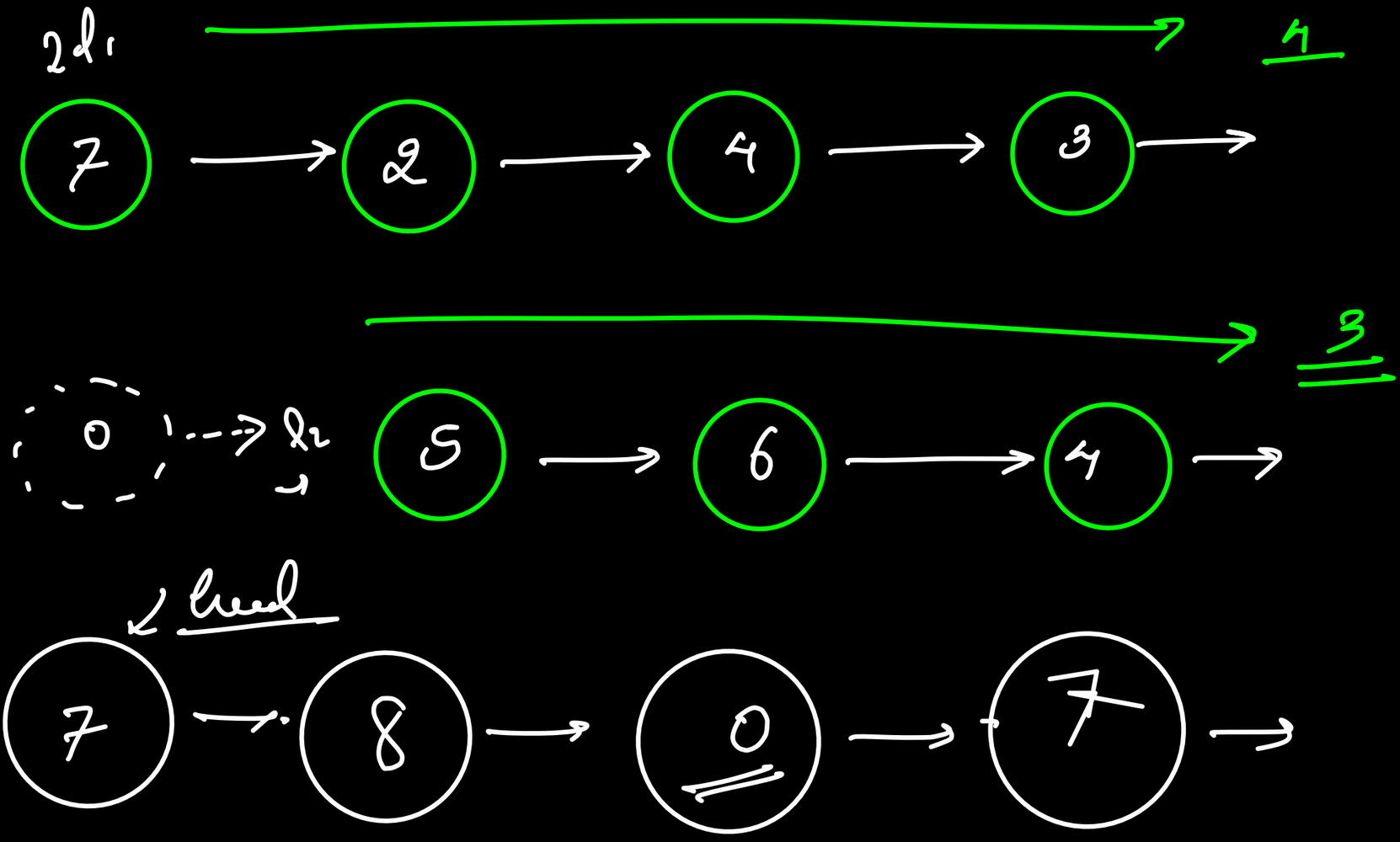
O(n)

#A·1 →



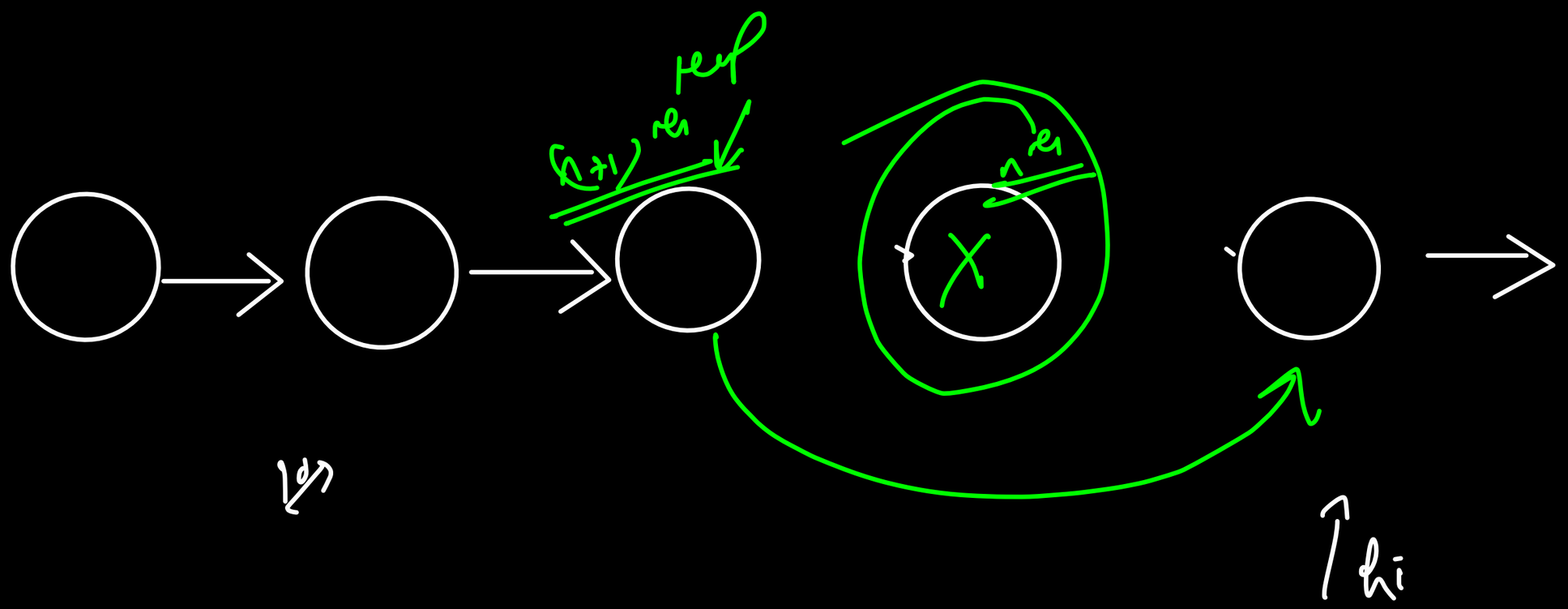
O(1)



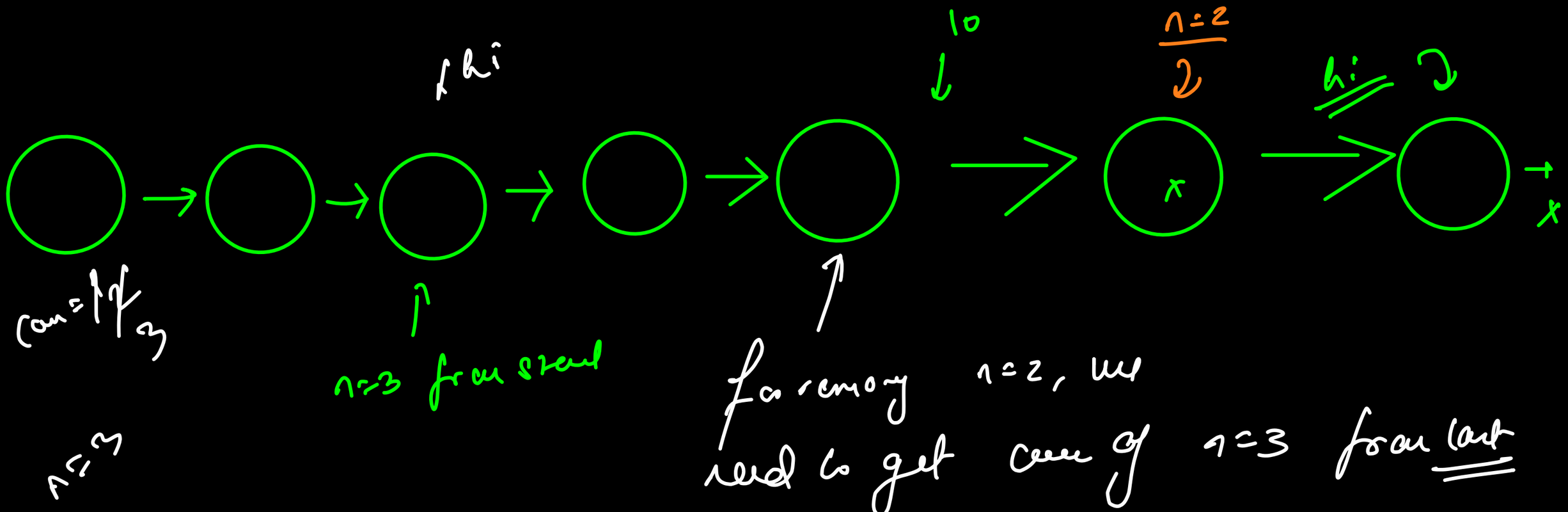


4 - 3 → 1

O(n)



n=4



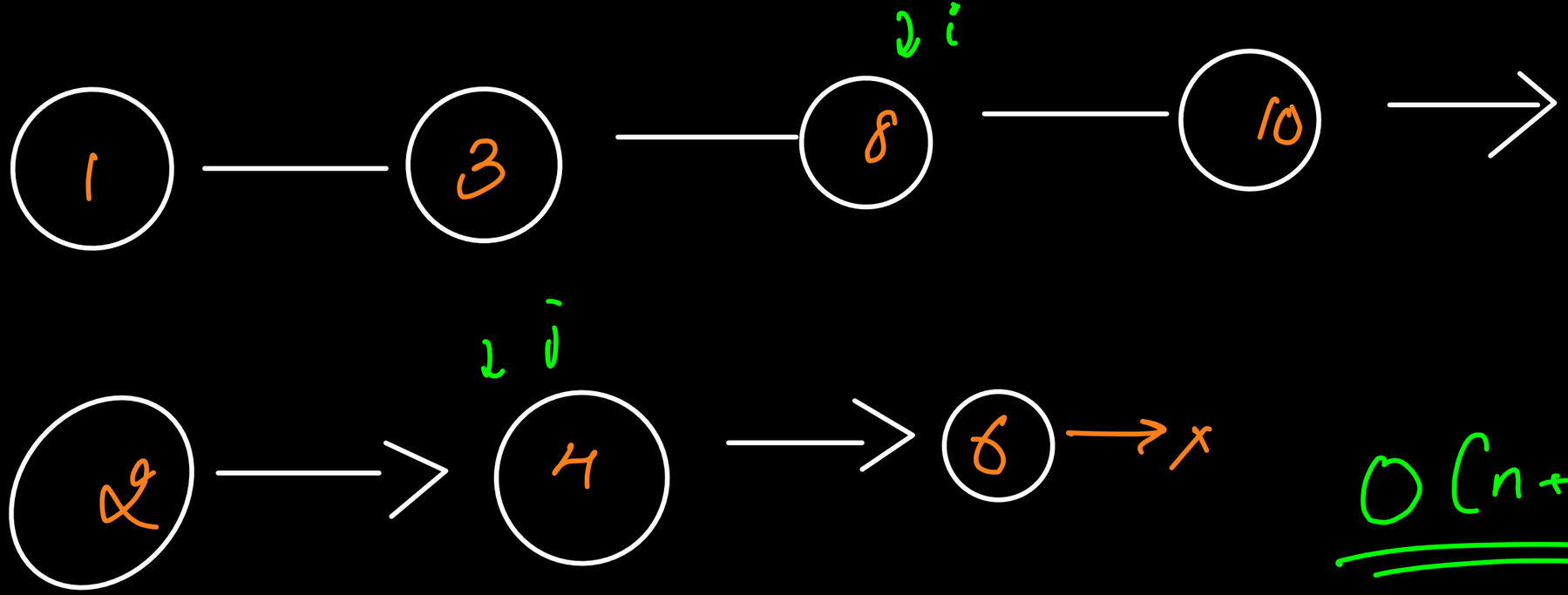
2 pointers

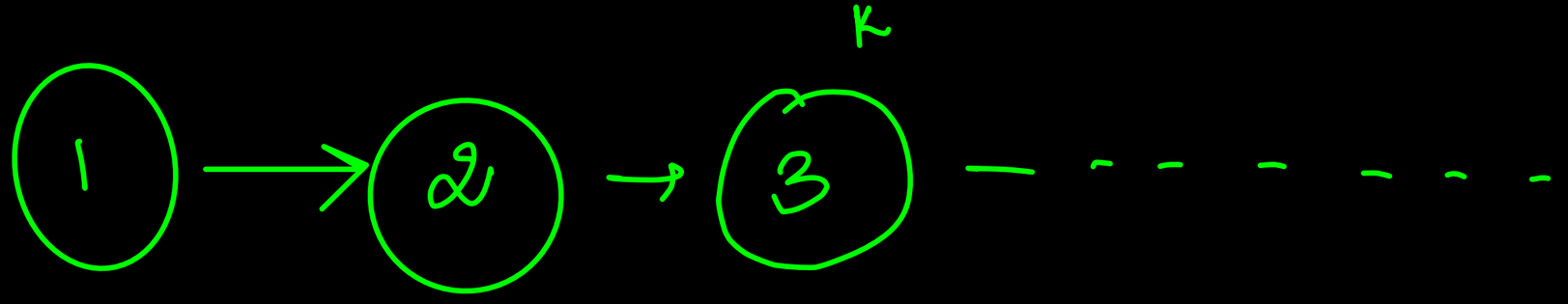
O(n)

O(1)

Merge 2 sorted arrays

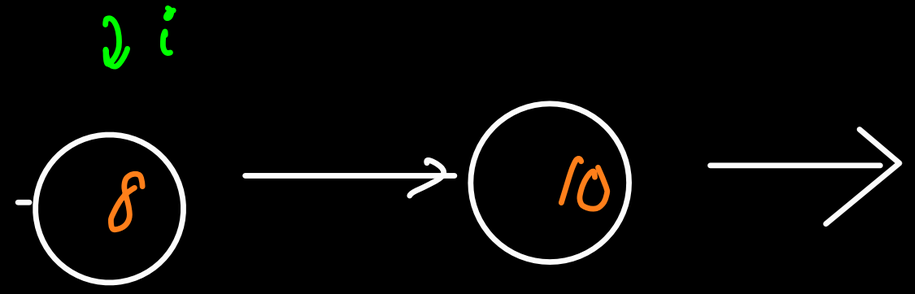
→ 2 pointers



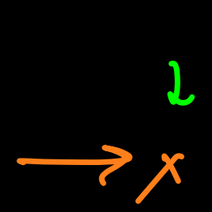


$O(n+m)$

$O(1)$



temp.next = i



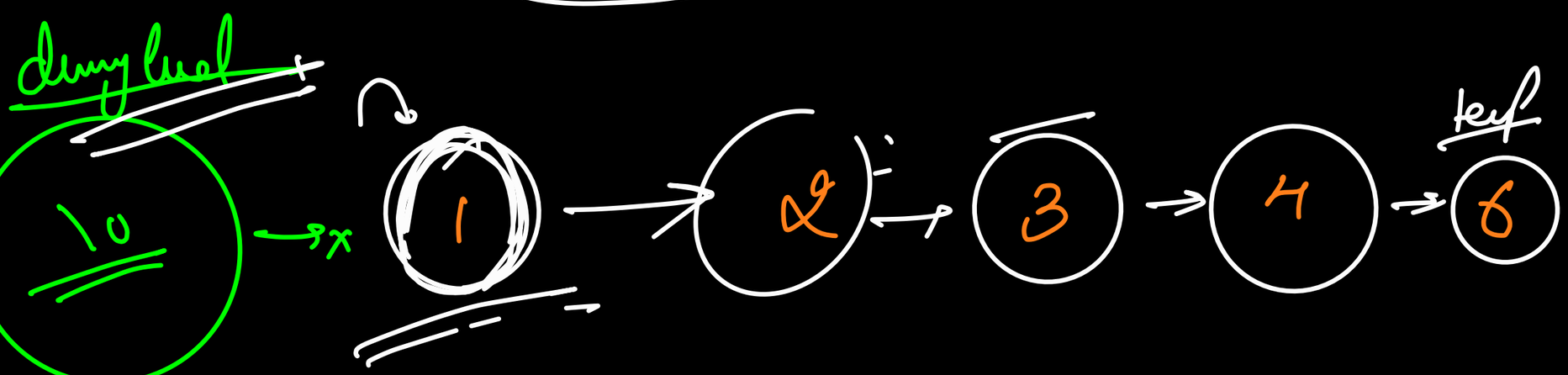
temp.next = i

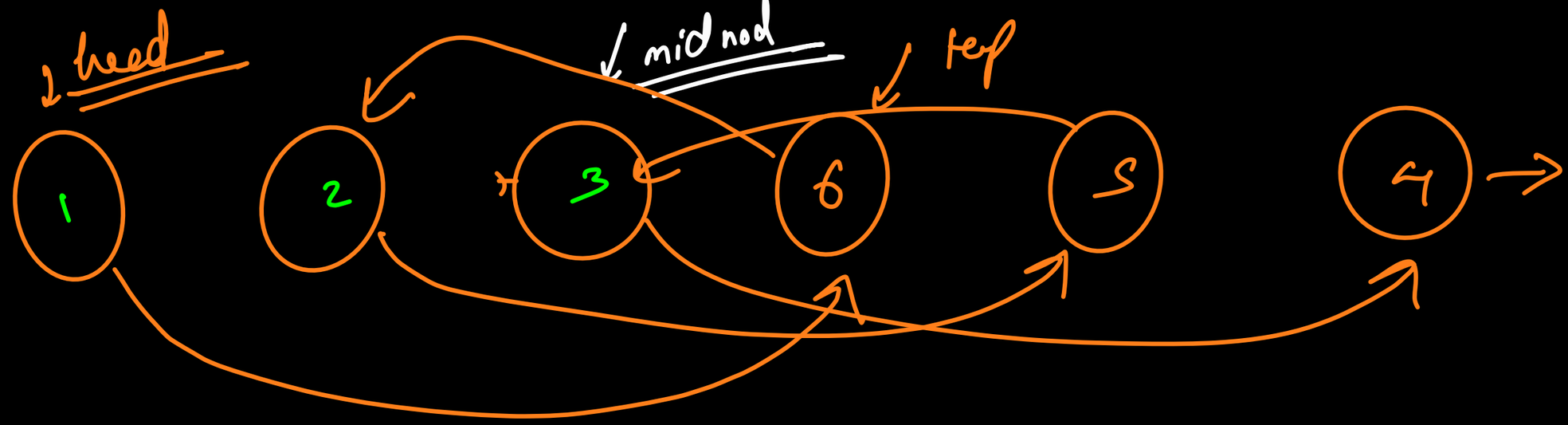
temp = temp.next

i = i.next

temp.next = null

return dummyHead.next





1) go & calc midnode → O(n)

fast = midnode.next.next

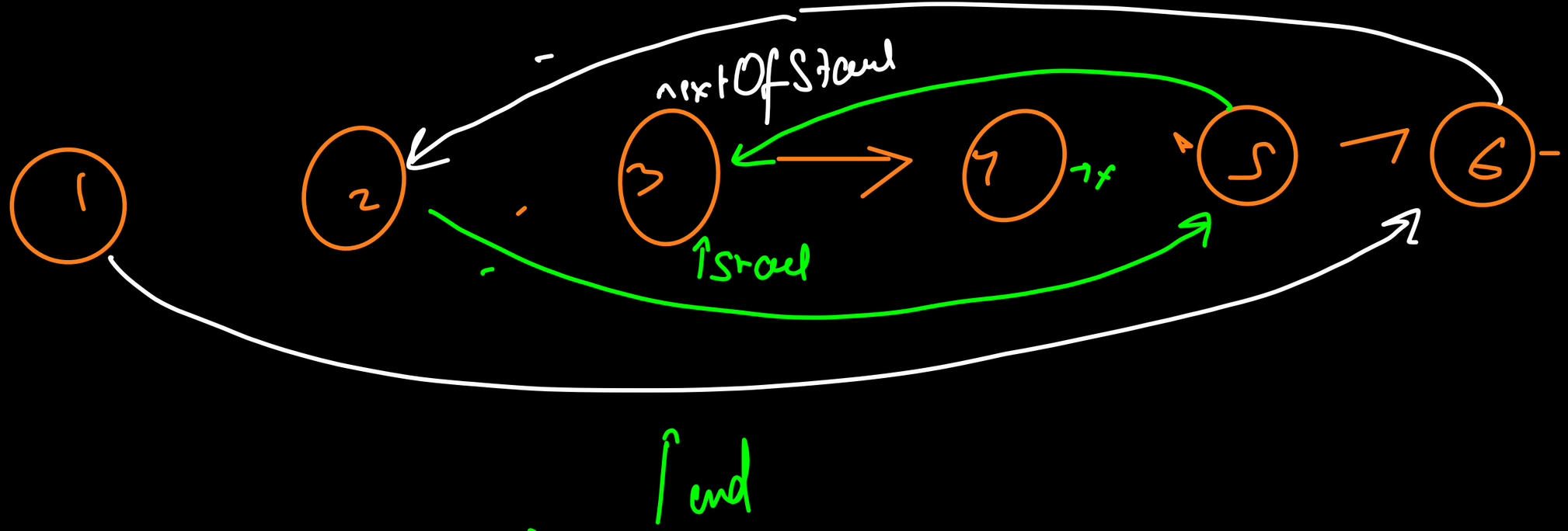
midnode.next = null

O(n)

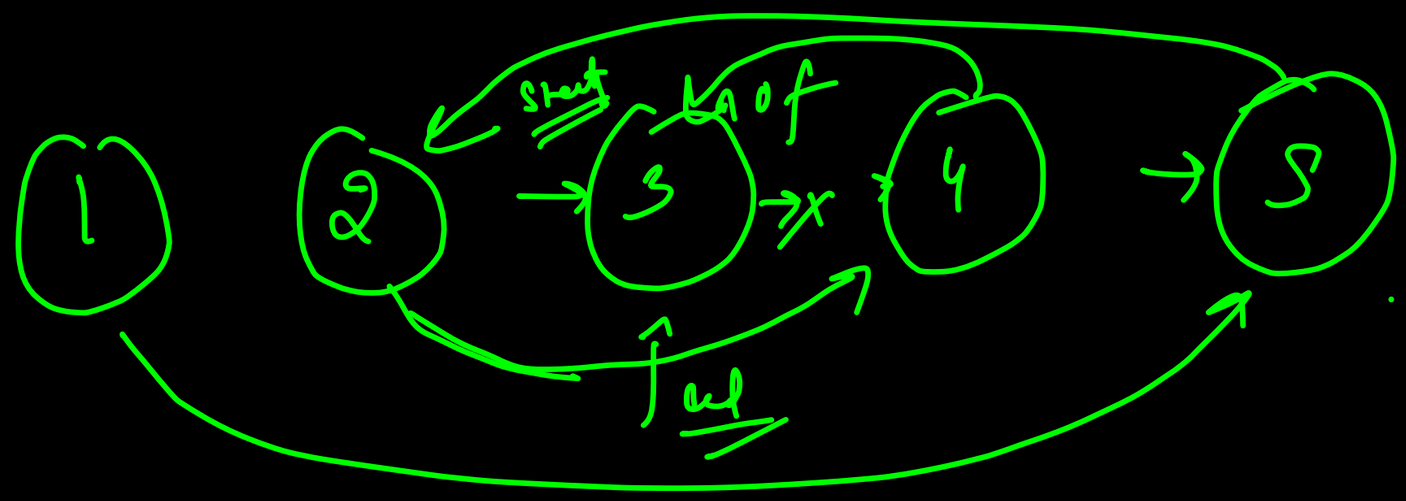
(2) Reverse the 2nd half → O(n)

(3) Merge → O(n)

O(1)




```
if (start.next == end)
    end.ext = null
nextOfStart = start.next
start.next  = end
end.next    = next != null ? next
start = next of start2
return n
```



if (start.next == end or start == end)
    end.next = null

nextOfStart = start.next

start.next = end

end.next = next != start ? start

start = next of start

return start

O(n)

O(1)