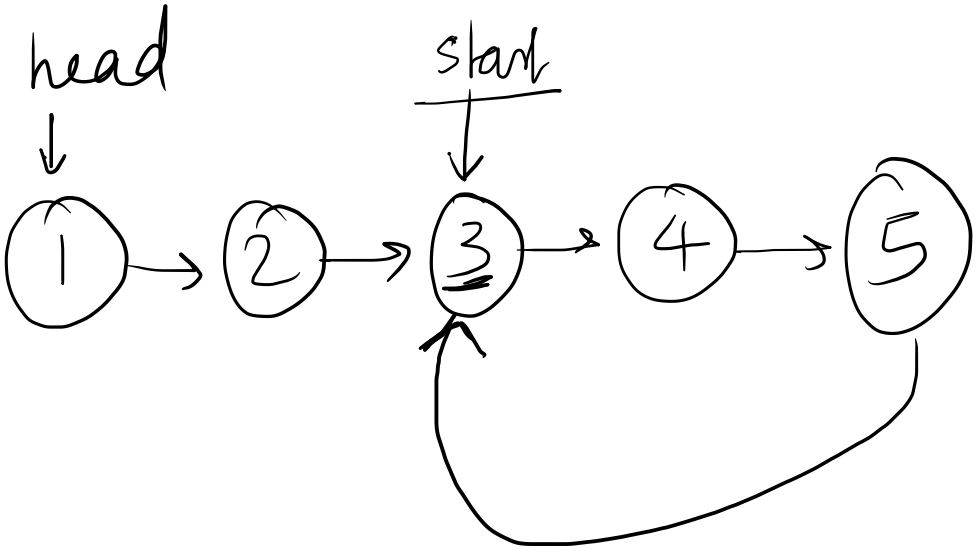
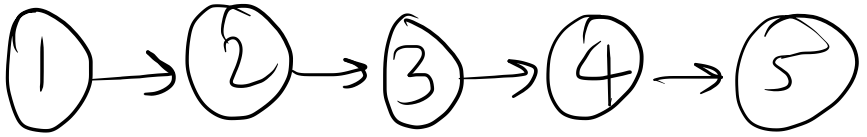
LINKED LIST


head
start
1
2
3
4
5



No cycle → return null

1. Detect whether cycle exist

2. If cycle exisks,
   ↳ find the start node

   else
   ↳ return null

1. Detect whether cycle exist

2. If cycle exisks,
   ↳ find the start node

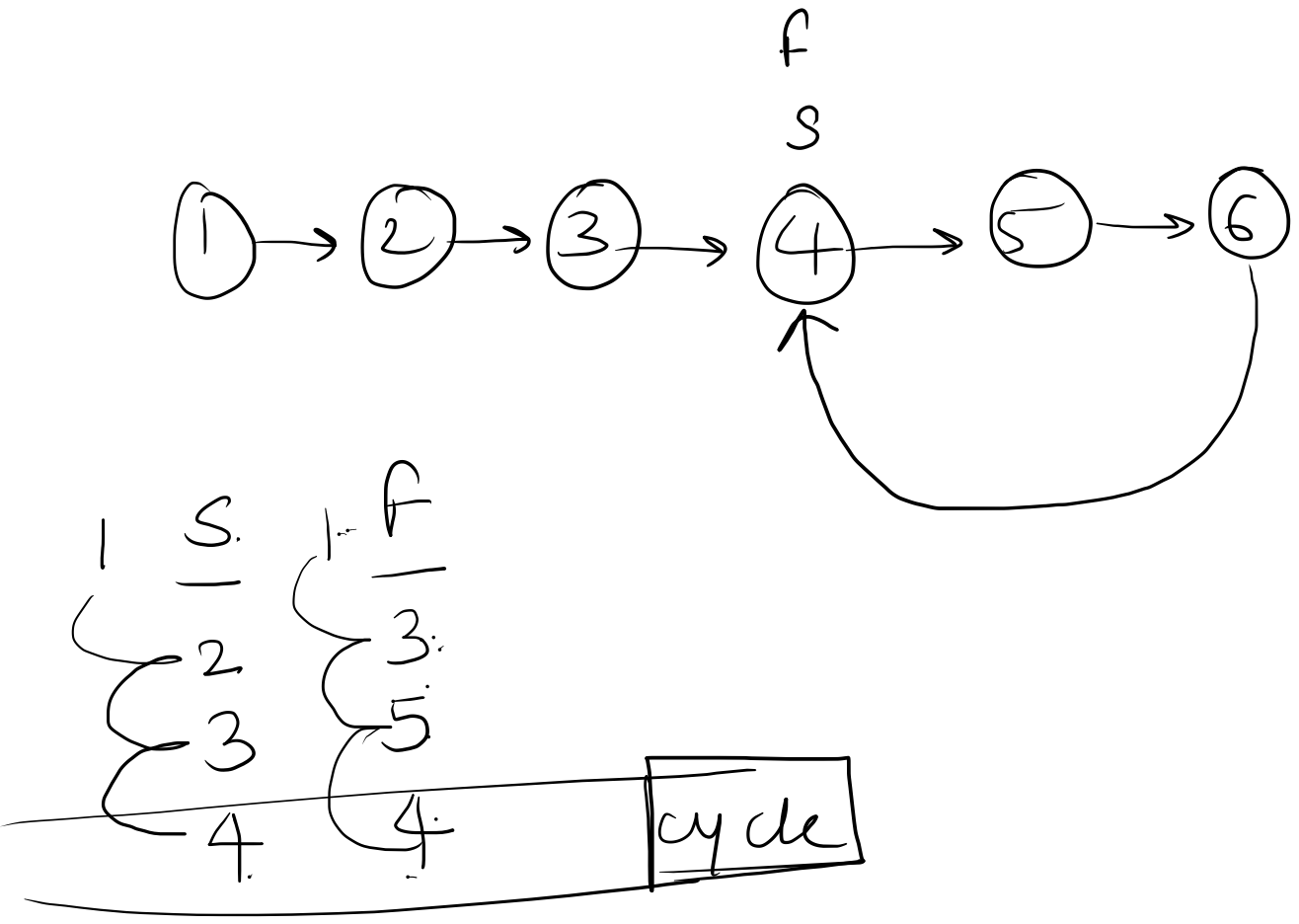
   else
   ↳ return null

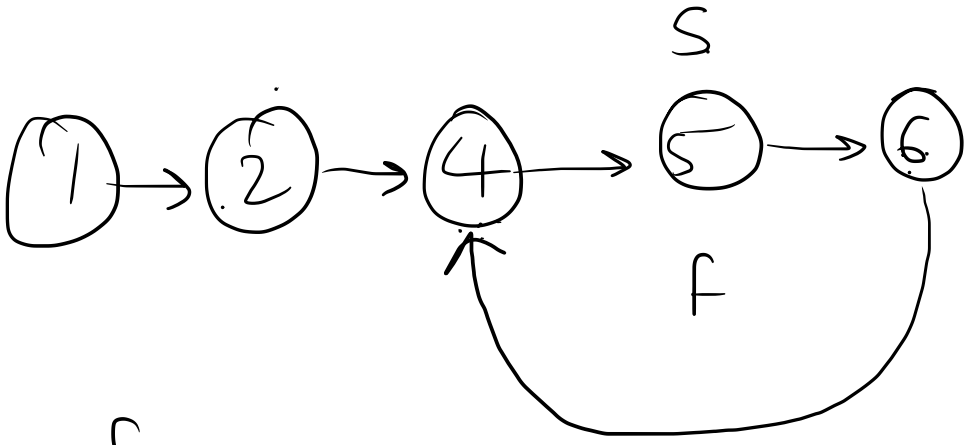
Detect cycle

↳ Two pointers Approach

Slow fast ptr app.

Hare & tortoise app

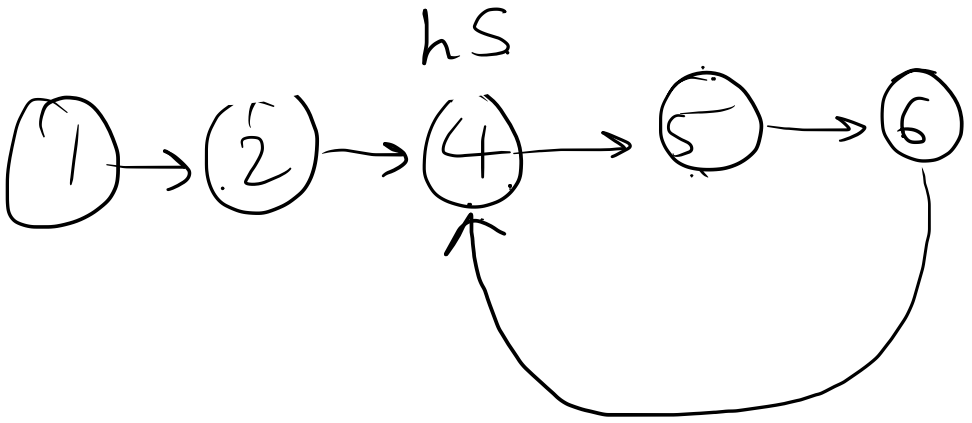

f
s
1
2
3
4
5
6
s
1
2
3
4
f
1
3
5
4
cycle



S	f
1	1
2	4
4	6
5	5

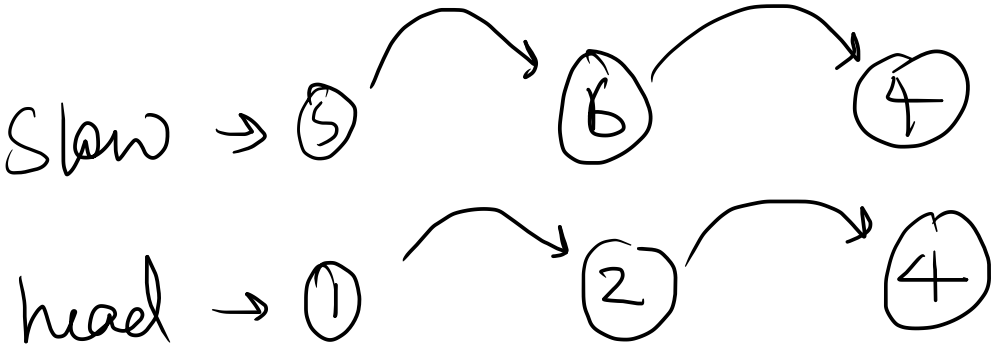
+2 +2 +2

cycle ✓



when (s = h)

return s



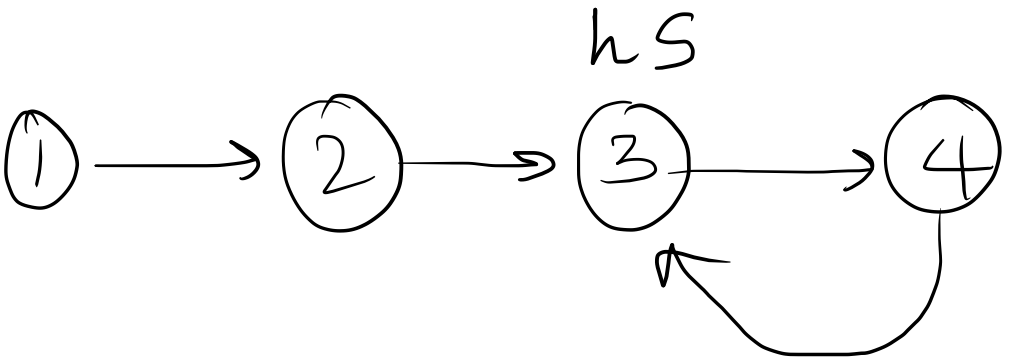
① slow = fast

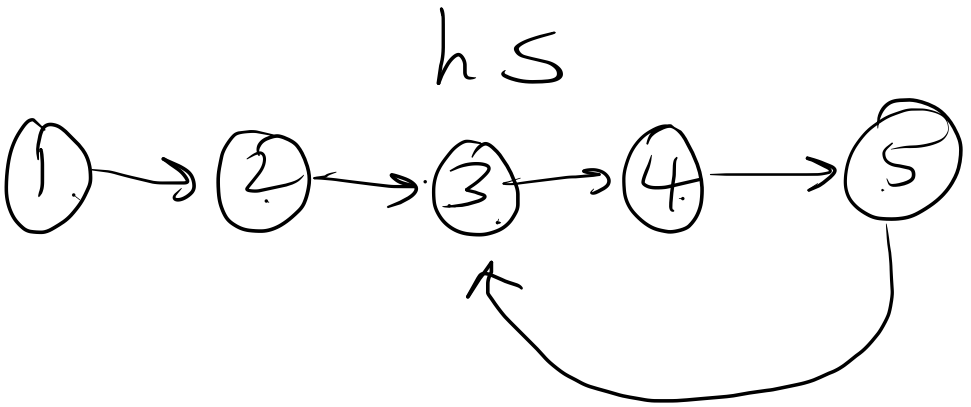
cycle detected

② move (head, slow)

when head = slow

ans = head/slow


hs
1
2
3
4



s	f
1	1
2	3
3	5
4	4

cycle

s = f

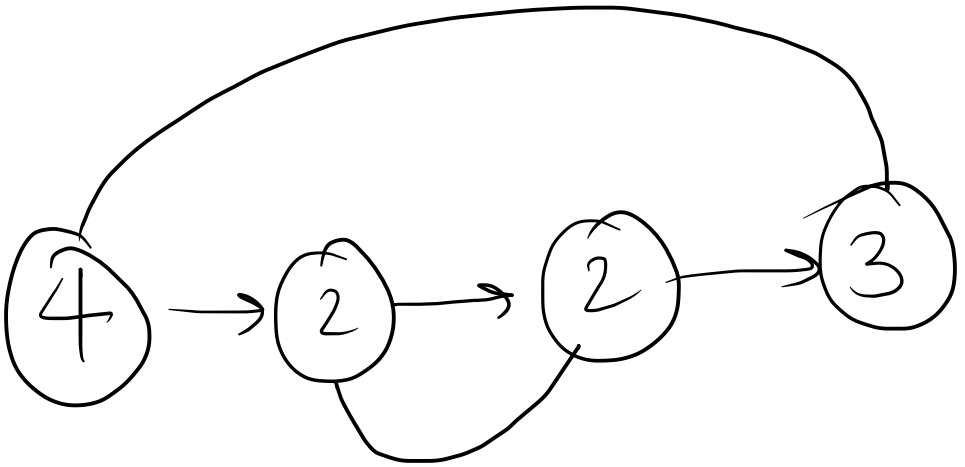
h	s
1	4
2	5
3	3

start

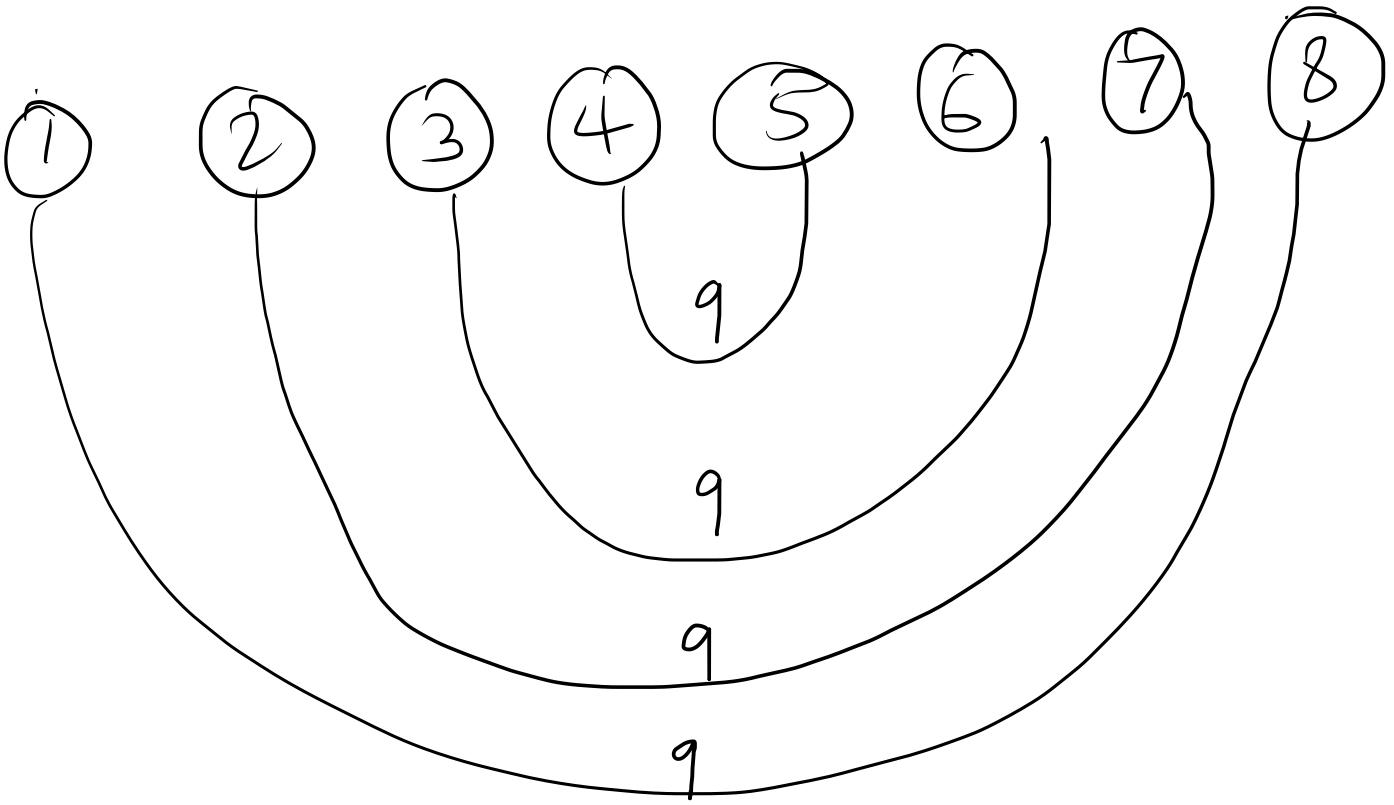
s = h

$TC = O(n)$

Aux Space = $O(1)$

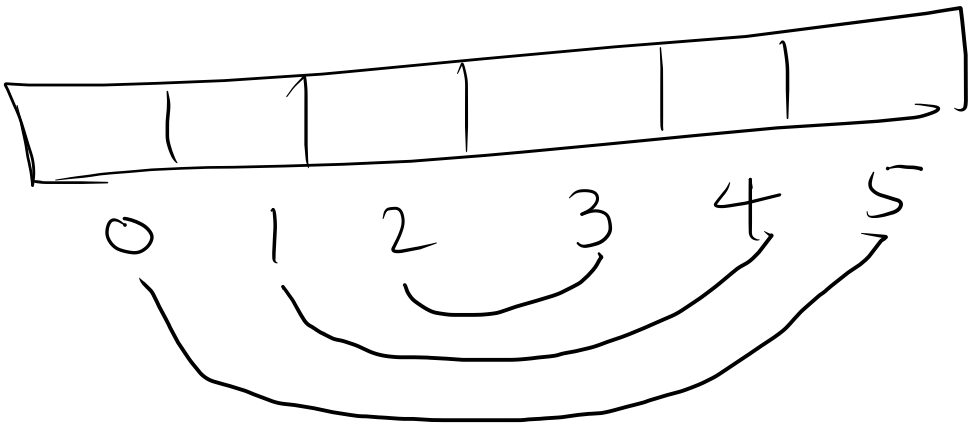

4
2
2
3

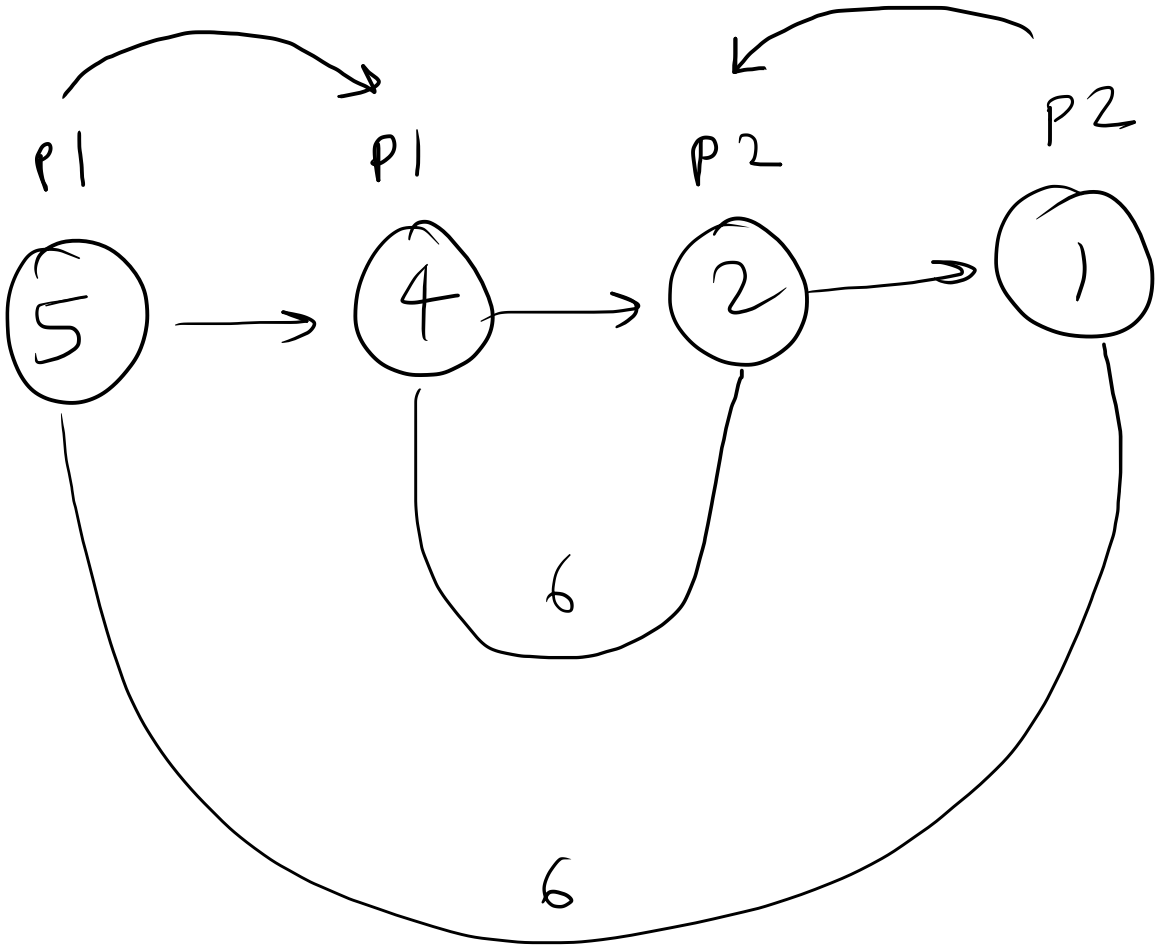
max sum = 9



n = 6

i	n-1-i
0	5
1	4
2	3



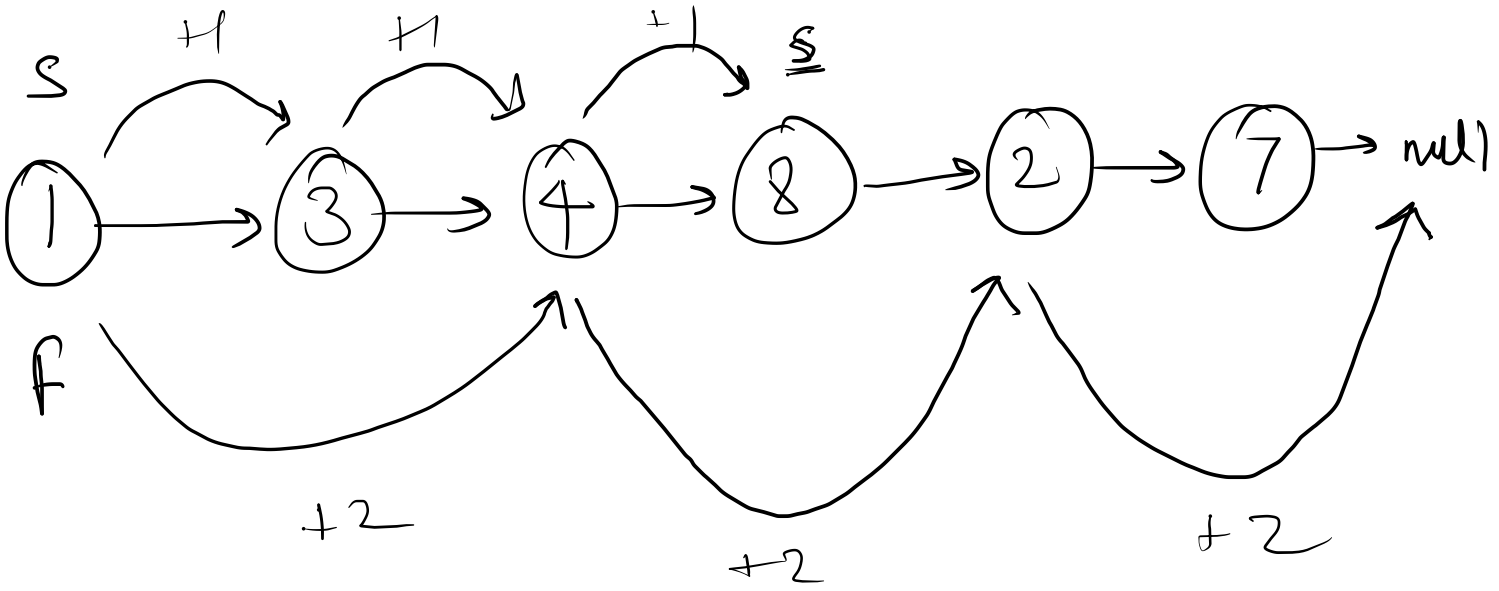

P1
P1
P2
P2
5
4
2
1
6
6

✓ ① Find the middle

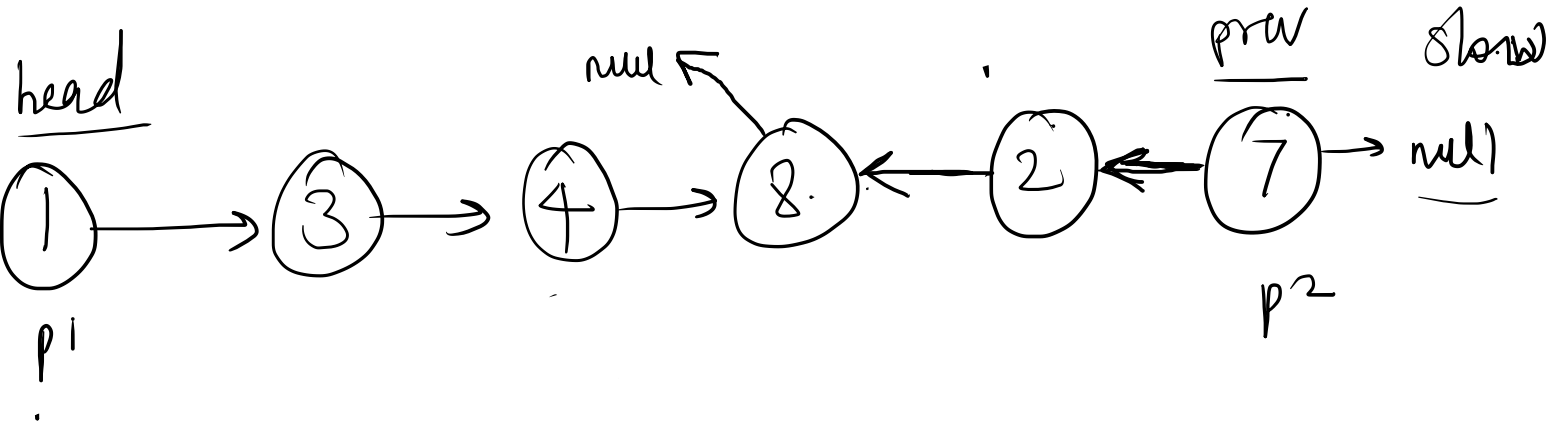
✓ ② Reverse the second half

③ Travers p1 from start
p2 from end
find max pair sum

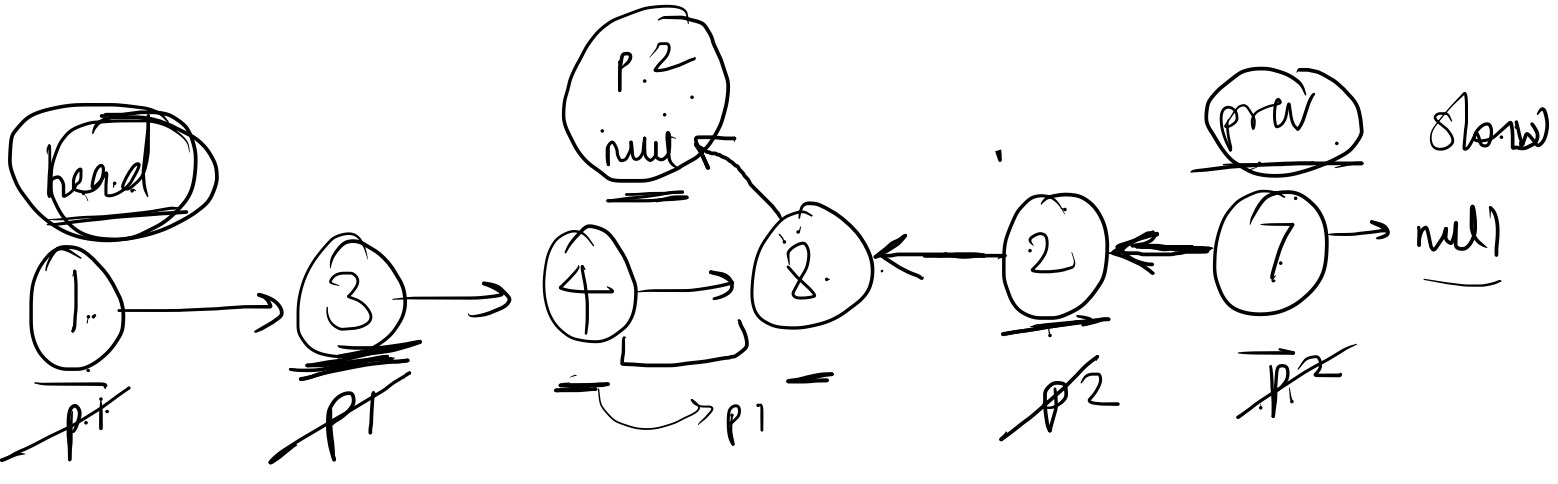
max(val1 + val2)



① find middle



(2) Reverse second half

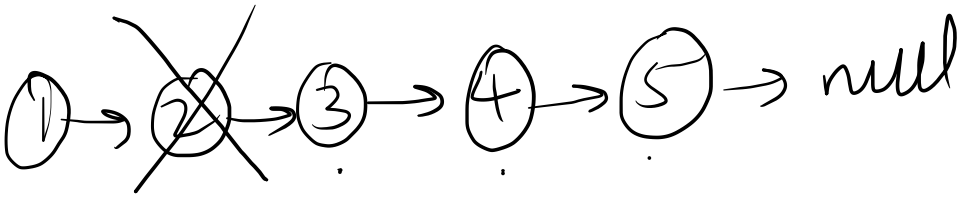


maxsum = 12

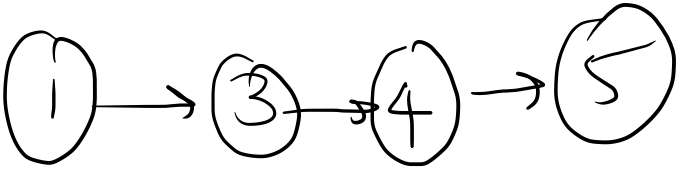
$TC = O(n)$

Aux $SC = O(1)$

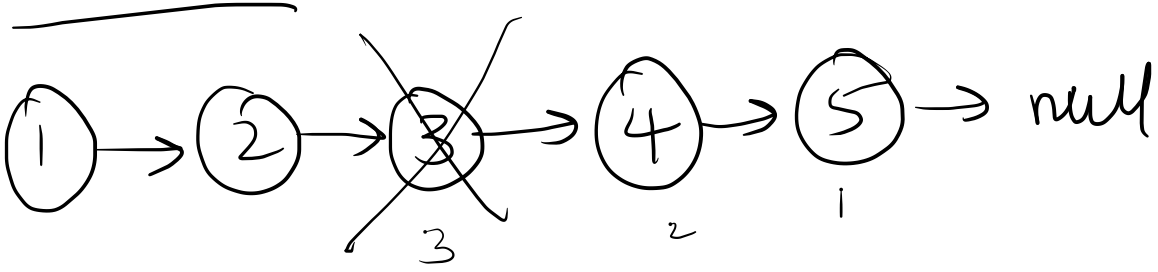
Remove the n^{th} element from end

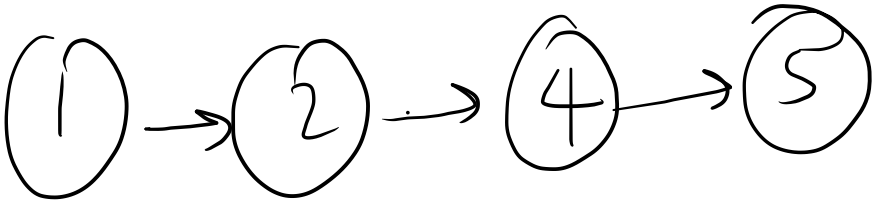


n = 4

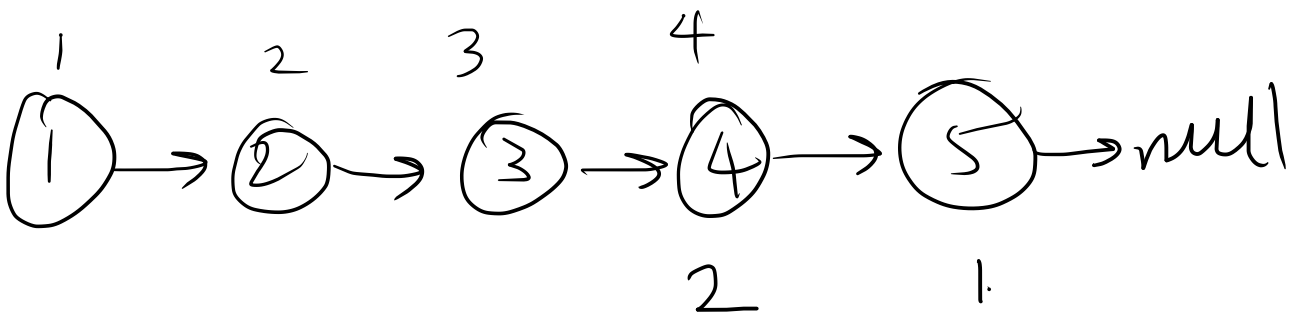


n = 3





n=2



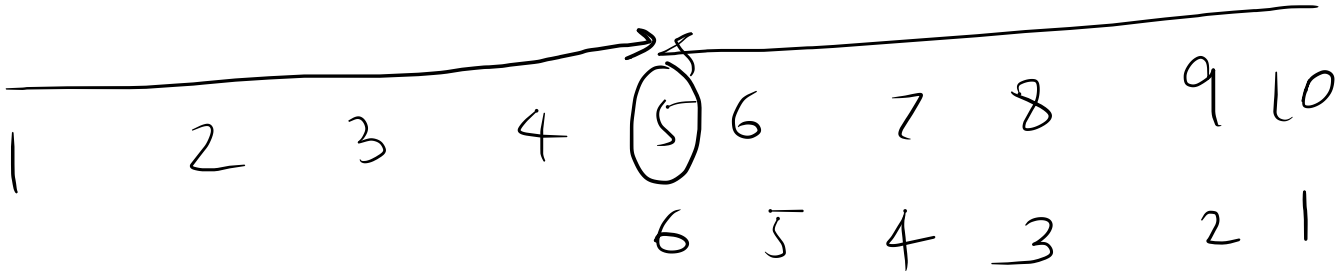
length - n + 1

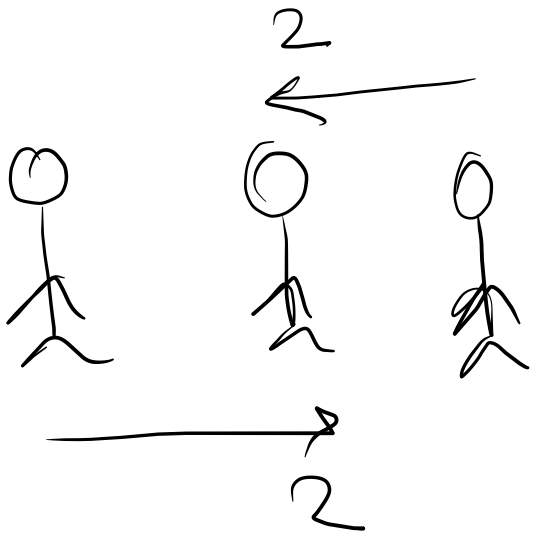
5 - 2 + 1 = 4

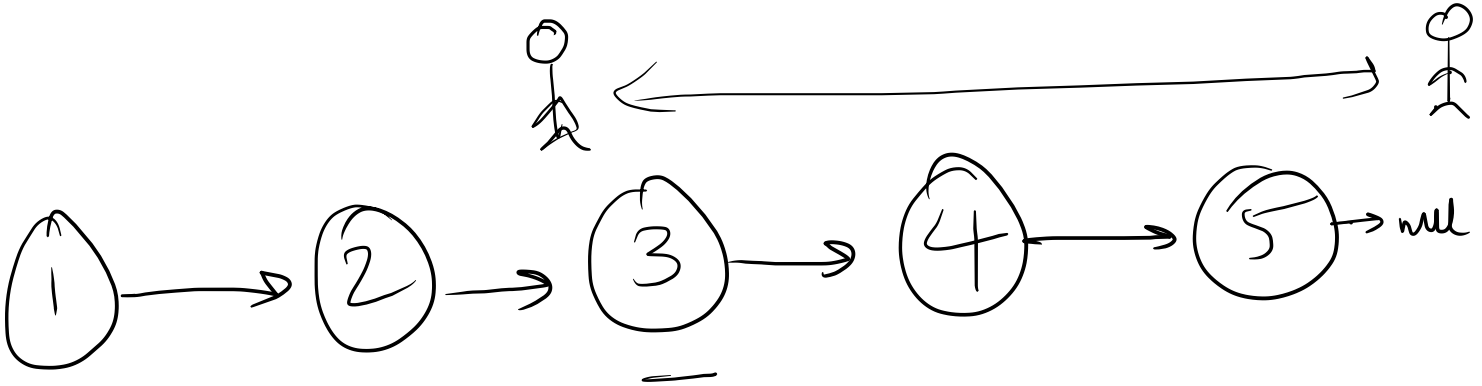
10 students

backward rank = 6

forward rank

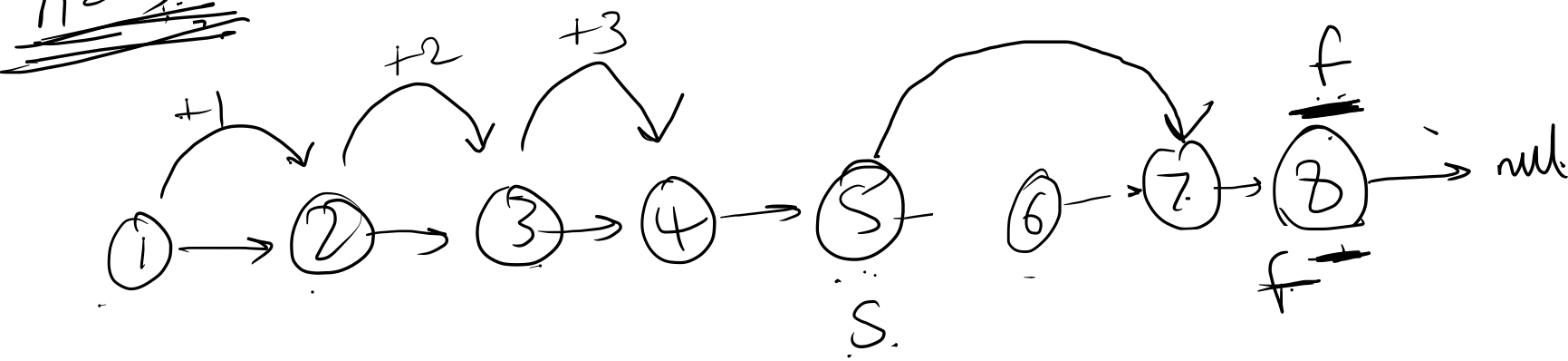



2
2

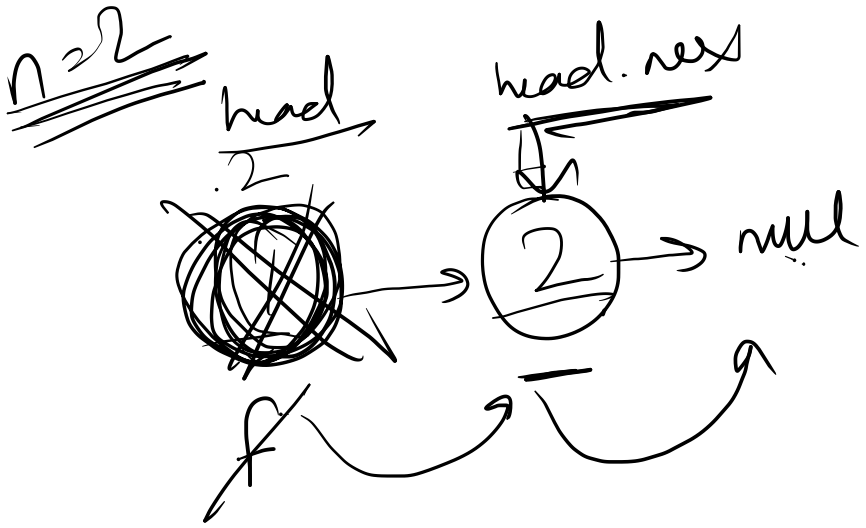


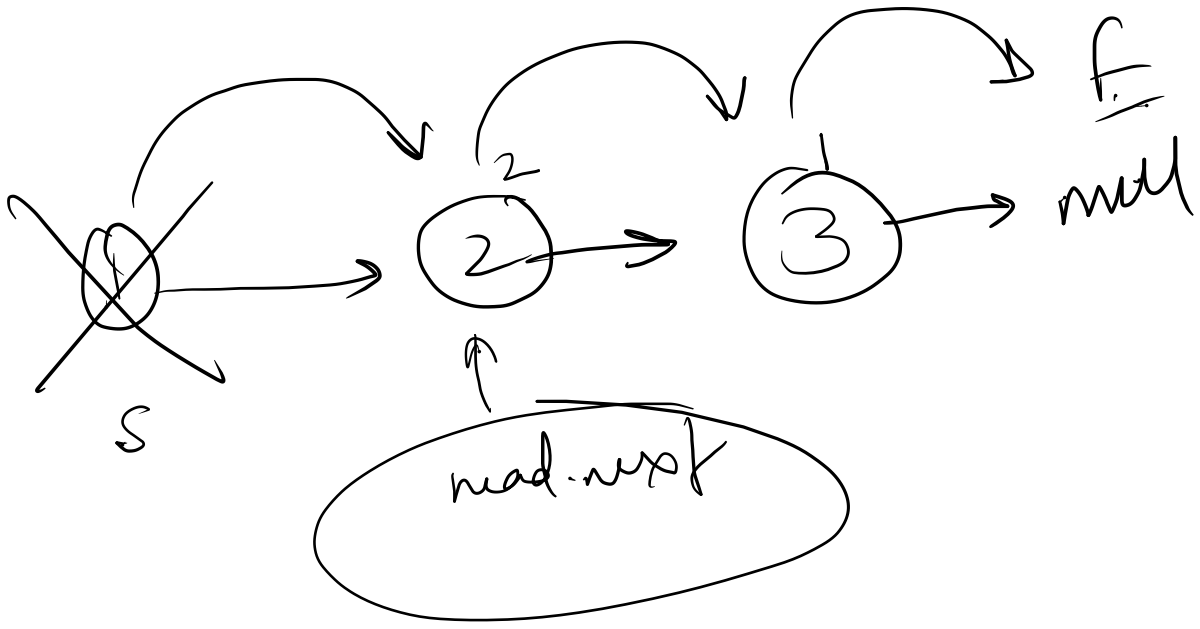
n = 3

n = 3.



slow.next = slow.next.next

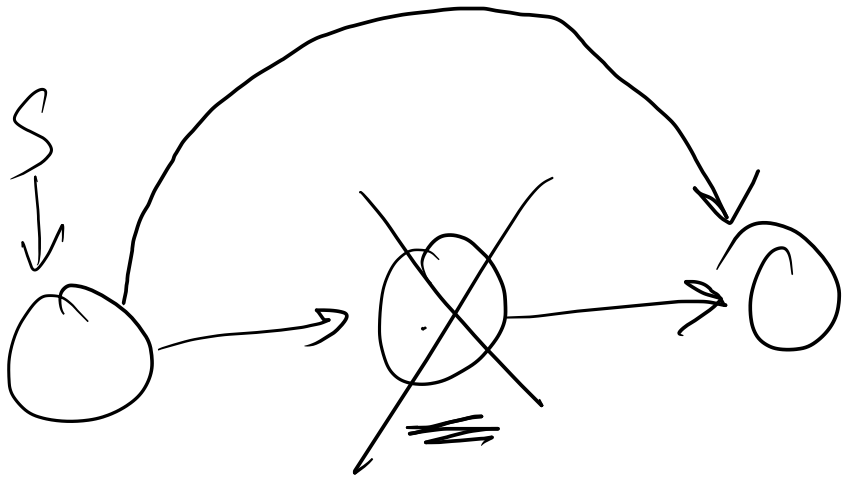

n >= 2
head
head.next
2
2
null
f
g



n=3

TC = O(n)

Aux space = O(1)



slow.next = slow.next.next