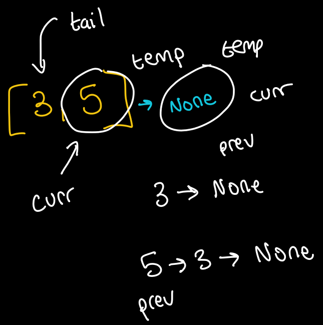
Problem: Given a LL, and #s m and n, return it back with only positions m to n in reverse

1 → 2 → 3 → 4 → 5 → null ⇒ 4 → 3 → 2

m=2 n=4

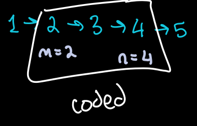


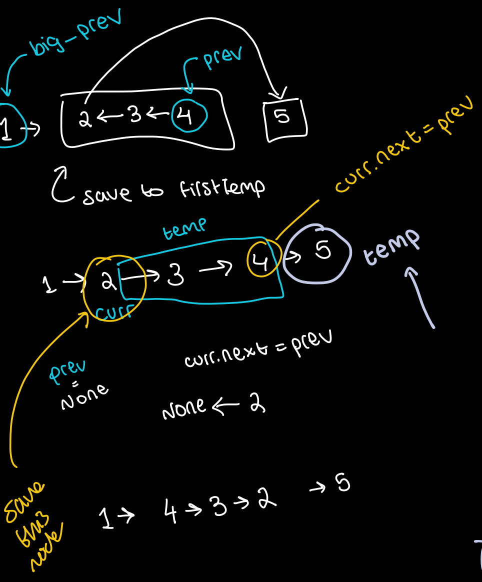
Intro:

- Verify Constraints
    - m and n always within bounds of LL? yes
- Create Testcases
    - bestcase: 1, 2, 3, 4, 5 m=2, n=4 —> 1,4,3,2,5
    - 1, 4, 3, 2, 5 m=2, n=4 —> 1,4,5,2,3
    - 1, m = n = 1
    - null, m = n = 0

Optimal:

- Brainstorming & Pattern Observations





- Pseudocode

```
prev = None
count = 1
  while curr:
    if m ≤ count:
          temp = curr.next
          curr.next = prev
          prev = curr
          curr = temp
    curr = curr.next
    count += 1
```

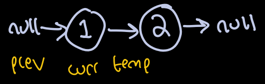
only 2 majo differences

if count == n:
return prev

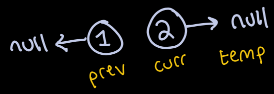
- Write code
- Run through testcases
- Analyze time and space complexity
    - Time: O(n)
    - Space: O(1)

Technique:

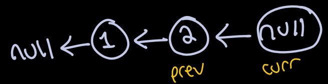
swapping



temp = curr.next
curr.next = prev
prev = curr
curr = temp



repeat



return prev

Edge Case:

If m==left==head

then there won't be a previous node to connect back to!

⇒ check if the prev is null first