**Fast & Slow Pointer Pattern - Complete Sheet**

**✨ What is Fast & Slow Pointer Pattern?**

A powerful technique where two pointers move through the data structure at different speeds to solve problems like cycle detection, finding middle element, duplicate number detection, etc.

**🌟 Fixed Template (Universal Skeleton)**

def fast\_slow\_template(start):

slow = start

fast = start

while fast and fast.next:

slow = slow.next

fast = fast.next.next

# Problem-specific condition check

if slow == fast:

# Optional post-processing

return <depends on problem>

# Optional post-processing

return <depends on problem>

**🔹 LeetCode Problem Patterns**

**Problem 1: Linked List Cycle (LeetCode 141)**

**Statement:** Detect if a linked list has a cycle.

**Example:** Input: head = [3,2,0,-4], pos = 1  
Output: True

**Solution:**

def hasCycle(head):

slow = head

fast = head

while fast and fast.next:

slow = slow.next

fast = fast.next.next

if slow == fast:

return True

return False

**Problem 2: Linked List Cycle II (LeetCode 142)**

**Statement:** Return the node where the cycle begins.

**Example:** Input: head = [3,2,0,-4], pos = 1  
Output: Node with value 2

**Solution:**

def detectCycle(head):

slow = head

fast = head

while fast and fast.next:

slow = slow.next

fast = fast.next.next

if slow == fast:

break

else:

return None

slow = head

while slow != fast:

slow = slow.next

fast = fast.next

return slow

**Problem 3: Middle of the Linked List (LeetCode 876)**

**Statement:** Find the middle node of a linked list.

**Example:** Input: [1,2,3,4,5]  
Output: 3

**Solution:**

def middleNode(head):

slow = head

fast = head

while fast and fast.next:

slow = slow.next

fast = fast.next.next

return slow

**Problem 4: Happy Number (LeetCode 202)**

**Statement:** Determine if a number is happy.

**Example:** Input: n = 19  
Output: True

**Solution:**

def sum\_of\_squares(n):

total = 0

while n > 0:

digit = n % 10

total += digit \* digit

n //= 10

return total

def isHappy(n):

slow = n

fast = n

while True:

slow = sum\_of\_squares(slow)

fast = sum\_of\_squares(sum\_of\_squares(fast))

if slow == fast:

break

return slow == 1

**Problem 5: Remove N-th Node From End (LeetCode 19)**

**Statement:** Remove the N-th node from end.

**Example:** Input: head = [1,2,3,4,5], n = 2  
Output: [1,2,3,5]

**Solution:**

def removeNthFromEnd(head, n):

dummy = ListNode(0)

dummy.next = head

slow = dummy

fast = dummy

for \_ in range(n):

fast = fast.next

while fast.next:

slow = slow.next

fast = fast.next

slow.next = slow.next.next

return dummy.next

**Problem 6: Find Duplicate Number (LeetCode 287)**

**Statement:** Find duplicate number in an array.

**Example:** Input: [1,3,4,2,2]  
Output: 2

**Solution:**

def findDuplicate(nums):

slow = nums[0]

fast = nums[0]

while True:

slow = nums[slow]

fast = nums[nums[fast]]

if slow == fast:

break

slow = nums[0]

while slow != fast:

slow = nums[slow]

fast = nums[fast]

return slow

**🔹 Summary**

**Common Steps:**

1. Initialize slow and fast pointers.
2. Move slow by 1 step, fast by 2 steps.
3. Condition check depends on problem:
   * Detect cycle: slow == fast
   * Find middle: when fast reaches end
   * Find cycle start: reset slow to head and move both by 1
   * Remove N-th node: move fast n steps ahead
   * Find duplicate: cycle detection in array

Master this pattern, and you will easily crack these problems in interviews!