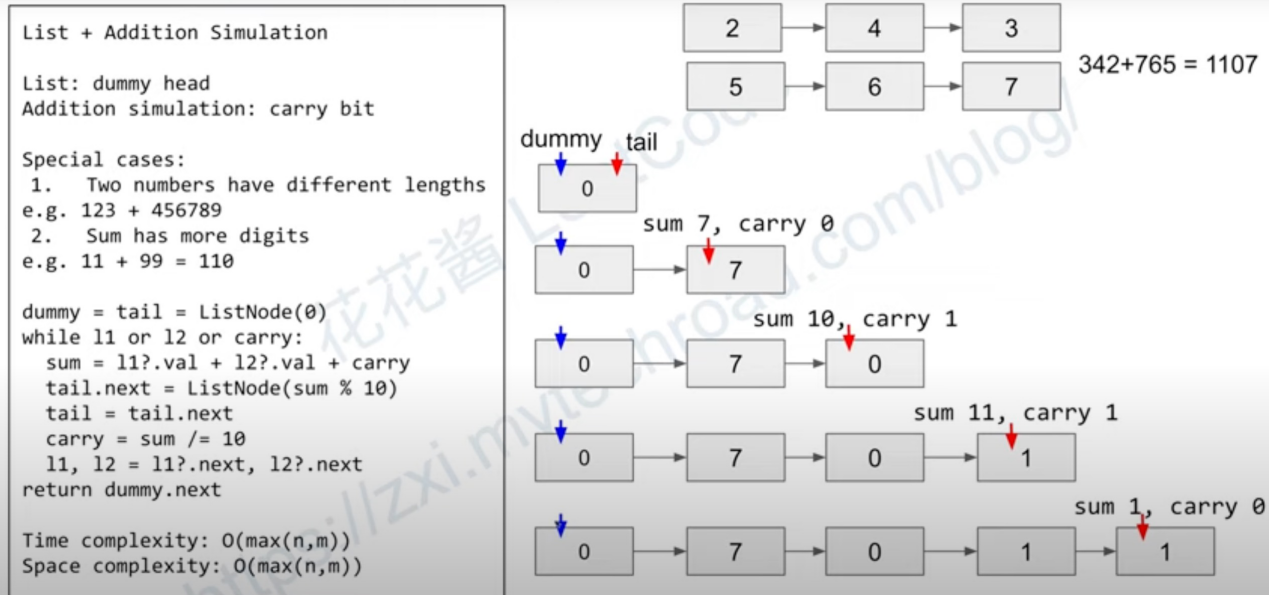
**Linked-List**

1. Add Two Numbers



Creat node:

ListNode dummy = new ListNode();

Two varibles pointing to one node:

ListNode tail = dummy;

Next node of last node is null;

last\_node.next == null;

-> true

Usually use a dummy and a tail, next node of dummy is start of result, tail traverse linked list:

return dummy.next;

In java, ListNode l1 doesn’t return boolean, (l1 != null) returns boolean.

Operator /=:

(a /= b) == (a = a/b)

discard reminder

Operator %:

take reminder

1. Swap Nodes in Pairs

ListNode dummy = head;

dummy and head point to the same object, doesn’t change object

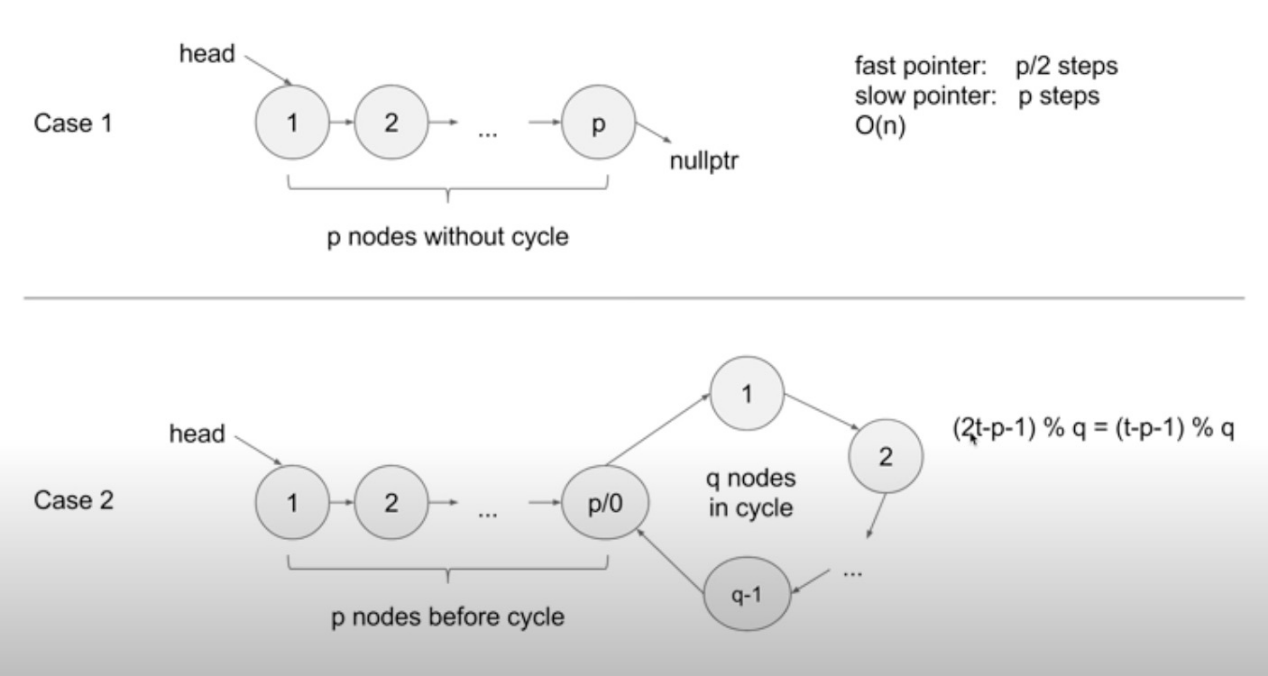
dummy.next = something;

changes object, dummy.next and head.next both changed

1. Reverse Linked List

Use 3 pointers, prev, curr, next

1. Linked List Cycle



Solution 1:

store node in Hash Set and check,

space complexity O(n)

time complexity O(n)

Solution 2:

fast pointer and slow pointer,

if there is a cycle, fast will meet slow eventually

space complexity O(1)

time complexity O(n)

1. Add Two Number II

write a function to reverse linked list

reverse l1, l2

add them

reverse the added linked list

1. Linked List Cycle II

Use a hashset to store nodes