TIPS template

# Interviewer:

## Behavioral:

Ask the interviewee what the number one company for which they want to work is. Ask them why they are interested in that company and what they can offer to that company.

## Question:

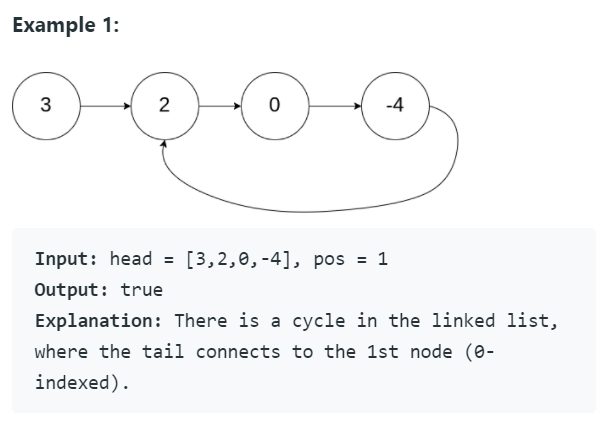
Given head, the head of a linked list, determine if the linked list has a cycle in it.

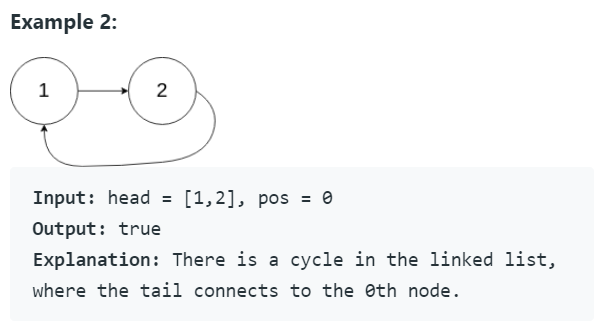
There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter.

Return true if there is a cycle in the linked list. Otherwise, return false.

Source: <https://leetcode.com/problems/linked-list-cycle/>

## Examples:





## Follow up Q&A:

* What if the head is null?
  + Return true, because there is no cycle
* What if the list has no cycle?
  + Return true
* What if the list has multiple cycles?
  + Ask them if that is possible? The answer is no.
* Are the numbers unique?
  + No, the node’s numbers are not unique.
* Are there any time or space complexity restrictions?
  + No, but the optimal solution uses at most O(1) space.

## Hint(s):

*Ask if they would like a hint before giving a hint*

Note: If they are giving a more brute force solution (e.g. hashmap solution) let them finish their thought before giving them a hint.

Hint 1: How could you do this in O(1) space complexity?

Hint 2: What strategies have we learned about in class that can be used to solve linked list problems?

Hint 3: What do you know about slow pointer, fast pointer? How can you apply it to this situation?

## Solution(s): (General concept and time/space complexity)

### Slow Pointer Fast Pointer solution

Have a slow pointer that moves one node for every two nodes the fast pointer moves. If the fast pointer encounters the slow pointer node, we have a cycle. If the fast pointer reaches a nullptr, we do not have a cycle.

Time complexity: O(N) for when there is and isn’t a cycle.

Space complexity: O(1)

### Hashmap solution

Create a hashmap/set and store a reference to each node encountered along the way. If you encounter a node that is already in the map/set, you have a cycle. If you reach a nullptr, you do not have a cycle.

Time complexity: O(N) for when there is and isn’t a cycle.

Space complexity: O(N) to store N nodes in the hashmap

### Other questions follow up

*Ask if there is more than 5 minutes remaining when they finish their code and testing.*

How would you change your solution to return the index of the node in the list where the cycle begins? (the node first visited twice in the cycle)

Source: <https://leetcode.com/problems/linked-list-cycle-ii/>

# 

# 

# 

# 

# 

# 

# 

# Interviewee:

## Question:

## Example(s):

## Code below