

PROBLEM 1

141. Linked List Cycle

Easy Topics Companies

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`.

Example 1:

```
graph LR; 3((3)) --> 2((2)); 2 --> 0((0)); 0 --> -4((-4)); -4 --> 3;
```

Input: `head = [3,2,0,-4]`, `pos = 1`
Output: `true`
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).

Code Accepted

Accepted 29 / 29 testcases passed
oSharik submitted at Jan 20, 2026 13:14

Editorial Solution

Runtime 9 ms | Beats 50.60%

Memory 11.88 MB | Beats 52.60%

Analyze Complexity

Code C++

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     ListNode *next;
6  *     ListNode(int x) : val(x), next(NULL) {}
7  * };
8  */
9 class Solution {
10 public:
11     bool hasCycle(ListNode *head) {
12         if (head == NULL || head->next == NULL)
13             return false;
14
15         ListNode* slow = head;
16         ListNode* fast = head;
17
18         while (fast != NULL && fast->next != NULL) {
19             slow = slow->next;
20             fast = fast->next->next;
21
22             if (slow == fast)
23                 return true;
24         }
25
26         return false;
27     }
28 };
29
```

PROBLEM 2

142. Linked List Cycle II

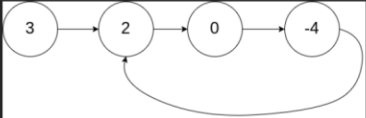
Medium Topics Companies

Given the `head` of a linked list, return the node where the cycle begins. If there is no cycle, return `null`.

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 (0-indexed). It is `-1` if there is no cycle. **Note that** `pos` is not passed as a parameter.

Do not modify the linked list.

Example 1:



Input: `head = [3,2,0,-4], pos = 1`

Accepted 18 / 18 testcases passed

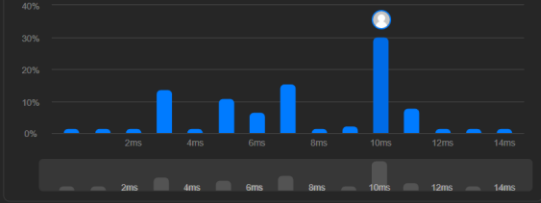
0xSharik submitted at Jan 20, 2026 13:10

Editorial Solution

Runtime: 10 ms / Beats 47.28%

Memory: 11.45 MB / Beats 23.63%

Analyze Complexity



Code | C++

All Submissions

```
9 class Solution {
10 public:
11     ListNode *detectCycle(ListNode *head) {
12         if(head == NULL || head->next == NULL) return NULL;
13
14         ListNode *slow = head;
15         ListNode *fast = head;
16
17
18         while(fast && fast->next){
19             slow = slow->next;
20             fast = fast->next->next;
21
22             if(slow == fast) break;
23         }
24
25
26         if(fast == NULL || fast->next == NULL) return NULL;
27
28
29         slow = head;
30         while(slow != fast){
31             slow = slow->next;
32             fast = fast->next;
33         }
34
35         return slow;
36     }
37 };
```

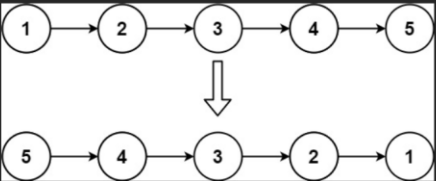
PROBLEM 3

206. Reverse Linked List Solved

Easy Topics Companies

Given the `head` of a singly linked list, reverse the list, and return the *reversed list*.

Example 1:



Input: `head = [1,2,3,4,5]`
Output: `[5,4,3,2,1]`

Example 2:

Runtime: 0 ms | Beats 100.00%
Memory: 13.39 MB | Beats 70.83%

Accepted 28 / 28 testcases passed
OxShark submitted at Jan 20, 2026 13:11

Code | C++

Code | Accepted

All Submissions

```
1  /**
2   * Definition for singly-linked list.
3   * struct ListNode {
4   *     int val;
5   *     ListNode *next;
6   *     ListNode() : val(0), next(nullptr) {}
7   *     ListNode(int x) : val(x), next(nullptr) {}
8   *     ListNode(int x, ListNode *next) : val(x), next(next) {}
9   * };
10 */
11 class Solution {
12 public:
13     ListNode* reverseList(ListNode* head) {
14         ListNode* prev = NULL;
15         ListNode* curr = head;
16
17         while (curr != NULL) {
18             ListNode* nextNode = curr->next;
19             curr->next = prev;
20             prev = curr;
21             curr = nextNode;
22         }
23
24         return prev;
25     }
26 };
27
```

PROBLEM 4

Problem List

Description

Editorial

Solutions

Submissions


876. Middle of the Linked List

Easy Topics Companies

Given the **head** of a singly linked list, return the **middle node** of the linked list.


If there are two middle nodes, return **the second middle node**.

Example 1:



Input: head = [1,2,3,4,5]
Output: [3,4,5]
Explanation: The middle node of the list is node 3.

Example 2:



Code

Accepted

All Submissions

Accepted 36 / 36 testcases passed

OcSharik submitted at Jan 20, 2026 13:12

Editorial

Solution

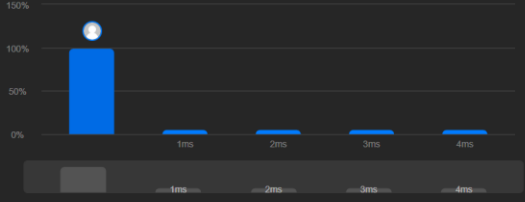
Runtime

0 ms | Beats 100.00%

Memory

9.98 MB | Beats 62.09%

Analyze Complexity



Code | C++

Code | C++

```
1  /**
2   * Definition for singly-linked list.
3   * struct ListNode {
4   *     int val;
5   *     ListNode *next;
6   *     ListNode() : val(0), next(nullptr) {}
7   *     ListNode(int x) : val(x), next(nullptr) {}
8   *     ListNode(int x, ListNode *next) : val(x), next(next) {}
9   * };
10 */
11 class Solution {
12 public:
13     ListNode* middleNode(ListNode* head) {
14         ListNode* slow = head;
15         ListNode* fast = head;
16
17         while (fast != NULL && fast->next != NULL) {
18             slow = slow->next;
19             fast = fast->next->next;
20         }
21
22         return slow;
23     }
24 };
25
```