

## PROBLEM 141

DescriptionEditorialSolutionsSubmissions

### 141. Linked List Cycle

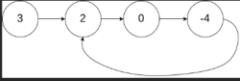
EasyTopicsCompanies

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:**



**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).

**Example 2:**



17.3K484215 Online

CodeAccepted

All Submissions

Accepted29 / 29 testcases passed

Harshit4456 submitted at Jan 20, 2026 13:15

EditorialSolution

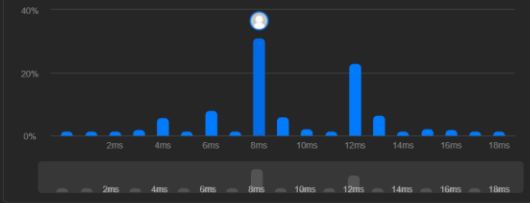
Runtime

8 msBeats: 81.73%

Analyze Complexity

Memory

11.92 MBBeats: 22.38%



TestcaseTest Result

You must run your code first

CodeC++

```
1 class Solution {
2 public:
3     bool hasCycle(ListNode *head) {
4         if (head == NULL) return false;
5
6         ListNode *slow = head;
7         ListNode *fast = head;
8
9         while (fast != NULL && fast->next != NULL) {
10             slow = slow->next;
11             fast = fast->next->next;
12
13             if (slow == fast)
14                 return true;
15         }
16         return false;
17     }
18 };
19
```

View less

## PROBLEM 142

Problem List

Description

Editorial

Solutions

Submissions

### 142. Linked List Cycle II

Medium Topics Companies

Solved

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:**

```

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

**Input:** `head = [3,2,0,-4]`, `pos = 1`  
**Output:** tail connects to node index 1  
**Explanation:** There is a cycle in the linked list, where tail connects to the second node.

15K 250 98 Online

Code Accepted

All Submissions

Accepted 18 / 18 testcases passed  
 Harshit4456 submitted at Jan 20, 2026 13:17

Runtime

3 ms Beats 98.27%

Analyze Complexity

Memory

11.44 MB Beats 23.63%

Testcase

Test Result

You must run your code first

```

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     ListNode* detectCycle(ListNode* head) {
12         ListNode *slow = head, *fast = head;
13
14         while (fast && fast->next) {
15             slow = slow->next;
16             fast = fast->next->next;
17
18             if (slow == fast) {
19                 while (head != slow) {
20                     head = head->next;
21                     slow = slow->next;
22                 }
23                 return slow;
24             }
25         }
26         return NULL;
27     }
28 };
29

```

## PROBLEM 206

Problem List

206. Reverse Linked List

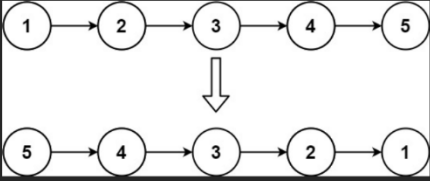
Easy

Topics

Companies

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

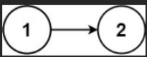
Example 1:



```
graph LR; 1((1)) --> 2((2)); 2 --> 3((3)); 3 --> 4((4)); 4 --> 5((5)); 5 --> null; 5 --> 4; 4 --> 3; 3 --> 2; 2 --> 1; 1 --> null;
```

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

Example 2:



```
graph LR; 1((1)) --> 2((2)); 2 --> null;
```

24.1K 384 302 Online

Accepted

All Submissions

Accepted 28 / 28 testcases passed

Harshit4456 submitted at Jan 20, 2026 13:19

Editorial

Solution

Runtime

0 ms Beats 100.00%

Analyze Complexity

Memory

13.47 MB Beats 40.69%

150%

100%

50%

0%

1ms

2ms

3ms

4ms

Testcase

Test Result

You must run your code first

Code

C++

Auto

```
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, *cur = head;
15
16         while (cur) {
17             ListNode* nxt = cur->next;
18             cur->next = prev;
19             prev = cur;
20             cur = nxt;
21         }
22         return prev;
23     }
24 };
```

Saved

Testcase

Test Result

## PROBLEM 876

Problem List

876. Middle of the Linked List

Solved

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:**



**Input:** `head = [1,2,3,4,5,6]`  
**Output:** `[4,5,6]`  
**Explanation:** Since the list has two middle nodes with values 3 and 4, we return the

13.1K

250

114 Online

Accepted

36 / 36 testcases passed

Harshit4456 submitted at Jan 19, 2026 15:59

Editorial

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

10.02 MB | Beats 25.07%

100%

50%

0%

1ms

2ms

3ms

4ms

Testcase

Test Result

You must run your code first

### Code

C++

```
1 class Solution {
2 public:
3     ListNode* middleNode(ListNode* head) {
4         ListNode* slow = head;
5         ListNode* fast = head;
6
7         while (fast != nullptr && fast->next != nullptr) {
8             slow = slow->next;
9             fast = fast->next->next;
10        }
11
12        return slow;
13    }
14 };
15
```

Saved