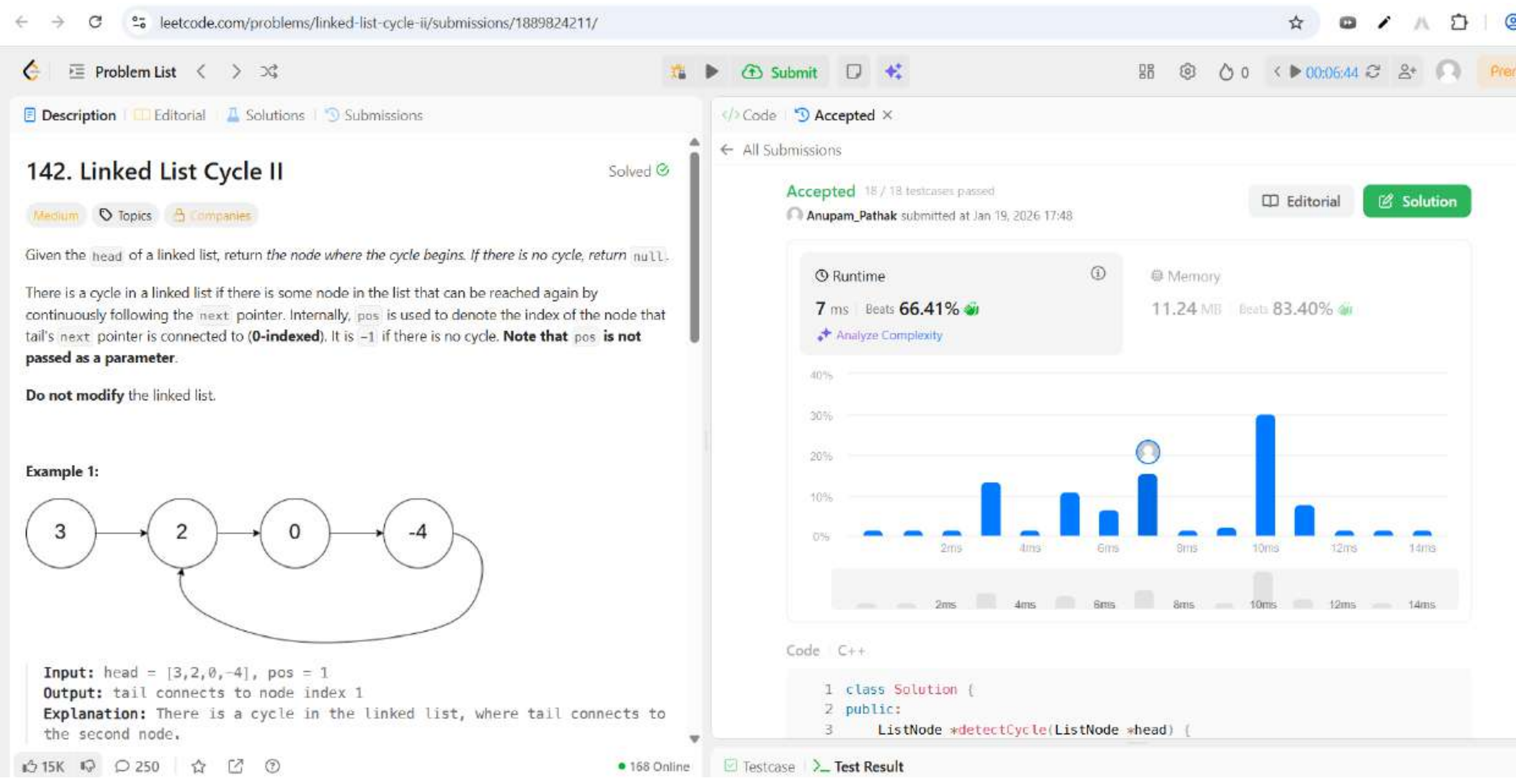


142. Linked List Cycle II

Solved

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
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 | 168 Online

Accepted 18 / 18 testcases passed

Anupam_Pathak submitted at Jan 19, 2026 17:48

Editorial | Solution

Runtime: 7 ms | Beats 66.41%

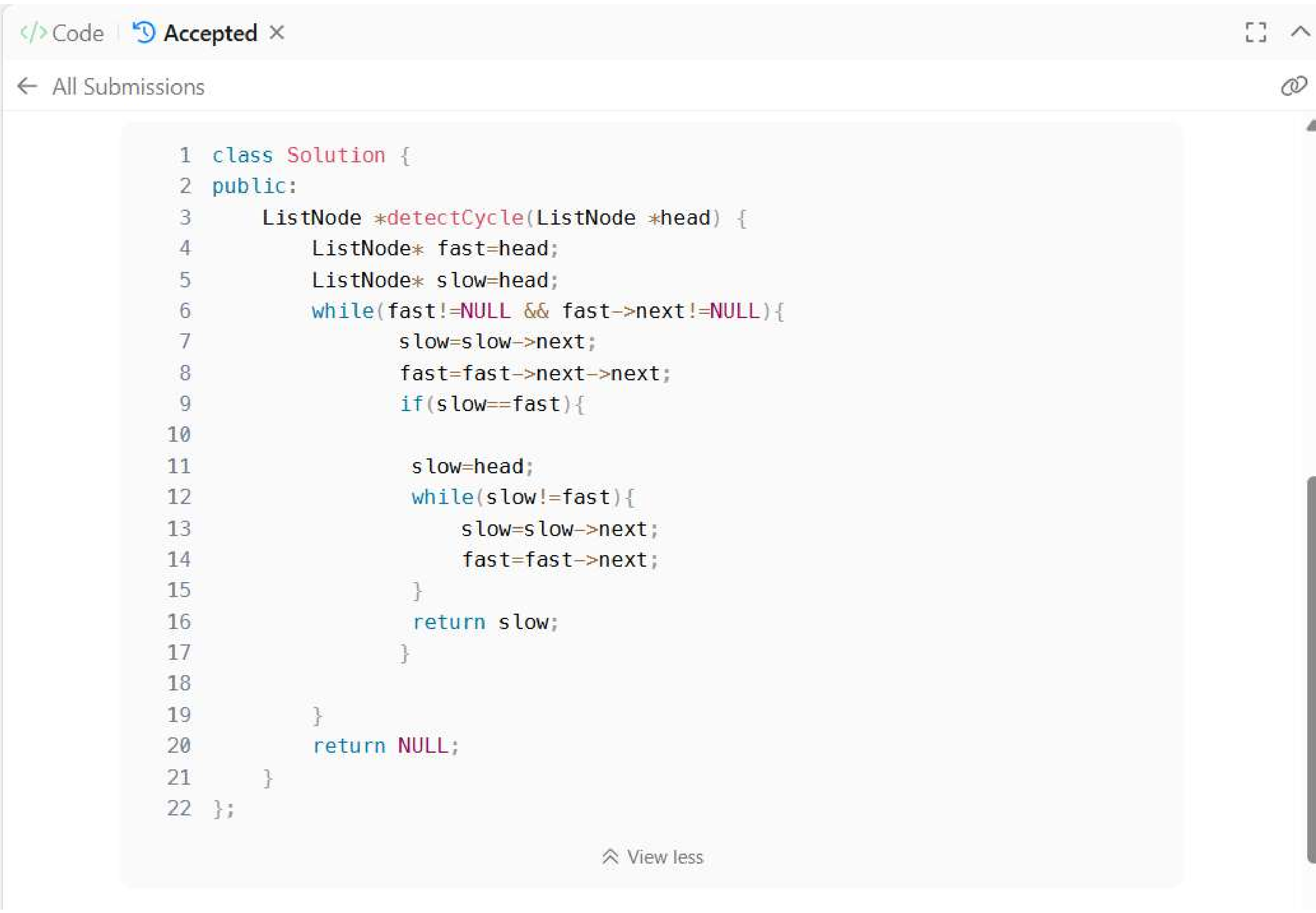
Analyze Complexity

Memory: 11.24 MB | Beats 83.40%

Code | C++

```
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
```

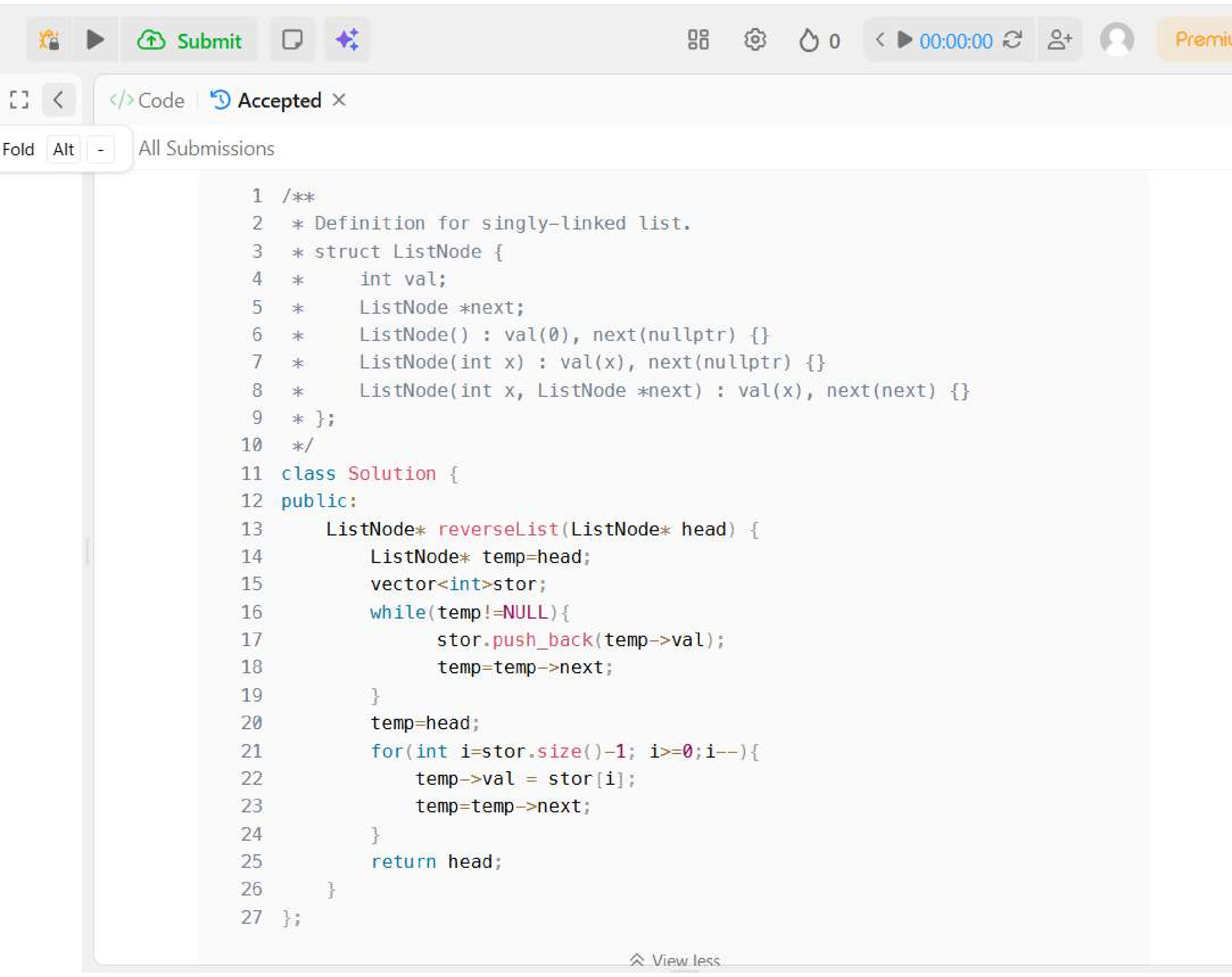
Testcase | Test Result



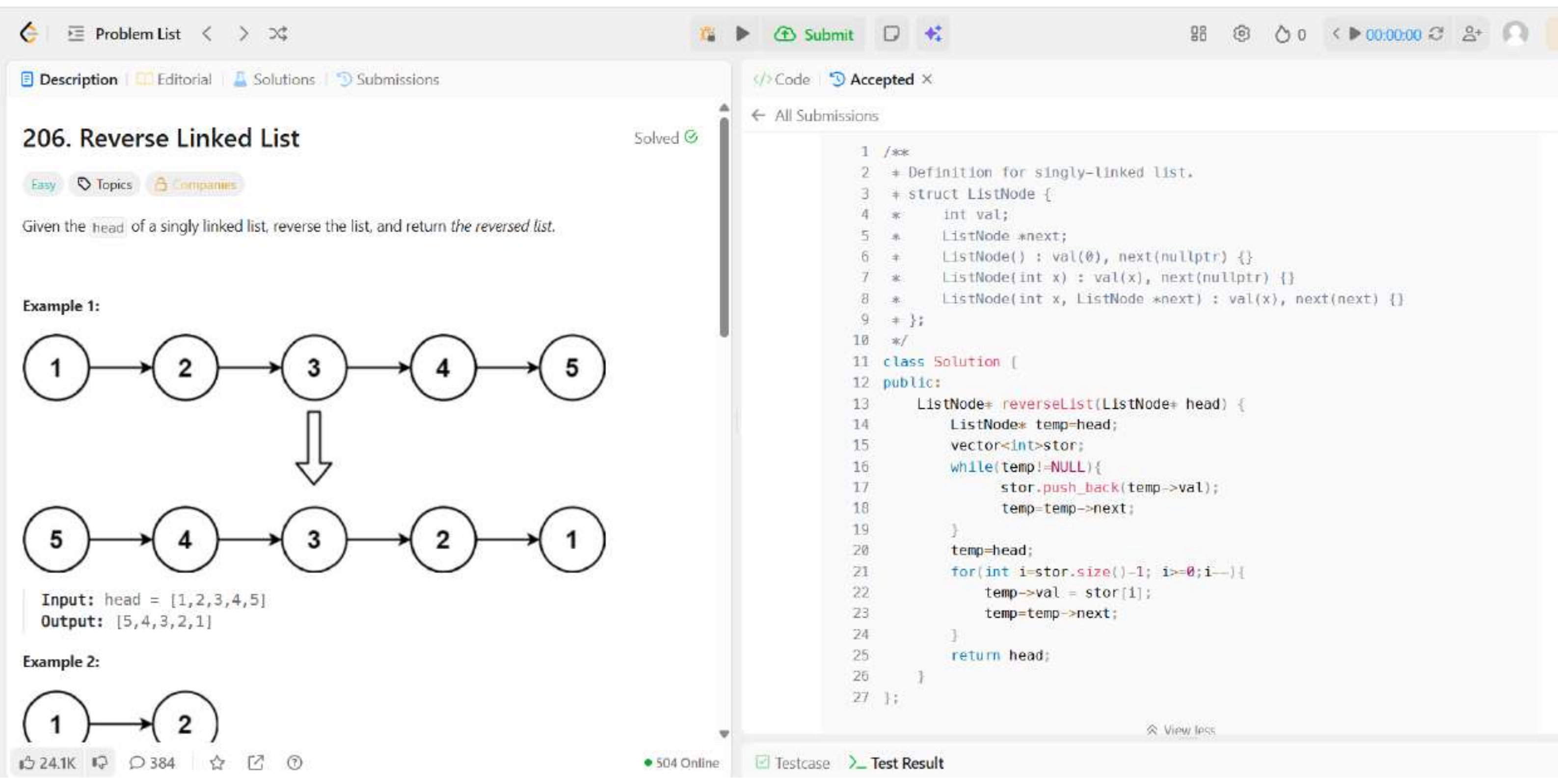
```cpp
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        ListNode* fast=head;
        ListNode* slow=head;
        while(fast!=NULL && fast->next!=NULL){
                slow=slow->next;
                fast=fast->next->next;
                if(slow==fast){

                 slow=head;
                 while(slow!=fast){
                     slow=slow->next;
                     fast=fast->next;
                 }
                 return slow;
                }

        }
        return NULL;
    }
};
```



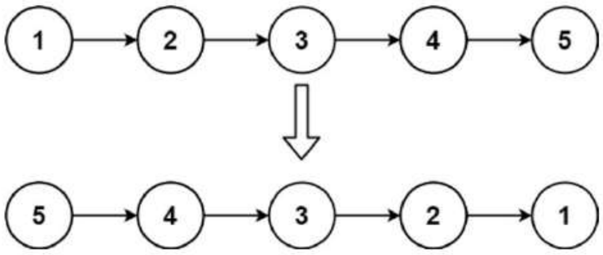
```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* temp=head;
        vector<int>stor;
        while(temp!=NULL){
              stor.push_back(temp->val);
              temp=temp->next;
        }
        temp=head;
        for(int i=stor.size()-1; i>=0;i--){
            temp->val = stor[i];
            temp=temp->next;
        }
        return head;
    }
};
```



206. Reverse Linked List

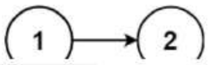
Given the `head` of a singly linked list, reverse the list, and return *the reversed list.*

Example 1:



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

Example 2:



```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* temp=head;
        vector<int>stor;
        while(temp!=NULL){
              stor.push_back(temp->val);
              temp=temp->next;
        }
        temp=head;
        for(int i=stor.size()-1; i>=0;i--){
             temp->val = stor[i];
             temp=temp->next;
        }
        return head;
    }
};
```