

Linked list cycle

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode* fast = head;
        ListNode* slow = head;

        while (fast != nullptr && fast->next != nullptr) {
            fast = fast->next->next;
            slow = slow->next;

            if (fast == slow) {
                return true;
            }
        }

        return false;
    }
};
```

Description | Accepted | Editorial | Solutions | Submissions


← All Submissions

Accepted 29 / 29 testcases passed
12ananya submitted at Feb 13, 2025 22:05

Editorial Solution

Runtime
6 ms | Beats 90.77%
[Analyze Complexity](#)

Memory
11.84 MB | Beats 54.01%



</> Code

C++ v Auto

```
8  */
9  class Solution {
10 public:
11     bool hasCycle(ListNode *head) {
12         ListNode* fast = head;
13         ListNode* slow = head;
14
15         while (fast != nullptr && fast->next != nullptr) {
16             fast = fast->next->next;
17             slow = slow->next;
18
19             if (fast == slow) {
20                 return true;
21             }
22         }
23     }
24 }
```

Saved

Ln

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3