



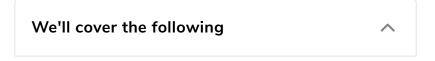






Solution Review: Detect Loop in a Linked List

This review provides a detailed analysis of the different ways to solve the Detect a Loop in a Linked List challenge.



- Solution: Floyd's Cycle-Finding Algorithm
 - Time Complexity

Solution: Floyd's Cycle-Finding Algorithm

```
main.py
LinkedList.py
Node.py
    from LinkedList import LinkedList
    # Floyd's Cycle Finding Algorithm
  3
    def detect_loop(lst):
         # Keep two iterators
  4
  5
         onestep = lst.get head()
         twostep = lst.get_head()
  6
  7
         while onestep and twostep.next_element:
  8
             onestep = onestep.next_element # Moves one node at a time
  9
             twostep = twostep.next_element.next_element # Skips a node
 10
             if onestep == twostep: # Loop exists
 11
                 return True
```

```
12
        return False
13
15
16
17
    lst = LinkedList()
18
19
    lst.insert_at_head(21)
    lst.insert_at_head(14)
20
    lst.insert_at_head(7)
21
22
23
   # Adding a loop
24
    head = lst.get_head()
    node = lst.get_head()
25
26
27
    for i in range(4):
28
        if node.next_element is None:
```

This is perhaps the fastest algorithm for detecting a linked list loop. We keep track of two iterators, onestep and twostep.

onestep moves forward one node at a time, while twostep iterates over two nodes. In this way, twostep is the faster iterator.

By principle, if a loop exists, the two iterators will meet. Whenever this condition is fulfilled, the function returns True.

Time Complexity#

We iterate the list once, which makes the total running time of this solution O(n).

Note: The solution above has another approach that uses sets. We will cover that approach in Hashing Chapter: Challenge 10



In the next lesson, we'll figure out a way to find the middle next lesson.

Interviewing soon? We've partnered with Hired so that $$\times$$ companies apply to you instead of you applying to them. See how \odot



Challenge 6: Detect Loop in a Linked L...

Challenge 7: Find Middle Node of Link...



! Report an Issue

