

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.

We'll cover the following



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

Solution: Floyd's Cycle-Finding Algorithm

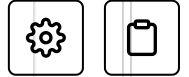
main.py

LinkedList.py

Node.py

```
1 from LinkedList import LinkedList
2 # Floyd's Cycle Finding Algorithm
3 def detect_loop(lst):
4     # Keep two iterators
5     onestep = lst.get_head()
6     twostep = lst.get_head()
7     while onestep and twostep 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
14 # -----
```

```
15
16
17 lst = LinkedList()
18
19 lst.insert_at_head(21)
20 lst.insert_at_head(14)
21 lst.insert_at_head(7)
22
23 # Adding a loop
24 head = lst.get_head()
25 node = lst.get_head()
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. On average, lookup in a list takes $O(1)$ time, which makes the total running time of this solution $O(n)$. However, if we use sets in place of lists to store visited nodes, then a single look-up may take $O(n)$ time. This can cause the algorithm to take $O(n^2)$ time.



Note: The solution above has another approach that uses sets. We will cover that approach in Hashing Chapter: Challenge 10 (<https://www.educative.io/courses/data-structures-in-python-an-interview-refresher/3w7qnv17rEQ>)

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

← Back

Next →

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

Challenge 7: Find Middle Node of Link...

☒ Mark as Completed



Report an
Issue



Ask a Question

(https://discuss.educative.io/tag/solution-review-detect-loop-in-a-linked-list__introduction-to-linked-lists__data-structures-for-coding-interviews-in-python)