Solution Review: Is Circular Linked List

This lesson contains the solution review for the challenge of determining whether a given linked list is circular or not.

WE'LL COVER THE FOLLOWING

In this lesson, we investigate how to determine whether a given linked list is either a singly linked list or a circular linked list.

Implementation

Have a look at the coding solution below:

```
def is_circular_linked_list(self, input_list):
  cur = input_list.head
  while cur.next:
    cur = cur.next
    if cur.next == input_list.head:
      return True
  return False
```

Explanation

Let's discuss the class method `is_circular_linked_list`. On **line 2**, we initialize `cur` to `input_list.head` so that we are able to traverse `input_list` that has been passed to `is_circular_linked_list`. On **line 3** we set up a `while` loop which runs until `cur.next` becomes `None`. If `cur.next` becomes `None`, it implies that `input_list` must not be a circular linked list and therefore, if the `while` loop terminates, we return `False` on **line 7**. On the other hand, we update `cur` to `cur.next` in the `while` loop on **line 4**, and if we reach a node while traversing such that the next node is the head node, then this confirms that `input_list` is indeed a `circular_linked_list`. Hence, if the condition on

line 5 turns out to be `True`, we return `True` from the method on **line 6**.

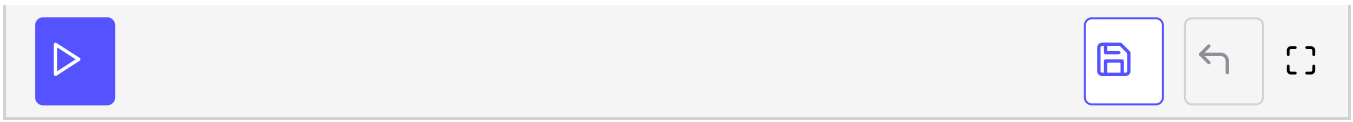
As you can see, the `is_circular_linked_list` method was pretty straightforward. Below we test it on a singly linked list and on a circular linked list:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None


class LinkedList:
    def __init__(self):
        self.head = None

    def print_list(self):
        cur_node = self.head
        while cur_node:
            print(cur_node.data)
            cur_node = cur_node.next

    def append(self, data):
        new_node = Node(data)

        if self.head is None:
            self.head = new_node
            return

        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node


class CircularLinkedList:
    def __init__(self):
        self.head = None

    def prepend(self, data):
        new_node = Node(data)
        cur = self.head
        new_node.next = self.head

        if not self.head:
            new_node.next = new_node
        else:
            while cur.next != self.head:
                cur = cur.next
            cur.next = new_node
        self.head = new_node

    def append(self, data):
        if not self.head:
            self.head = Node(data)
            self.head.next = self.head
        else:
            new_node = Node(data)
```

```
            new_node = Node(data)
            cur = self.head
            while cur.next != self.head:
                cur = cur.next
            cur.next = new_node
            new_node.next = self.head

    def print_list(self):
        cur = self.head

        while cur:
            print(cur.data)
            cur = cur.next
            if cur == self.head:
                break

    def __len__(self):
        cur = self.head
        count = 0
        while cur:
            count += 1
            cur = cur.next
            if cur == self.head:
                break
        return count

    def split_list(self):
        size = len(self)

        if size == 0:
            return None
        if size == 1:
            return self.head

        mid = size//2
        count = 0

        prev = None
        cur = self.head

        while cur and count < mid:
            count += 1
            prev = cur
            cur = cur.next
        prev.next = self.head

        split_cllist = CircularLinkedList()
        while cur.next != self.head:
            split_cllist.append(cur.data)
            cur = cur.next
        split_cllist.append(cur.data)

        self.print_list()
        print("\n")
        split_cllist.print_list()

    def remove(self, key):
        if self.head:
            if self.head.data == key:
                cur = self.head
                while cur.next != self.head:
                    cur = cur.next
                if self.head == self.head.next:
```

```
                    self.head = None
                else:
                    cur.next = self.head.next
                    self.head = self.head.next
            else:
                cur = self.head
                prev = None
                while cur.next != self.head:
                    prev = cur
                    cur = cur.next
                    if cur.data == key:
                        prev.next = cur.next
                        cur = cur.next


    def remove_node(self, node):
        if self.head:
            if self.head == node:
                cur = self.head
                while cur.next != self.head:
                    cur = cur.next
                if self.head == self.head.next:
                    self.head = None
                else:
                    cur.next = self.head.next
                    self.head = self.head.next
            else:
                cur = self.head
                prev = None
                while cur.next != self.head:
                    prev = cur
                    cur = cur.next
                    if cur == node:
                        prev.next = cur.next
                        cur = cur.next

    def josephus_circle(self, step):
        cur = self.head

        while len(self) > 1:
            count = 1
            while count != step:
                cur = cur.next
                count += 1
            print("REMOVED: " + str(cur.data))
            self.remove_node(cur)
            cur = cur.next

    def is_circular_linked_list(self, input_list):
        cur = input_list.head
        while cur.next:
            cur = cur.next
            if cur.next == input_list.head:
                return True
        return False

cllist = CircularLinkedList()
cllist.append(1)
cllist.append(2)
cllist.append(3)
cllist.append(4)
```

```
llist = LinkedList()
llist.append(1)
llist.append(2)
llist.append(3)
llist.append(4)

print(cllist.is_circular_linked_list(cllist))
print(cllist.is_circular_linked_list(llist))
```



That was all we had about circular linked lists in this course. In the next chapter, we'll explore another type of linked list: Doubly Linked Lists.

Hope you are enjoying the lessons and challenges!