Swap-Nodes

May 3, 2020

0.1 Problem Statement

Given a linked list, swap the two nodes present at position `i` and `j`, assuming `0 <= i <= j`. The positions are based on 0-based indexing.

Note: You have to swap the nodes and not just the values.

Example: * `linked_list = 3 4 5 2 6 1 9` * `positions = 2 5` * `output = 3 4 1 2 6 5 9`

Explanation: * The node at position 3 has the value `2` * The node at position 4 has the value `6` * Swapping these nodes will result in a final order of nodes of `3 4 5 6 2 1 9`

0.1.1 Let's take an example to understand a simple approach -

Given linked list = [3, 4, 5, 2, 6, 1, 9] position_one = 2 position_two = 5 **Note the original order of indexes - 0, 1, 2, 3, 4, 5, 6**

Step 1 - Identify the two nodes to be swapped. Also, identify the previous of both the two nodes.

Step 2 - Swap the references making use of a temporary reference **Check the order of the updated indexes as - 0, 1, 5, 3, 4, 2, 6**, which implies that index 2 and index 5 have been swapped.

0.1.2 Helper Class

```
In [1]: class Node:
            """LinkedListNode class to be used for this problem"""
            def __init__(self, data):
                self.data = data
                self.next = None
```

0.1.3 Exercise - Write the function definition here

```
In [1]: """
        :param: head- head of input linked list
        :param: `position_one` - indicates position (index) ONE
        :param: `position_two` - indicates position (index) TWO
        return: head of updated linked list with nodes swapped

        TODO: complete this function and swap nodes present at position_one and position_two
        Do not create a new linked list
        """
```

```
        def swap_nodes(head, left_index, right_index):
            pass
```

Hide Solution

```
In [ ]: # Solution
        """
        :param: head- head of input linked list
        :param: `position_one` - indicates position (index) ONE
        :param: `position_two` - indicates position (index) TWO
        return: head of updated linked list with nodes swapped
        """
        def swap_nodes(head, position_one, position_two):

            # If both the indices are same
            if position_one == position_two:
                return head

            # Helper references
            one_previous = None
            one_current = None

            two_previous = None
            two_current = None

            current_index = 0
            current_node = head
            new_head = None

            # LOOP - find out previous and current node at both the positions (indices)
            while current_node is not None:

                # Position_one cannot be equal to position_two,
                # so either one of them might be equal to the current_index
                if current_index == position_one:
                    one_current = current_node

                elif current_index == position_two:
                    two_current = current_node
                    break

                # If neither of the position_one or position_two are equal to the current_index
                if one_current is None:
                    one_previous = current_node

                two_previous = current_node

                # increment both the current_index and current_node
```

```
            current_node = current_node.next
            current_index += 1


        # Loop ends


        '''SWAPPING LOGIC'''
        # We have identified the two nodes: one_current & two_current to be swapped,
        # Make use of a temporary reference to swap the references
        two_previous.next = one_current
        temp = one_current.next
        one_current.next = two_current.next
        two_current.next = temp

        # if the node at first index is head of the original linked list
        if one_previous is None:
            new_head = two_current
        else:
            one_previous.next = two_current
            new_head = head
        # Swapping logic ends

        return new_head
```

0.1.4 Test - Let's test your function

```
In [3]: def test_function(test_case):
            head = test_case[0]
            left_index = test_case[1]
            right_index = test_case[2]

            left_node = None
            right_node = None

            temp = head
            index = 0
            try:
                while temp is not None:
                    if index == left_index:
                        left_node = temp
                    if index == right_index:
                        right_node = temp
                        break
                    index += 1
                    temp = temp.next

                updated_head = swap_nodes(head, left_index, right_index)
```

```
            temp = updated_head
            index = 0
            pass_status = [False, False]

            while temp is not None:
                if index == left_index:
                    pass_status[0] = temp is right_node
                if index == right_index:
                    pass_status[1] = temp is left_node

                index += 1
                temp = temp.next

            if pass_status[0] and pass_status[1]:
                print("Pass")
            else:
                print("Fail")
            return updated_head
        except Exception as e:
            print("Fail")
```

In [7]:
```
# helper functions for testing purpose
def create_linked_list(arr):
    if len(arr)==0:
        return None
    head = Node(arr[0])
    tail = head
    for data in arr[1:]:
        tail.next = Node(data)
        tail = tail.next
    return head

def print_linked_list(head):
    while head:
        print(head.data, end=" ")
        head = head.next
    print()
```

In [8]:
```
arr = [3, 4, 5, 2, 6, 1, 9]
head = create_linked_list(arr)
left_index = 3
right_index = 4

test_case = [head, left_index, right_index]
updated_head = test_function(test_case)
```

```
Pass
```

```
In [9]: arr = [3, 4, 5, 2, 6, 1, 9]
        left_index = 2
        right_index = 4
        head = create_linked_list(arr)
        test_case = [head, left_index, right_index]
        updated_head = test_function(test_case)
```

```
Pass
```

```
In [10]: arr = [3, 4, 5, 2, 6, 1, 9]
         left_index = 0
         right_index = 1
         head = create_linked_list(arr)
         test_case = [head, left_index, right_index]
         updated_head = test_function(test_case)
```

```
Pass
```