Skip-i-delete-j

May 14, 2020

0.0.1 Problem Statement

You are given the head of a linked list and two integers, `i` and `j`. You have to retain the first `i` nodes and then delete the next `j` nodes. Continue doing so until the end of the linked list.

Example: * `linked-list = 1 2 3 4 5 6 7 8 9 10 11 12` * `i = 2` * `j = 3` * `Output = 1 2 6 7 11 12`

```
In [ ]: # LinkedList Node class for your reference
        class Node:
            def __init__(self, data):
                self.data = data
                self.next = None
```

0.0.2 Exercise - Write the function definition here

```
In [ ]: def skip_i_delete_j(head, i, j):
            """
            :param: head - head of linked list
            :param: i - first `i` nodes that are to be skipped
            :param: j - next `j` nodes that are to be deleted
            return - return the updated head of the linked list
            """
            pass
```

Hide Solution

```
In [ ]: # Solution
        """
        :param: head - head of linked list
        :param: i - first `i` nodes that are to be skipped
        :param: j - next `j` nodes that are to be deleted
        return - return the updated head of the linked list
        """
        '''
        The Idea:
        Traverse the Linkedist. Make use of two references - `current` and `previous`.
         - Skip `i-1` nodes. Keep incrementing the `current`. Mark the `i-1`^th node as `previou
         - Delete next `j` nodes. Keep incrementing the `current`.
```

```
 - Connect the `previous.next` to the `current`
    '''
    def skip_i_delete_j(head, i, j):
        # Edge case - Skip 0 nodes (means Delete all nodes)
        if i == 0:
            return None

        # Edge case - Delete 0 nodes
        if j == 0:
            return head

        # Invalid input
        if head is None or j < 0 or i < 0:
            return head

        # Helper references
        current = head
        previous = None

        # Traverse - Loop untill there are Nodes available in the LinkedList
        while current:

            '''skip (i - 1) nodes'''
            for _ in range(i - 1):
                if current is None:
                    return head
                current = current.next
            previous = current
            current = current.next

            '''delete next j nodes'''
            for _ in range(j):
                if current is None:
                    break
                next_node = current.next
                current = next_node

            '''Connect the `previous.next` to the `current`'''
            previous.next = current

        # Loop ends

        return head
```

0.0.3 Test - Let's test your function

```
In [ ]: # 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 [ ]:
```
def test_function(test_case):
    head = test_case[0]
    i = test_case[1]
    j = test_case[2]
    solution = test_case[3]

    temp = skip_i_delete_j(head, i, j)
    index = 0
    try:
        while temp is not None:
            if temp.data != solution[index]:
                print("Fail")
                return
            index += 1
            temp = temp.next
        print("Pass")
    except Exception as e:
        print("Fail")
```

In [ ]:
```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
i = 2
j = 2
head = create_linked_list(arr)
solution = [1, 2, 5, 6, 9, 10]
test_case = [head, i, j, solution]
test_function(test_case)
```

In [ ]:
```
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
i = 2
j = 3
head = create_linked_list(arr)
solution = [1, 2, 6, 7, 11, 12]
test_case = [head, i, j, solution]
test_function(test_case)
```

In [ ]:
```
arr = [1, 2, 3, 4, 5]
i = 2
j = 4
head = create_linked_list(arr)
solution = [1, 2]
test_case = [head, i, j, solution]
test_function(test_case)
```

In [ ]:
```
arr = [1, 2, 3, 4, 5]
i = 2
j = 0
head = create_linked_list(arr)
solution = [1, 2, 3, 4, 5]
test_case = [head, i, j, solution]
test_function(test_case)
```