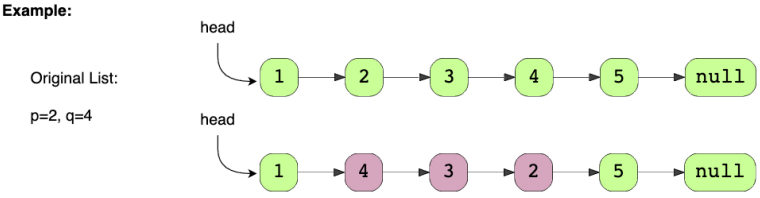
Reverse a Sub-list (medium)

We'll cover the following

Problem Statement

Given the head of a LinkedList and two positions 'p' and 'q', reverse the LinkedList from position 'p' to 'q'.

Example:



Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```java
import java.util.*;

class ListNode {
  int value = 0;
  ListNode next;

  ListNode(int value) {
    this.value = value;
  }
}

class ReverseSubList {

  public static ListNode reverse(ListNode head, int p, int q) {
    // TODO: Write your code here
    return head;
  }

  public static void main(String[] args) {
    ListNode head = new ListNode(1);
    head.next = new ListNode(2);
    head.next.next = new ListNode(3);
    head.next.next.next = new ListNode(4);
    head.next.next.next.next = new ListNode(5);

    ListNode result = ReverseSubList.reverse(head, 2, 4);
    System.out.print("Nodes of the reversed LinkedList are: ");
    while (result != null) {
```

Run | Save | Reset

Solution

The problem follows the **In-place Reversal of a LinkedList** pattern. We can use a similar approach as discussed in Reverse a LinkedList. Here are the steps we need to follow:

1. Skip the first `p-1` nodes, to reach the node at position `p`.
2. Remember the node at position `p-1` to be used later to connect with the reversed sub-list.
3. Next, reverse the nodes from `p` to `q` using the same approach discussed in Reverse a LinkedList.

4. Connect the `p-1` and `q+1` nodes to the reversed sub-list.

Code

Here is what our algorithm will look like:

Java | Python3 | C++ | JS

```java
import java.util.*;

class ListNode {
  int value = 0;
  ListNode next;

  ListNode(int value) {
    this.value = value;
  }
}

class ReverseSubList {

  public static ListNode reverse(ListNode head, int p, int q) {
    if (p == q)
      return head;

    // after skipping 'p-1' nodes, current will point to 'p'th node
    ListNode current = head, previous = null;
    for (int i = 0; current != null && i < p - 1; ++i) {
      previous = current;
      current = current.next;
    }

    // we are interested in three parts of the LinkedList, part before index 'p', part between 'p' and
    // 'q', and the part after index 'q'
    ListNode lastNodeOfFirstPart = previous; // points to the node at index 'p-1'
    // after reversing the LinkedList 'current' will become the last node of the sub-list
```

Run | Save | Reset

Time complexity

The time complexity of our algorithm will be $O(N)$ where 'N' is the total number of nodes in the LinkedList.

Space complexity

We only used constant space, therefore, the space complexity of our algorithm is $O(1)$.

Similar Questions

Problem 1: Reverse the first 'k' elements of a given LinkedList.

Solution: This problem can be easily converted to our parent problem; to reverse the first 'k' nodes of the list, we need to pass `p=1` and `q=k`.

Problem 2: Given a LinkedList with 'n' nodes, reverse it based on its size in the following way:

1. If 'n' is even, reverse the list in a group of n/2 nodes.
2. If n is odd, keep the middle node as it is, reverse the first 'n/2' nodes and reverse the last 'n/2' nodes.

Solution: When 'n' is even we can perform the following steps:

1. Reverse first 'n/2' nodes: `head = reverse(head, 1, n/2)`
2. Reverse last 'n/2' nodes: `head = reverse(head, n/2 + 1, n)`

When 'n' is odd, our algorithm will look like:

1. `head = reverse(head, 1, n/2)`
2. `head = reverse(head, n/2 + 2, n)`

Please note the function call in the second step. We're skipping two elements as we will be skipping the middle element.