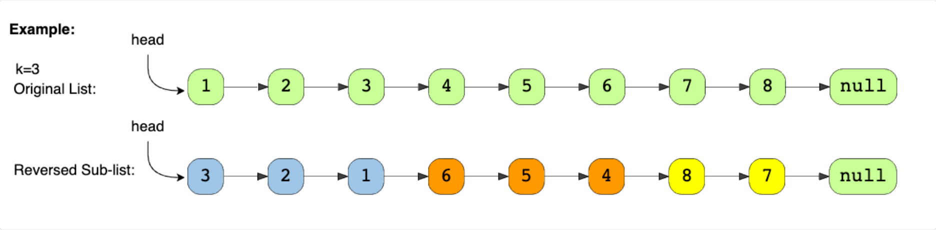
Reverse every K-element Sub-list (medium)

We'll cover the following

Problem Statement

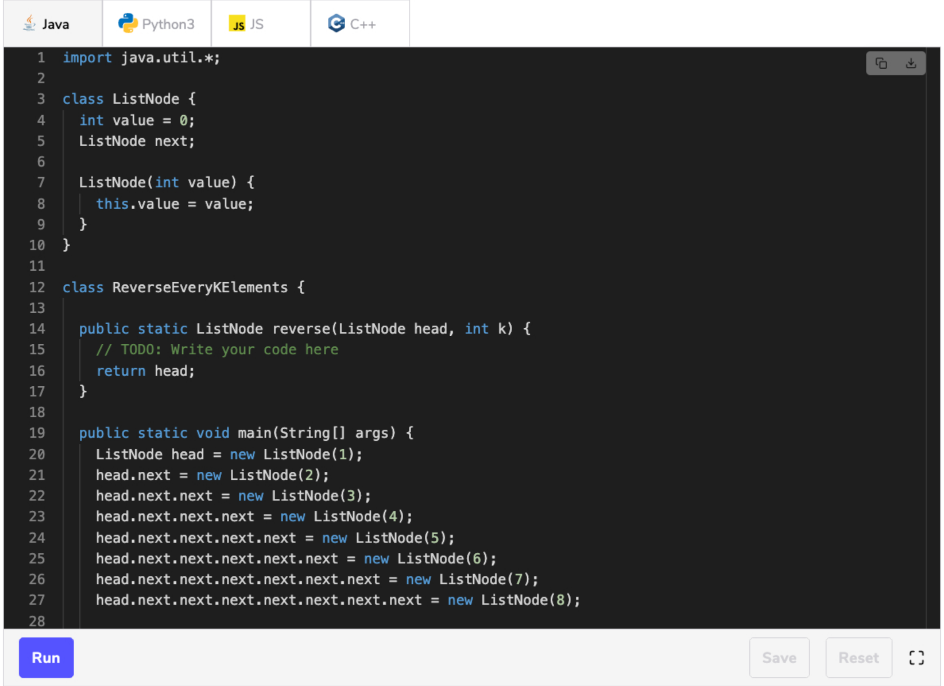
Given the head of a LinkedList and a number ‘k’, **reverse every ‘k’ sized sub-list** starting from the head.

If, in the end, you are left with a sub-list with less than ‘k’ elements, reverse it too.



Try it yourself

Try solving this question here:



```java
import java.util.*;

class ListNode {
  int value = 0;
  ListNode next;

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

class ReverseEveryKElements {

  public static ListNode reverse(ListNode head, int k) {
    // 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);
    head.next.next.next.next.next = new ListNode(6);
    head.next.next.next.next.next.next = new ListNode(7);
    head.next.next.next.next.next.next.next = new ListNode(8);

```

Solution

The problem follows the **In-place Reversal of a LinkedList** pattern and is quite similar to Reverse a Sub-list. The only difference is that we have to reverse all the sub-lists. We can use the same approach, starting with the first sub-list (i.e. `p=1, q=k`) and keep reversing all the sublists of size ‘k’.

Code

Most of the code is the same as Reverse a Sub-list; only the highlighted lines have a majority of the changes:

Java | Python3 | C++ | JS

```java
import java.util.*;

class ListNode {
  int value = 0;
  ListNode next;

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

class ReverseEveryKElements {

  public static ListNode reverse(ListNode head, int k) {
    if (k <= 1 || head == null)
      return head;

    ListNode current = head, previous = null;
    while (true) {
      ListNode lastNodeOfPreviousPart = previous;
      // after reversing the LinkedList 'current' will become the last node of the sub-list
      ListNode lastNodeOfSubList = current;
      ListNode next = null; // will be used to temporarily store the next node
      // reverse 'k' nodes
      for (int i = 0; current != null && i < k; i++) {
        next = current.next;
        current.next = previous;
        previous = current;
```

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 $O(1)$.

← Back
Reverse a Sub-list (medium)

Next →
Problem Challenge 1

☑ Mark as Completed

Report an Issue Ask a Question