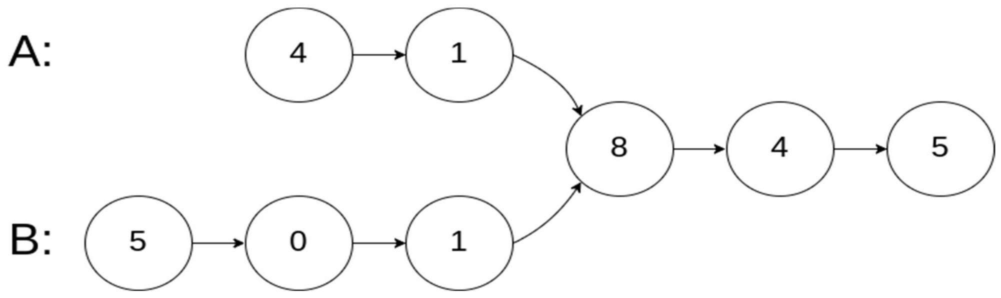
Intersection point in a Linked List :-



Algorithm: -

1- Finalize the base conditions
2- Get lengths of two linked lists
3- Start with LinkedList with larger length and iterate until the length difference.
4- Start iterating both LinkedLists until the next pointer becomes null
    a. If both nodes are equal in iteration, return true
    b. Else, return false.

Program: -

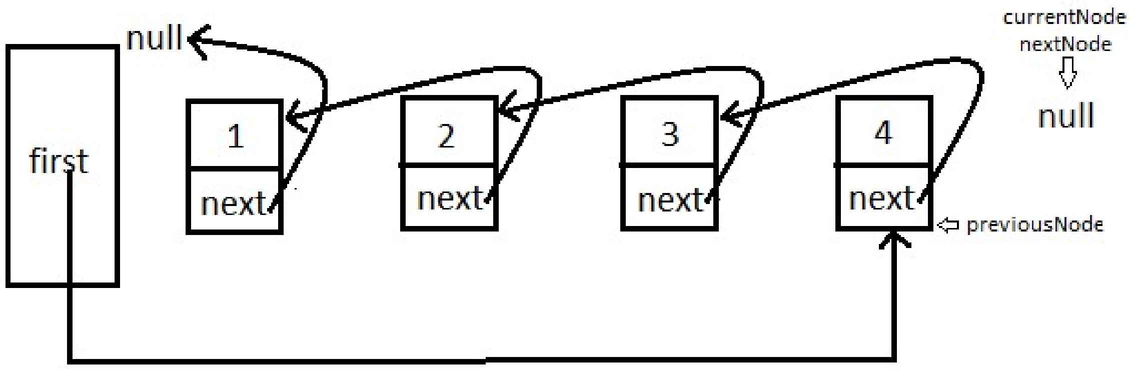
```
public class IntersectionPoint {
	public Node getIntersectionNode(Node headA, Node headB) {
	  //Base condition
		if(headA==null || headB==null) {
			return null;
		}
		// Step 1
		int lenA = getLength(headA);
		int lenB = getLength(headB);
		// Step 2
		Node p = headA;
		Node q = headB;
		while (lenA > lenB) {
			p = p.next;
			lenA--;
		}
		while (lenA < lenB) {
			q = q.next;
			lenB--;
		}
		// Step 3
		while (p != q) {
			p = p.next;
			q = q.next;
		}
		return p;
    }
```

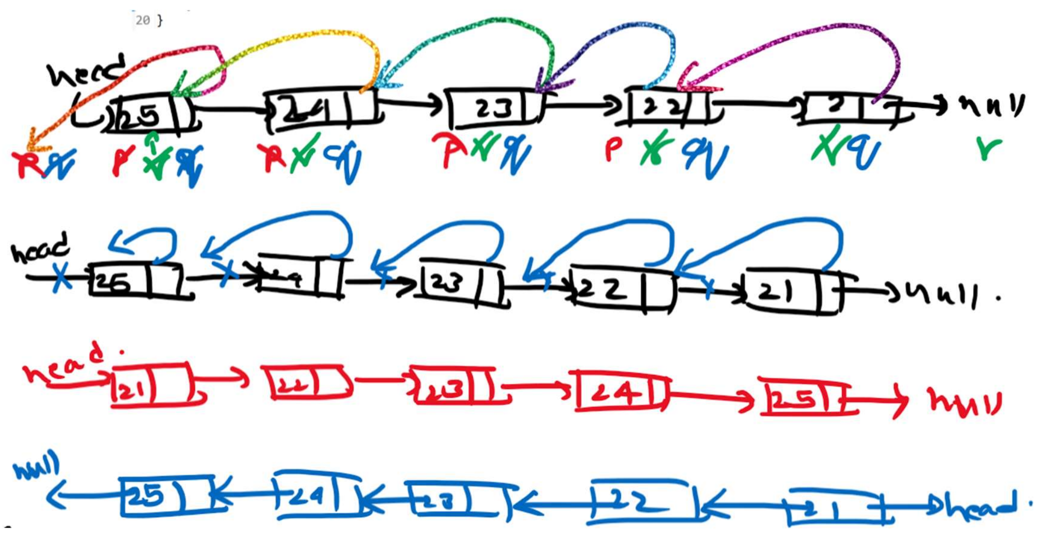
```
        private int getLength(Node node) {
                int length = 0;
                while (node != null) {
                        node = node.next;
                        length++;
                }
                return length;
        }
}
```

LeetCode Link: -

https://leetcode.com/problems/intersection-of-two-linked-lists/

Find reverse of a LinkedList: -

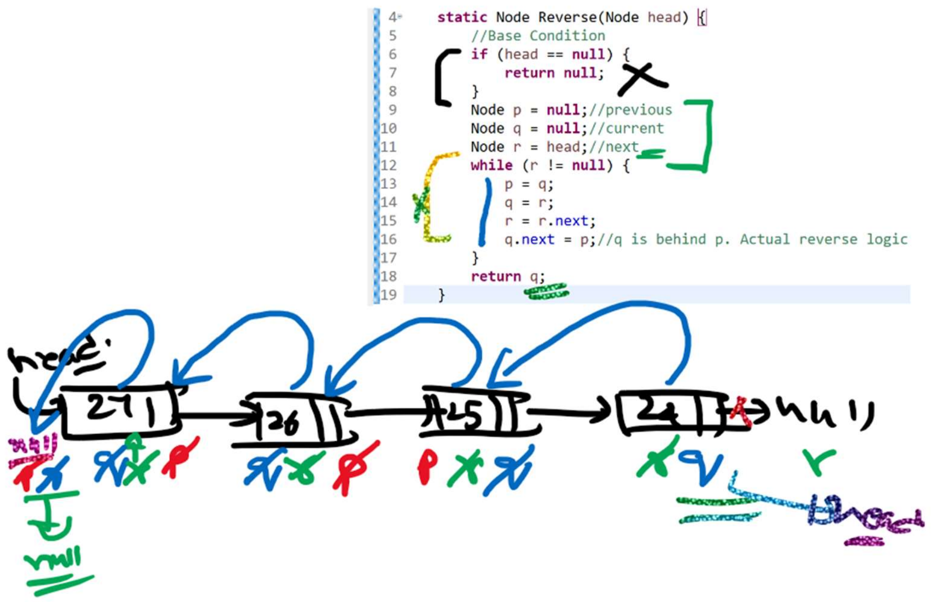




Merch Technologies

```java
public class ReverseLinkedList {
	static Node Reverse(Node head) {
		//Base Condition
		if (head == null) {
			return null;
		}
		Node p = null;//previous
		Node q = null;//current
		Node r = head;//next
		while (r != null) {
			p = q;
			q = r;
			r = r.next;
			q.next = p;//q is behind p. Actual reverse logic.
		}
		return q;
	}

	public static void main(String[] args) {
		Node head = new Node(28);
		Node node27 = new Node(27);
		Node node26 = new Node(26);
		Node node25 = new Node(25);
		head.next = node27;
		node27.next = node26;
		node26.next = node25;
		LinkedListTraversal.linkedListTraversal(head);
		Node reverseLinkedList = Reverse(head);
		LinkedListTraversal.linkedListTraversal(reverseLinkedList);
	}
}
```



```java
	static Node Reverse(Node head) {
		//Base Condition
		if (head == null) {
			return null;
		}
		Node p = null;//previous
		Node q = null;//current
		Node r = head;//next
		while (r != null) {
			p = q;
			q = r;
			r = r.next;
			q.next = p;//q is behind p. Actual reverse logic
		}
		return q;
	}
```

Hacker Rank Link: -

https://www.hackerrank.com/challenges/reverse-a-linked-list/problem

Merch Technologies

Merch Technologies