Print Reverse of Linked List

Shikha Nair

February 1, 2018

Question

Complete the method void ReversePrint(Node head);
Given the head node of a singly-linked list, print the elements of the list in reverse order.

Explanation and Algorithm

Iterative Method
There are a few ways to solve this iteratively, but we will look at one way specifically: this solution involves creating a second list which is the reverse of the given list. Then the new list is printed. We will reverse the list in place, and to do this, we need three temporary variables, which we will call curr, prev, and tmp. Traverse the given lit and for each node, save the next node in tmp, set curr.next to prev, and set prev to curr. Once this has been done for every node, you can now print the reversed list!

Recursive Method
The best way to solve this problem is using recursion. If the current node is null, then you should not print it, so this is our base case. Each call of the method will have head.next passed to it.

Hints

1. Since we are given the head of the list, how do we go to the end of the list to start printing?
2. Is there a way to iterate backwards through a linked list?
3. Is recursion useful for this problem?
4. Try using recursion. What is a possible base case for this problem?
5. What if your method encounters a null node?

Code

Iterative Method

```
void ReversePrint(Node head) {
       Node curr = head;
       Node prev = null;
       Node tmp = null;


       while(curr.next != null){

           tmp = curr.next;

           curr.next = prev;
           prev = curr;

           curr = tmp;
       }

       head = curr;
       head.next = prev;

    while(head!=null){
       System.out.println(head.data);
       head=head.next;
    }

}
```

Recursive Method

```
void ReversePrint(Node head){

   if(head == null)
      return;

   ReversePrint(head.next);

   System.out.println(head.data + " ");

}
```

Big O analysis

Iterative: This method 'reaches" each node once, and then iterates through again, so it will have O(2n) = O(n).

Recursive: Since this method 'reaches" each node one time, the run-time is O(n). However, as this is a recursive method, f the given list is very large it will take a significant amount of space and the iterative method will be more effective and efficient.

Sources

Question, answer and other material taken from HackerRank.com.