Merge Linked Lists (Recursion)

Shikha Nair

October 18, 2017

Question

Given the head nodes of two sorted linked lists, merge them into a single sorted linked list (in ascending order). This question focuses on a recursive solution, though it may be solved iteratively.

Explanation and Algorithm

For the recursive solution, our base case would be that we have reached the end (NULL node) of one or both of our lists. In this case, we would just return the non-empty list, as the rest of the nodes in that list are already in order and just need to be tacked on to the end of our result list. For the next step, we want to know which list head is smaller, as that is the next node we add to the result list. After setting the new result node, we recurse to find the next node.

Hints

1. What should our base case be for this algorithm?
2. The base case is probably when you have reached a null node. What should you do in this case?
3. Is there some work that needs to be done before the recursive call? For example, what does each call achieve?
4. If your call is supposed to set the current node, then the recursive call should set the next field of the current node.

Code

```
Node mergeLists(Node headA, Node headB) {
```

```
    Node sl;

    if(headA==null){
        return headB;
    }else if(headB==null){
        return headA;
    }

    if(headA.data <= headB.data){
        sl = headA;
        sl.next = mergeLists(headA.next, headB);
    }else{
        sl = headB;
        sl.next = mergeLists(headA, headB.next);
    }

    return sl;

}
```

Big O Analysis

Since this method 'reaches" each node one time, the run-time is O(n), where n is the size of the merged list. A recursive solution my not be space-efficient, however, as it will use stack space proportional to the size of the lists.

Sources

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