Question 2.2 from CTCI: Finding Kth to Last

September 11, 2017

Question

Implement an algorithm to find the nth to last element of a singly linked list.

Hints

1. Think brute force at first. Obviously this is not the optimal solution but it will help you get there. What are some easy ways to traverse a singly linked list? Think of the using two pointers, a first and second!

2. We're still thinking brute force here. What are some kinds of loops that we can use with n (nth last element) and the pointers to get one of them to point to the nth to last element? What are some conditions that we can use?

3. If you've figured out hint 1 and hint 2, we can start thinking of an optimal solution. Usually, recursion tends to be most efficient. What is a common base case?

Code

```
/* Optimal Solution: Recursive */
public int nthToLast(Node i, int n){
   if(i == null){
      return 0;
   }

   int k = nthToLast(i.next, n)+1

   if(n == k){
      System.out.println(i.data);
   }
```

```
   return k;
}
```

```
/* Brute Force Solution */
public LinkedListNode nthToLast(Node front, int n){

   if(front == null || n<1){
      return 0;
   }

   Node p1 = front; //first pointer

    /* traverse linked list until n has decremented to 0. Essentially
        traverse n times */
   while(n>0){
      p1 = p1.next;
      n--;
   }

   Node p2 = front; //second pointer

 /*move both pointers and when curr reaches the end of the list, sec
     will be nth to last*/
   while(p1.next != null){
      p1 = p1.next; //eventually reaches end of list
      p2 = p2.next; //eventually reaches nth to last element
   }

   return p2;

}
```

Big O Analysis

For optimal time, O(n), where n=length of the linked list. For brute force, O(n), where n=length of linked list.

Sources

Question, answer and other material taken from Cracking the Coding Interview 6th edition by Gayle Laakmann McDowell.