Reverse Linked List II

Reverse a linked list from position m to n. Do it in-place and in one-pass.

For example:
Given `1->2->3->4->5->NULL`, $m = 2$ and $n = 4$,

return `1->4->3->2->5->NULL`.

Note:
Given m, n satisfy the following condition:
$1 \le m \le n \le$ length of list.

Solution 1

Simply just reverse the list along the way using 4 pointers: dummy, pre, start, then

```
public ListNode reverseBetween(ListNode head, int m, int n) {
    if(head == null) return null;
    ListNode dummy = new ListNode(0); // create a dummy node to mark the head of this list
    dummy.next = head;
    ListNode pre = dummy; // make a pointer pre as a marker for the node before reversing
    for(int i = 0; i<m-1; i++) pre = pre.next;

    ListNode start = pre.next; // a pointer to the beginning of a sub-list that will be reversed
    ListNode then = start.next; // a pointer to a node that will be reversed

    // 1 - 2 -3 - 4 - 5 ; m=2; n =4 ---> pre = 1, start = 2, then = 3
    // dummy-> 1 -> 2 -> 3 -> 4 -> 5

    for(int i=0; i<n-m; i++)
    {
        start.next = then.next;
        then.next = pre.next;
        pre.next = then;
        then = start.next;
    }

    // first reversing : dummy->1 - 3 - 2 - 4 - 5; pre = 1, start = 2, then = 4
    // second reversing: dummy->1 - 4 - 3 - 2 - 5; pre = 1, start = 2, then = 5 (finish)

    return dummy.next;

}
```

written by ardyadipta original link here

Solution 2

```
ListNode *reverseBetween(ListNode *head, int m, int n) {
    if(m==n)return head;
    n-=m;
    ListNode prehead(0);
    prehead.next=head;
    ListNode* pre=&prehead;
    while(--m)pre=pre->next;
    ListNode* pstart=pre->next;
    while(n--)
    {
        ListNode *p=pstart->next;
        pstart->next=p->next;
        p->next=pre->next;
        pre->next=p;
    }
    return prehead.next;
}
```

written by harpe1999 original link here

Solution 3

The basic idea is as follows:

(1) Create a `new_head` that points to `head` and use it to locate the immediate node before the `m`-th (notice that it is `1`-indexed) node `pre`;

(2) Set `cur` to be the immediate node after `pre` and at each time move the immediate node after `cur` (named `move`) to be the immediate node after `pre`. Repeat it for `n - m` times.

```
class Solution {
public:
    ListNode* reverseBetween(ListNode* head, int m, int n) {
        ListNode* new_head = new ListNode(0);
        new_head -> next = head;
        ListNode* pre = new_head;
        for (int i = 0; i < m - 1; i++)
            pre = pre -> next;
        ListNode* cur = pre -> next;
        for (int i = 0; i < n - m; i++) {
            ListNode* move = cur -> next;
            cur -> next = move -> next;
            move -> next = pre -> next;
            pre -> next = move;
        }
        return new_head -> next;
    }
};
```

written by jianchao.li.fighter original link here