328. Odd Even Linked List

April 3, 2016 | 175.9K views

Average Rating: 4.66 (124 votes)

Given a singly linked list, group all odd nodes together followed by the even nodes. Please note here we are talking about the node number and not the value in the nodes.

You should try to do it in place. The program should run in O(1) space complexity and O(nodes) time complexity.

Example 1:

```
Input: 1->2->3->4->5->NULL
Output: 1->3->5->2->4->NULL
```

Example 2:

```
Input: 2->1->3->5->6->4->7->NULL
Output: 2->3->6->7->1->5->4->NULL
```

Constraints:

- The relative order inside both the even and odd groups should remain as it was in the input.
- The first node is considered odd, the second node even and so on ...
- The length of the linked list is between `[0, 10^4]`.

Solution

Intuition

Put the odd nodes in a linked list and the even nodes in another. Then link the evenList to the tail of the oddList.

Algorithm

The solution is very intuitive. But it is not trivial to write a concise and bug-free code.

A well-formed `LinkedList` need two pointers head and tail to support operations at both ends. The variables `head` and `odd` are the head pointer and tail pointer of one `LinkedList` we call oddList; the variables `evenHead` and `even` are the head pointer and tail pointer of another `LinkedList` we call evenList. The algorithm traverses the original LinkedList and put the odd nodes into the oddList and the even nodes into the evenList. To traverse a LinkedList we need at least one pointer as an iterator for the current node. But here the pointers `odd` and `even` not only serve as the tail pointers but also act as the iterators of the original list.

The best way of solving any linked list problem is to visualize it either in your mind or on a piece of paper. An illustration of our algorithm is following:

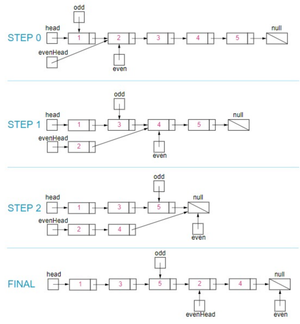

Figure 1. Step by step example of the odd and even linked list.

Java

```java
public class Solution {
    public ListNode oddEvenList(ListNode head) {
        if (head == null) return null;
        ListNode odd = head, even = head.next, evenHead = even;
        while (even != null && even.next != null) {
            odd.next = even.next;
            odd = odd.next;
            even.next = odd.next;
            even = even.next;
        }
        odd.next = evenHead;
        return head;
    }
}
```

Complexity Analysis

- Time complexity : $O(n)$. There are total n nodes and we visit each node once.
- Space complexity : $O(1)$. All we need is the four pointers.

Rate this article:

Previous Next

Comments: 35

Sort By

Type comment here... (Markdown is supported)

Preview Post

Aris94 ★62 April 18, 2018 5:34 AM

I believe my Python solution is a bit more readable. It's more clear in my mind. if I just think of each iteration of the loop as a toggle between odd and even.

```
def oddEvenList(self, head):
```

Read More

61 Share Reply

SHOW 10 REPLIES

patroniusgremmy ★6 January 5, 2018 4:25 AM

Was getting a timelimit error on the [1, 2, 3] case even though the first test case passed. I added a null to the end of the returned list and it solved the problem. It might be some test cases expect a null to be the last node whereas the first test case didn't. The example in the instructions show a null as the last linked list so you want to make sure to add that. It hung me up for a while.

4 Share Reply

SHOW 2 REPLIES

vanderpuye ★17 April 13, 2018 2:31 AM

I'm having an issue with the test cases. After I submitted my solution, this was the feedback I had

Input:
[2,1,4,3,6,5,7,8]
Output:

Read More

5 Share Reply

SHOW 4 REPLIES

srigopalsai ★5 November 13, 2016 2:29 PM

This is what I tried. Iterate list with current node with a loop counter. if loop counter is even then add current node to evenlist otherwise add to oddlist.

```
public ListNode OddEvenList(ListNode head)
{
```

Read More

2 Share Reply

Mrmagician ★333 March 26, 2020 5:25 PM

Damm. **this question is Hard!!!**

3 Share Reply

SHOW 1 REPLY

ronaksengupta ★1 March 28, 2020 9:17 PM

Solution in C++
class Solution {
public:
ListNode* oddEvenList(ListNode* head)
{

Read More

1 Share Reply

arun279 ★0 February 22, 2019 3:09 AM

My Python solution:

```
def oddEvenList(self, head):
    if(head is None or head.next is None):
        return head
```

Read More

0 Share Reply

shashikiran47 ★0 March 13, 2018 9:17 PM

@lucas.yang.357 : Yes. Since we are not creating any new nodes here. space complexity is O(1). we are just creating three new pointers overall irrespective of the input. So the space remains constant.

0 Share Reply

SHOW 1 REPLY

lucasyang357 ★11 January 30, 2018 3:28 AM

The even list has n/2 nodes. Why does your solution has O(1) space complexity? Is it just because you created a new evenHead pointer and did not create any new listNodes in memory?

0 Share Reply

SHOW 1 REPLY

vegito2002 ★1273 January 15, 2018 10:17 AM

My slightly different solution **here**, illustrated and explained. My approach does not rely on building a separate list on the side, but rather just relies on deleting and inserting. not that they are so much different in essence though.

Admittedly, I do think this solution is more elegant than mine.

Read More

0 Share Reply