CSC 3100 Data Structures

Tutorial 4 Linked Lists

Mickey Ma

E-mail: mickeyma@cuhk.edu.cn

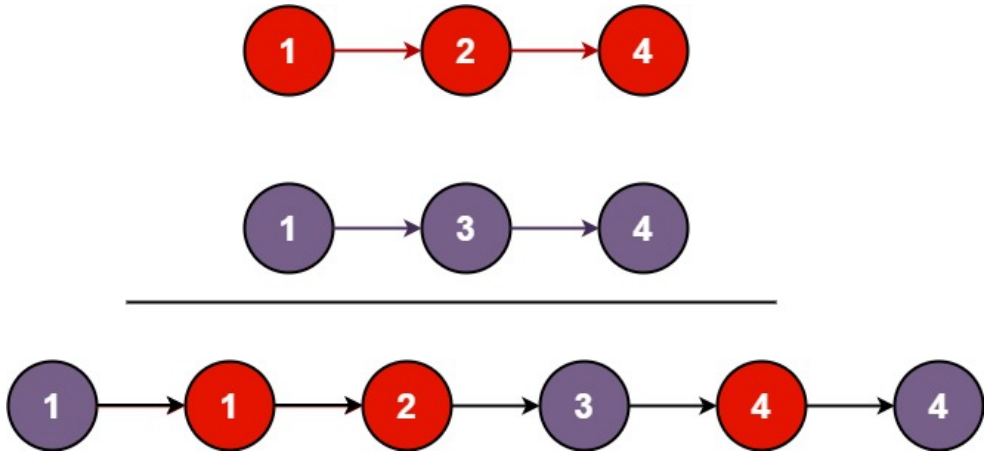
Daoyuan Bldg 508

Office Hour: We. 16:00~17:00

Lists - Examples

Ex. 1 Merge Two Sorted Lists

Merge two sorted linked lists and return it as a sorted list. The list should be made by splicing together the nodes of the first two lists.



Lists - Examples

Ex. 1 Merge Two Sorted Lists

Merge two sorted linked lists and return it as a sorted list. The list should be made by splicing together the nodes of the first two lists.

```
Input: l1 = [1,2,4], l2 = [1,3,4]
Output: [1,1,2,3,4,4]

Input: l1 = [], l2 = []
Output: []

Input: l1 = [], l2 = [0]
Output: [0]
```

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode mergeTwoLists(ListNode l1, ListNode l2) {

    }
}
```

Constraints:

- The number of nodes in both lists is in the range `[0, 50]`.
- `-100 <= Node.val <= 100`
- Both `l1` and `l2` are sorted in **non-decreasing** order.

Analysis

Ex. 1 Merge Two Sorted Lists

This is a simple example for linked lists.

Iteration: Compare the first node of these two linked lists. If the value is smaller, move to the second node and repeat comparing. Use this node with smaller value to construct a new list.

Recursion: Compare the first node of these two linked list. After comparison, choose the node with smaller value to form the output list and link it to the new output of the function.

Solutions

Ex. 1 Merge Two Sorted Lists - iteration

```
class Solution {
    public ListNode mergeTwoLists(ListNode l1, ListNode l2) {
        ListNode head = new ListNode(0);
        ListNode cur = head;
        while (l1 != null && l2 != null) {
            if (l1.val < l2.val){
                cur.next = l1;
                l1 = l1.next;
            }
            else{
                cur.next = l2;
                l2 = l2.next;
            }
            cur = cur.next;
        }
        if (l1 == null){
            cur.next = l2;
        }
        else{
            cur.next = l1;
        }
        return head.next;
    }
}
```

Solutions

Ex. 1 Merge Two Sorted Lists - recursion

```
class Solution {
    public ListNode mergeTwoLists(ListNode l1, ListNode l2) {
        if (l1 == null) return l2;
        if (l2 == null) return l1;
        ListNode resNode = new ListNode();
        if (l1.val < l2.val) {
            resNode = l1;
            resNode.next = mergeTwoLists(l1.next, l2);
        }
        else {
            resNode = l2;
            resNode.next = mergeTwoLists(l1, l2.next);
        }
        return resNode;
    }
}
```

Lists - Examples

Ex. 2 Linked List Cycle II

Given the head of a linked list, return the node where the cycle begins. If there is no cycle, return null.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer.

Do not modify the linked list.

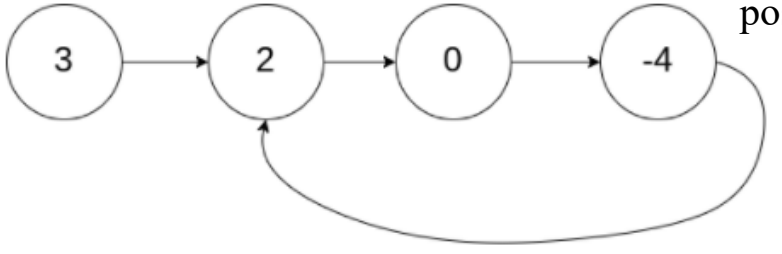
```
1  /**
2   * Definition for singly-linked list.
3   * class ListNode {
4   *     int val;
5   *     ListNode next;
6   *     ListNode(int x) {
7   *         val = x;
8   *         next = null;
9   *     }
10  * }
11  */
12 public class Solution {
13     public ListNode detectCycle(ListNode head) {
14
15     }
16 }
```

Lists - Examples

Ex. 2 Linked List Cycle II

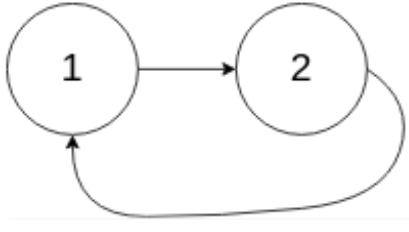
Internally, pos is used to denote the index of the node that tail's next pointer is connected to (0-indexed). It is -1 if there is no cycle. Note that pos is not passed as a parameter.

Example 1:




```
Input: head = [3,2,0,-4], pos = 1
Output: tail connects to node index 1
Explanation: There is a cycle in the
linked list, where tail connects to
the second node.
```

Example 2:




```
Input: head = [1,2], pos = 0
Output: tail connects to node index 0
Explanation: There is a cycle in the linked
list, where tail connects to the first node.
```

Example 3:


```
Input: head = [1], pos = -1
Output: no cycle
Explanation: There is no cycle
in the linked list.
```

Constraints:

- The number of the nodes in the list is in the range $[0, 10^4]$.
- $-10^5 <= Node.val <= 10^5$
- `pos` is `-1` or a **valid index** in the linked-list.

Analysis

Ex. 2 Linked List Cycle II

We should think about two questions:

- How to determine whether there is a cycle?

  Fast and slow pointer.

- How can we find the node where the cycle begin?

  Find the relationship between:
  - number of nodes before the target node
  - number of nodes which form the cycle
  - steps of the pointers

Solutions

Ex. 2 Linked List Cycle II – sample code

```
public class Solution {
    public ListNode detectCycle(ListNode head) {
        ListNode slowPointer = head;
        ListNode fastPointer = head;
        while (fastPointer != null && fastPointer.next != null){
            slowPointer = slowPointer.next;
            fastPointer = fastPointer.next.next;
            if (slowPointer == fastPointer){
                ListNode firstPtr = head;
                ListNode secondPtr = fastPointer;
                while (firstPtr != secondPtr){
                    firstPtr = firstPtr.next;
                    secondPtr = secondPtr.next;
                }
                return firstPtr;
            }
        }
        return null;
    }
}
```

References

- Merge Two Sorted Lists – LeetCode
  https://leetcode.com/problems/merge-two-sorted-lists/

- Linked List Cycle II – LeetCode
  https://leetcode.com/problems/linked-list-cycle-ii/