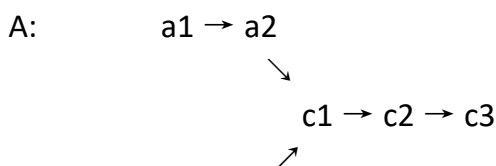


160 Intersection of Two Linked Lists

2018年4月3日 15:27

Question:

Write a program to find the node at which the intersection of two singly linked lists begins. For example, the following two linked lists:



B: b1 → b2 → b3
begin to intersect at node c1.

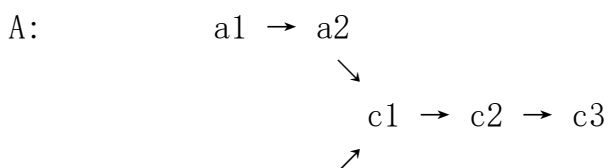
Notes:

- If the two linked lists have no intersection at all, return null.
- The linked lists must retain their original structure after the function returns.
- You may assume there are no cycles anywhere in the entire linked structure.
- Your code should preferably run in $O(n)$ time and use only $O(1)$ memory.

Credits:

来自 <<https://leetcode.com/problems/intersection-of-two-linked-lists/description/>>

编写一个程序，找到两个单链表相交的起始节点。
例如，下面的两个链表：



B: b1 → b2 → b3
在节点 c1 开始相交。

注意:

- 如果两个链表没有交点，返回 null.
- 在返回结果后，两个链表仍须保持原有的结构。
- 可假定整个链表结构中没有循环。
- 程序尽量满足 $O(n)$ 时间复杂度，且仅用 $O(1)$ 内存。

Solution for Python3:

```
1  # Definition for singly-linked list.
2  # class ListNode(object):
3  #     def __init__(self, x):
4  #         self.val = x
5  #         self.next = None
6
7  class Solution(object):
8      def getIntersectionNode(self, headA, headB):
9          """
```

```

10         :type head1, head1: ListNode
11         :rtype: ListNode
12         """
13         p1, p2 = headA, headB
14         if not p1 or not p2:
15             return None
16         while p1 != p2:
17             p1 = p1.next
18             p2 = p2.next
19             if p1 == p2:
20                 return p1
21             if not p1:
22                 p1 = headB
23             if not p2:
24                 p2 = headA
25         return p1

```

Solution for C++:

```

1  /**
2   * Definition for singly-linked list.
3   * struct ListNode {
4   *     int val;
5   *     ListNode *next;
6   *     ListNode(int x) : val(x), next(NULL) {}
7   * };
8   */
9  class Solution {
10 public:
11     ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {
12         ListNode *p1 = headA;
13         ListNode *p2 = headB;
14         if (!p1 || !p2) {
15             return NULL;
16         }
17         while (p1 != p2) {
18             p1 = p1->next;
19             p2 = p2->next;
20             if (p1 == p2) {
21                 // 有交点: 可能是一开始长度相同, 中间某个点相遇
22                 // 可能是到各自链表后在中间相遇
23                 // 没交点: 可能是一开始长度相同, 各自均到末尾NULL相等
24                 // 可能是到各自链表后再到各自末尾NULL相等
25                 return p1;
26             }
27         }
28         if (!p1) {
29             p1 = headB;

```

```
30         }
31         if (!p2) {
32             p2 = headA;
33         }
34     }
35     return p1;
36 }
};
```