160. Intersection of Two Linked Lists (/problems/intersection-of-two-linked-lists/)

March 9, 2016 | 36.2K views

Write a program to find the node at which the intersection of two singly linked lists begins.

For example, the following two linked lists:

```
A:          a1 → a2
                   ↘
                     c1 → c2 → c3
                   ↗
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:
Special thanks to @stellari (https://oj.leetcode.com/discuss/user/stellari) for adding this problem and creating all test cases.

Solution

Approach #1 (Brute Force) [Time Limit Exceeded]

For each node a_i in list A, traverse the entire list B and check if any node in list B coincides with a_i.

Complexity Analysis

- Time complexity : $O(mn)$.
- Space complexity : $O(1)$.

Approach #2 (Hash Table) [Accepted]

Traverse list A and store the address / reference to each node in a hash set. Then check every node b_i in list B: if b_i appears in the hash set, then b_i is the intersection node.

Complexity Analysis

- Time complexity : $O(m + n)$.
- Space complexity : $O(m)$ or $O(n)$.

Approach #3 (Two Pointers) [Accepted]

- Maintain two pointers pA and pB initialized at the head of A and B, respectively. Then let them both traverse through the lists, one node at a time.
- When pA reaches the end of a list, then redirect it to the head of B (yes, B, that's right.); similarly when pB reaches the end of a list, redirect it the head of A.
- If at any point pA meets pB, then pA/pB is the intersection node.
- To see why the above trick would work, consider the following two lists: A = {1,3,5,7,9,11} and B = {2,4,9,11}, which are intersected at node '9'. Since B.length (=4) < A.length (=6), pB would reach the end of the merged list first, because pB traverses exactly 2 nodes less than pA does. By redirecting pB to head A, and pA to head B, we now ask pB to travel exactly 2 more nodes than pA would. So in the second iteration, they are guaranteed to reach the intersection node at the same time.
- If two lists have intersection, then their last nodes must be the same one. So when pA/pB reaches the end of a list, record the last element of A/B respectively. If the two last elements are not the same one, then the two lists have no intersections.

Complexity Analysis

- Time complexity : $O(m + n)$.
- Space complexity : $O(1)$.

Analysis written by @stellari.