Intersection of Two Linked Lists

Contributed by
Arshit Babariya

Easy | 0/40 | 25 mins | 73 %

197 upvotes

+69 more

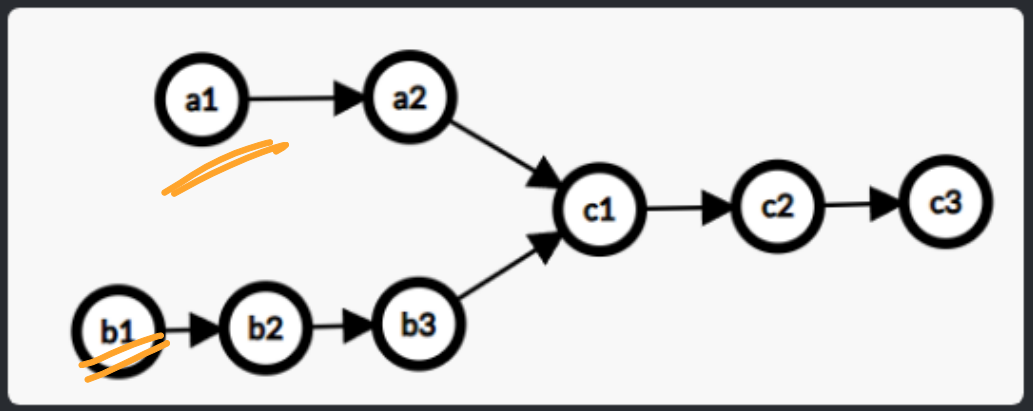
Problem Statement

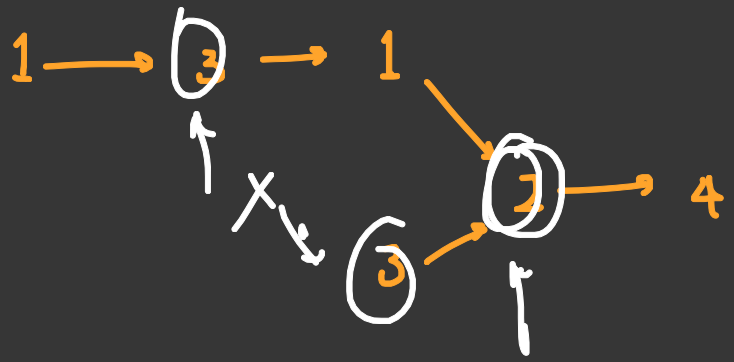
Suggest Edit

You are given two Singly Linked Lists of integers, which may have an intersection point.
Your task is to return the first intersection node. If there is no intersection return NULL.

Example:-

```
The Linked Lists, where a1, a2, c1, c2, c3 is the
first linked list and b1, b2, b3, c1, c2, c3 is the
second linked list, merging at node c1.
```



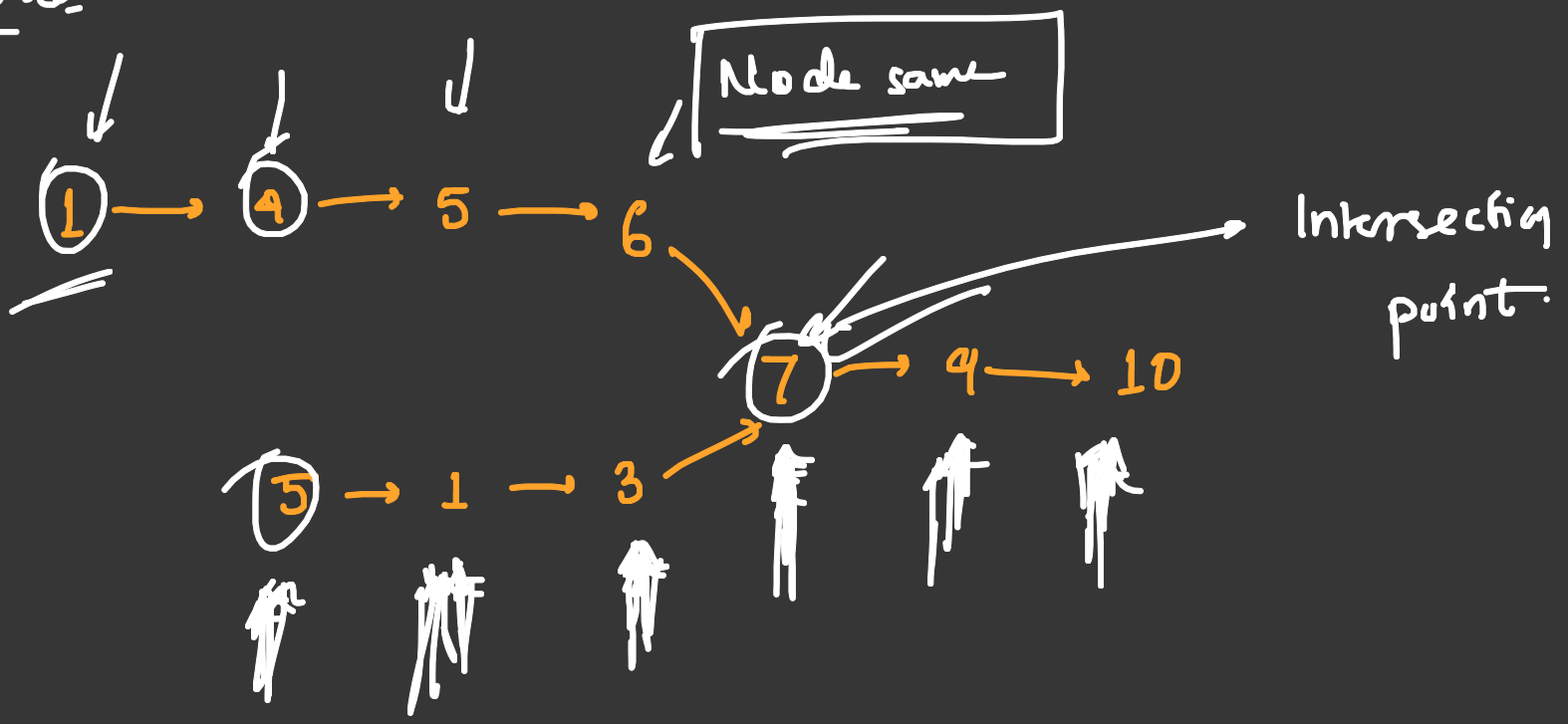


1, 3, 1, 2, 4 val

3, 2, 4

- We need to check the node instead of node value.

Brute force:



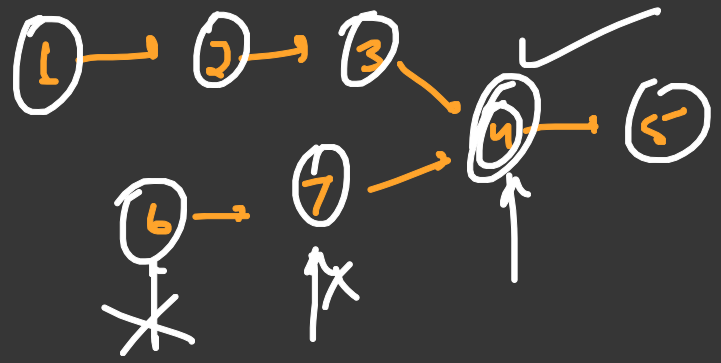
Iterate in one linked list and check similar node in other linked for each node. T.C. O(N×M)

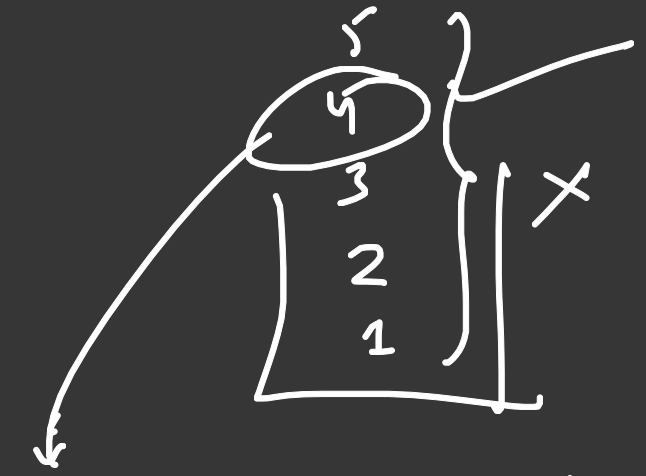
Better Approach:

We store one linked list in hashmap and check other LL if that is any node from second present in hashmap.

T.C. → O(N+M)

S.C → O(N) or O(M)

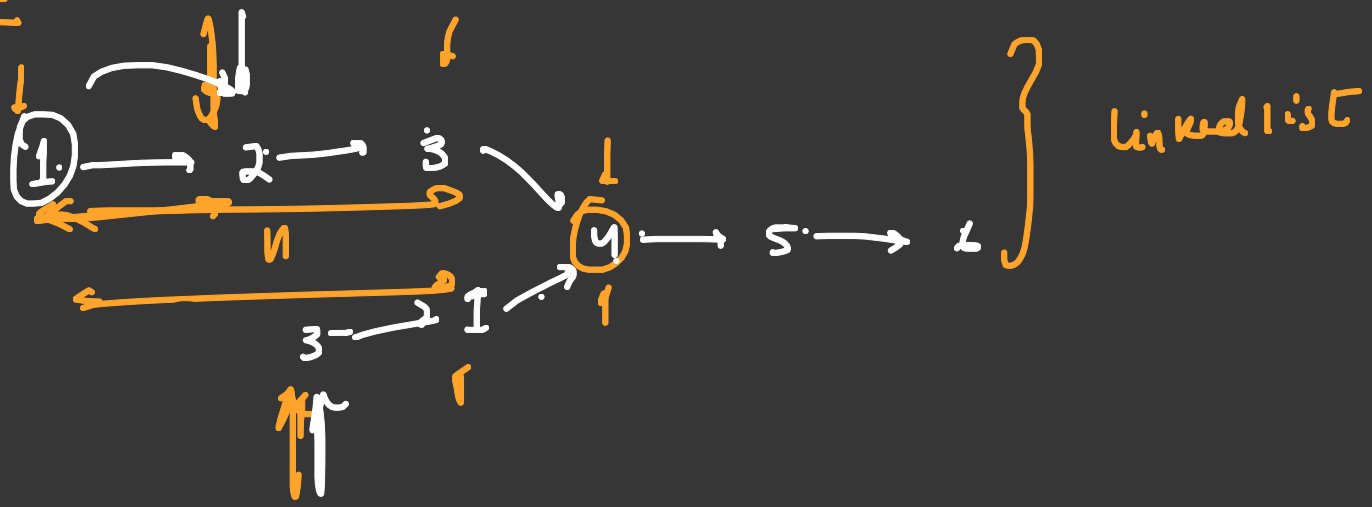




Return this as intersection point.

Optimal Approach:

Idea:



len of linked list 1 and list 2.

6 — 5 — 1

Simultaneously move both pointer such that they reach intersection point at same time.

1st Approach :

- find len of List 1
  " " " List 2
  } subtract = diff

- curr → headA / headB

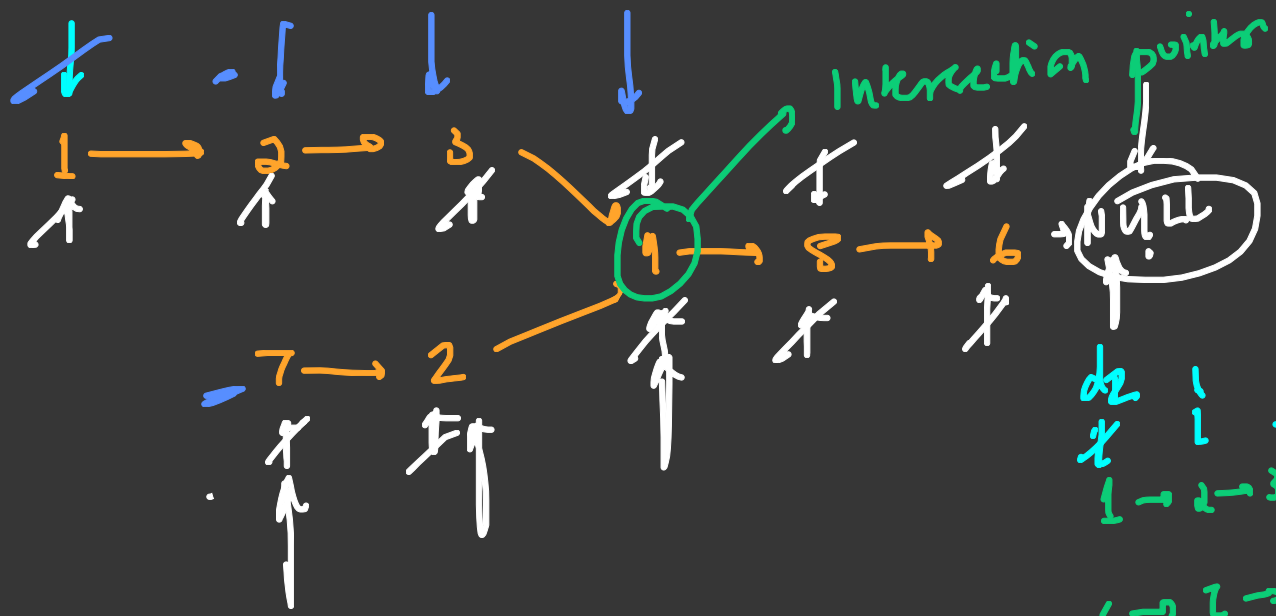
- Simultaneously move the pointer to get the intersection node.

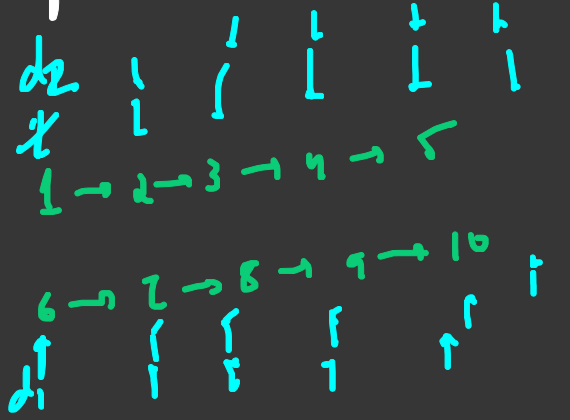
  T.C → O(2(N+m))

- 2nd Approach is similar idea but without finding no of nodes. (S.C → O(1))

  Code length is slightly shorter.

2nd Approach:





any pointer reaches NULL

if list 1 → NULL → pointer → head of list 2

list 2 → NULL →