Intersection of 2 linked lists

---

*Leetcode Link :

GFG Link :

$Approach-1$(⚠TLE chances)

code :-

```cpp
 #include<unordered_set>
class Solution {
    /* ✔Approach - 1  (using 2 vectors<node*> )

        T : O(n*m)
        S : O(n+m)

    */

public:
    ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {

        // create 2 vector<Node*> and store both lists into the sets
        vector<ListNode*> list1;
        vector<ListNode*> list2;

        ListNode* temp1 = headA;
        ListNode* temp2 = headB;
        while(temp1){
            list1.push_back(temp1);
            temp1 = temp1 -> next;
        }

        while(temp2){
            list2.push_back(temp2);
            temp2 = temp2 -> next;
        }

        // traverse list1 , and check if its node is present in list2 , if yes then we
found the intersetion
        for(ListNode* node1:list1){
            for(ListNode* node2:list2){
                if(node1 == node2) return node1;
            }
        }

        return nullptr;

    }

};
```

★ $Approach - 2$ (In Place Solution)

Solved in place (no extra space)

explanation :-

```
firstly we want to know if the 2 linked list have a intersection or not, now how to
check that? its very simple we would go to the tails of both the linked list, and
check if the tail1, tail2 pointers point to same node or not , and if not that means
that no intersection is present simply return nullptr



now if there is a intersection present then how to find that node?

for that we need to compare the lists, but both the linked lists can be of different
length like shown below, so

How to set the temp1, temp2 for the perfect comparision

        list1    8 -> 6 -> 5

                            \

                            4 -> 3 -> 2

                            /

             list2       10



                so to we need to put the temp2 somewhere from where the right nodes of
both the lists are equal number, lets put temp1 on 5 and temp2 on 10
```

Complexity

- Time complexity: $O(n)$
- Space complexity: $O(1)$

Code

```
class Solution {

public:
    ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {


        // step 1 : if both the linked list have different tails that means that there
is no intersection so return nullptr
        ListNode* tail1 = headA, *tail2 = headB;
        int list1Length = 1;
        int list2Length = 1;
        while(tail1 -> next){
            tail1 = tail1 -> next;
            list1Length++;
        }
        while(tail2 -> next){
            tail2 = tail2 -> next;
            list2Length++;
        }
        if(tail1 != tail2) return nullptr;

        // step 2 : now as i explained at start, we will set both the temp pointers
according to their lengths in such a way that after setting them both the number of
nodes on the right of linked list becomes equal
        ListNode* temp1 = headA;
        ListNode* temp2 = headB;
        if(list1Length > list2Length){ // when list1 is longer
            while(list1Length > list2Length){
                temp1 = temp1 -> next;
                list1Length--;
            }

        }
        else{
            while(list2Length > list1Length){
                temp2 = temp2 -> next;
                list2Length--;
            }
        }

        // step 3 : now our temp1, temp2 are all set, now lets compare both lists and
find the intersectoon, how? lets just move the temp1 and temp2 equal steps, and
whereever both temps are pointing to the same node, that means we found our
intersection
        while(temp1 != temp2){
            temp1 = temp1 -> next;
            temp2 = temp2 -> next;
        }

        return temp1;
    }
};
```

---END--