Data Analytics Lab

Linked List Components

Abbos Aliboev 2023041080

2025.01.09

Problem Definition (1)

- Source: Leetcode - 817
- Title: Linked List Components
- Difficulty: Medium

Array, Hash Table, Linked List, Data Structer

Problem Definition (2)

You are given the **head** of a linked list containing unique integer values and an integer array **nums** that is a subset of the linked list values.

Return *the number of connected components in* **nums** *where two values are connected if they appear* ***consecutively*** *in the linked list.*

Problem Definition (2)

Example 1:
Input: head = [0,1,2,3], nums = [0,1,3]
Output: 2
Explanation: 0 and 1 are connected, so [0, 1] and [3] are the two connected components.

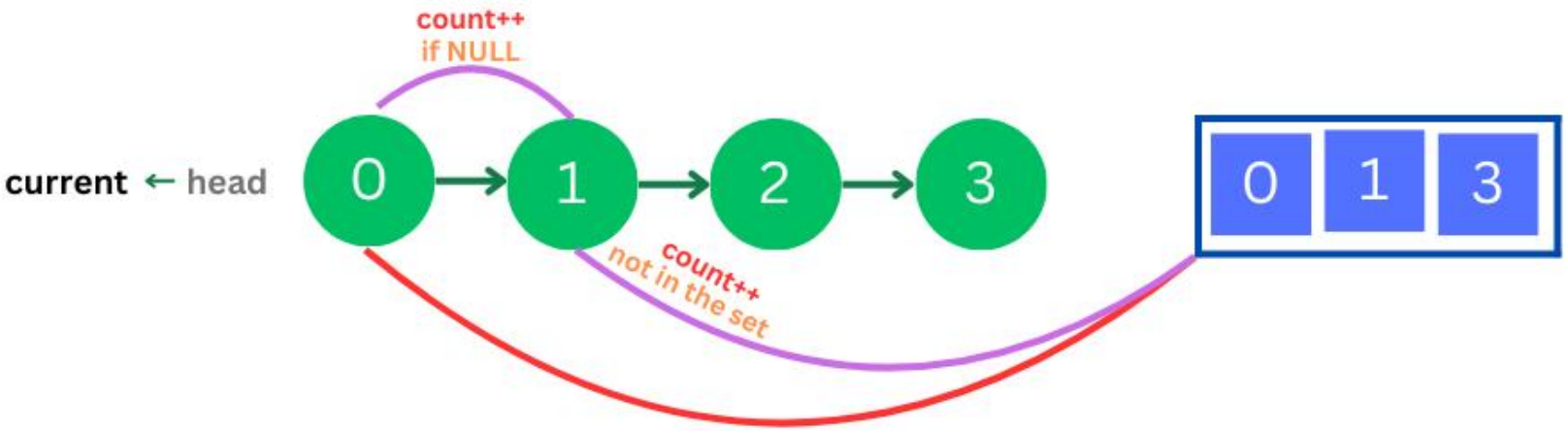
Example 2:
Input: head = [0,1,2,3,4], nums = [0,3,1,4]
Output: 2
Explanation: 0 and 1 are connected, 3 and 4 are connected, so [0, 1] and [3, 4] are the two connected components.

Constraints:
The number of nodes in the linked list is **n**.
1 <= n <= 104
0 <= Node.val < n
All the values Node.val are **unique**.
1 <= nums.length <= n
0 <= nums[i] < n
All the values of nums are **unique**.

Solution (1-0)

1. **Insert nums into a set** for fast membership checks (O(1) on average).
2. **Traverse the linked list** from head to the end.
3. **Check membership:** If current->val is in the set, but the next node is either nullptr or not in the set, increment the component count.
4. **Return the count** of connected components.

Solution (1-1)



Solution (2-0)

Code

C++ Auto

```
5   *     ListNode *next;
6   *     ListNode() : val(0), next(nullptr) {}
7   *     ListNode(int x) : val(x), next(nullptr) {}
8   *     ListNode(int x, ListNode *next) : val(x), next(next) {}
9   * };
10  */
11  class Solution {
12  public:
13      int numComponents(ListNode* head, vector<int>& nums) {
14          unordered_set<int> s(nums.begin(), nums.end());
15
16          ListNode* current = head;
17          int count = 0;
18
19          while (current != NULL){
20              if(s.count(current->val)){
21                  if(current->next == NULL || !s.count(current->next->val)){
22                      count++;
23                  }
24              }
25              current = current->next;
26          }
27
28          return count;
29      }
30  };
```

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

head =

[0,1,2,3,4]

nums =

[0,3,1,4]

Output

2

Expected

2

What you have learned

1. Learned the differences between **set** (ordered) and **unordered_set** (hash-based), and how they impact insertion and lookup complexities.
2. Explored how searching in an array (or vector) typically uses linear or binary search, depending on data ordering.
3. Practiced comparing these approaches in terms of efficiency and how they handle duplicates.
4. Gained insights into selecting the right data structure for optimal performance in various scenarios.

Questions and Answers

Greetings