141 linked list cycle

Saturday, November 28, 2020 10:26 AM

Given head, determine if an linked list has a cycle.

Eg 1.

Input: head = [3, 2, 0, -4], pos = 1.

Output: true.

Definition of singly-linked list

```
Struct ListNode{
    int val;
    ListNode *next;
    ListNode(int x): val(x), next(NULL) {}
};
```

解题思路:

快慢指针. 两个运动员在环形跑道上跑, 一快一慢, 快的运动员一定会追上慢的(套圈).

时间复杂度 O(n), 空间复杂度 O(1).

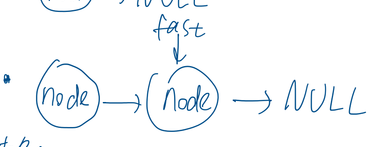
解法:

```
class Solution {
public:
    bool hasCycle(ListNode *head) {

        // base case
        if (head == NULL)
            return false;

        ListNode *slow = head, *fast = head;

        while (fast != NULL && fast->next != NULL) {

            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast)
                return true;

        }

        return false;
    }
};
```

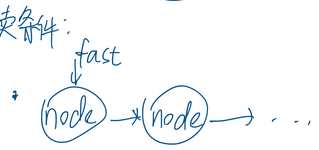
注意: 别写成 `ListNode* slow = head, fast = head;`

- 使用 fast 和 fast->next 不为 NULL 作为终止条件, 因为如果有环, fast 必定会绕跑道跑两圈, 并追上 slow. (至少两圈.)
- 如果无环, fast 一定会到达 NULL

终止条件:



继续条件:



另外一种解法:

使用 HashTable, 记录下遍历过的元素.

时间复杂度 O(n), 空间复杂度 O(n).

```
class Solution {
public:
    bool hasCycle(ListNode *head) {

        unordered_set<ListNode*> seen;

        while (head) {
            if (seen.count(head))
                return true;

            seen.insert(head);
            head = head->next;
        }

        return false;

    }
};
```

- unordered_set_name.count(element) 用来检查元素是否存在于 set 中. 如果存在, 返回 1.