链表

面试题02.07 链表相交

面试题 02.07. 链表相交 - 力扣

给你两个单链表的头节点 `headA` 和 `headB` ，请你找出并返回两个单链表相交的起始节点。如果两个链表没有交点，返回 `null` 。

题目数据 **保证** 整个链式结构中不存在环。

```
class Solution {
public:
    ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {
        int lenA = 0;
        int lenB = 0;
        ListNode* nodeA = headA;
        ListNode* nodeB = headB;

        ListNode* cur = headA;
        while (cur!=nullptr)
        {
            cur = cur->next;
            lenA++;
        }
        cur = headB;
        while (cur!=nullptr)
        {
            cur = cur->next;
            lenB++;
        }

        if (lenA>=lenB)
        {
            for (int i = 0;i<lenA - lenB;i++)
            {
                nodeA = nodeA->next;
            }

        }
        else
        {
            for (int i = 0;i<lenB - lenA;i++)
            {
                nodeB = nodeB->next;
            }
        }

        while (nodeA!=nullptr && nodeB!=nullptr)
        {
            if (nodeA == nodeB)
            {
                return nodeA;
            }
```

```
            else
            {
                nodeA = nodeA->next;
                nodeB = nodeB->next;
            }

        }
        return NULL;
    }
};
```

142. 环形链表II

142. 环形链表 II

给定一个链表的头节点 `head`，返回链表开始入环的第一个节点。 *如果链表无环，则返回* `null`。

如果链表中有某个节点，可以通过连续跟踪 `next` 指针再次到达，则链表中存在环。 为了表示给定链表中的环，评测系统内部使用整数 `pos` 来表示链表尾连接到链表中的位置（**索引从 0 开始**）。如果 `pos` 是 `-1`，则在该链表中没有环。**注意：** `pos` **不作为参数进行传递**，仅仅是为了标识链表的实际情况。

不允许修改 链表。

```
class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        ListNode* slow = head;
        ListNode* fast = head;
        while(fast!=nullptr && fast->next!=nullptr)
        {
            slow = slow->next;
            fast = fast->next->next;
            if (slow == fast)
            {
                ListNode* node1= head;
                ListNode* node2 = slow;
                while (node1!=node2)
                {
                    node1 = node1->next;
                    node2 = node2->next;
                }
                return node1;
            }
        }
        return NULL;

    }
};
```

哈希表

242. 有效的字母异位词

242. 有效的字母异位词 - 力扣

给定两个字符串 `s` 和 `t` ，编写一个函数来判断 `t` 是否是 `s` 的字母异位词。

注意：若 `s` 和 `t` 中每个字符出现的次数都相同，则称 `s` 和 `t` 互为字母异位词。

```
class Solution {
public:
    bool isAnagram(string s, string t) {
        unordered_map<char,int> mp;
        for (char i : s)
        {
            mp[i]++;
        }
        for (char i : t)
        {
            mp[i]--;
        }
        for (auto i : mp)
        {
            if (i.second != 0)
            {
                return false;
            }
        }
        return true;

    }
};
```

349. 两个数组的交集

349. 两个数组的交集 - 力扣

给定两个数组 `nums1` 和 `nums2` ，返回它们的交集。输出结果中的每个元素一定是 **唯一** 的。我们可以 **不考虑输出结果的顺序**

```
class Solution {
public:
    vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {
        unordered_set<int> result_set;
        unordered_set<int> st(nums2.begin(),nums2.end());
        for (int num : nums1)
        {
            if (st.find(num)!=st.end())
            {
                result_set.insert(num);
            }
```

```
        }
        vector<int> result(result_set.begin(),result_set.end());
        return result;
    }
};
```

202. 快乐数

202. 快乐数 - 力扣

编写一个算法来判断一个数 `n` 是不是快乐数。

「快乐数」 定义为：

- 对于一个正整数，每一次将该数替换为它每个位置上的数字的平方和。
- 然后重复这个过程直到这个数变为 1，也可能是 **无限循环** 但始终变不到 1。
- 如果这个过程 **结果为** 1，那么这个数就是快乐数。

如果 `n` 是 *快乐数* 就返回 `true` ；不是，则返回 `false` 。

```
class Solution {
public:
    bool isHappy(int n) {
        int sum = 0;
        unordered_set<int> st;
        while (1) {
            sum = 0;
            while (n)
            {
                sum += (n % 10) * (n % 10);
                n /= 10;
            }

            if (sum == 1) {
                return true;
            }

            if (st.find(sum) != st.end()) {
                return false;
            }
            else
            {
                st.insert(sum);
            }
            n = sum;
        }
    }
};
```