LeetCode 3217. 从链表中移除在数组中存在的节点

Feijoa_Li

一、 题目描述

给定一个整数数组 `nums` 和一个链表的头节点 `head`。从链表中移除所有存在于 `nums` 中的节点后，返回修改后的链表的头节点。

二、 解题思路

本题要求从链表中删除所有在给定数组 `nums` 中出现的节点。由于链表的删除操作需要知道前驱节点，因此我们需要维护前驱指针。

核心思路：

- 预处理数组： 将数组 `nums` 排序，以便使用二分查找快速判断节点值是否在数组中
- 处理头节点： 如果头节点的值在数组中，需要不断移动头指针，直到找到第一个不在数组中的节点
- 遍历链表： 使用两个指针 `cur` 和 `nxt` 遍历链表，当 `nxt` 节点的值在数组中时，跳过该节点(也就是将 $cur->next$ 指向 $nxt->next$).

算法步骤：

1. 对数组 `nums` 进行排序
2. 定义二分查找函数判断值是否在数组中
3. 处理头节点，确保头节点不在删除列表中
4. 使用双指针遍历链表，删除需要移除的节点

三、 代码实现

```
sort(nums.begin(), nums.end());
auto ok = [&](int x) {
    auto it = lower_bound(nums.begin(), nums.end(), x);
    if (it != nums.end() && *it == x)
        return true;
    return false;
};

while (ok(head->val)) {
    head = head->next;
}

auto cur = head;
auto nxt = head->next;

while (nxt != nullptr) {
    if (ok(nxt->val)) {
        cur->next = nxt->next;
    } else {
        cur = cur->next;
    }
    nxt = cur->next;
}
```

四、 复杂度分析

- **时间复杂度：** $O(n \log n + m \log n)$，其中 n 是数组 `nums` 的长度，m 是链表的长度。排序需要 $O(n \log n)$，每个节点的二分查找需要 $O(\log n)$。
- **空间复杂度：** $O(1)$，除了排序使用的栈空间外，只使用了常数额外空间。