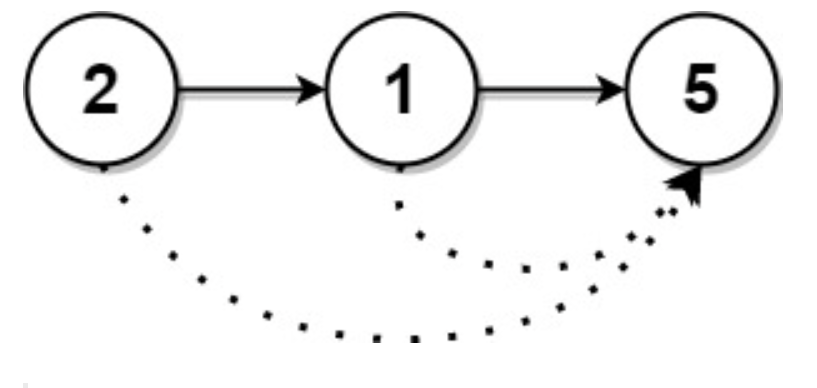
# 题目

给定一个长度为 n 的链表 head

对于列表中的每个节点，查找下一个 更大节点 的值。也就是说，对于每个节点，找到它旁边的第一个节点的值，这个节点的值 严格大于 它的值。

返回一个整数数组 answer ，其中 answer[i] 是第 i 个节点( 从1开始 )的下一个更大的节点的值。如果第 i 个节点没有下一个更大的节点，设置 answer[i] = 0 。

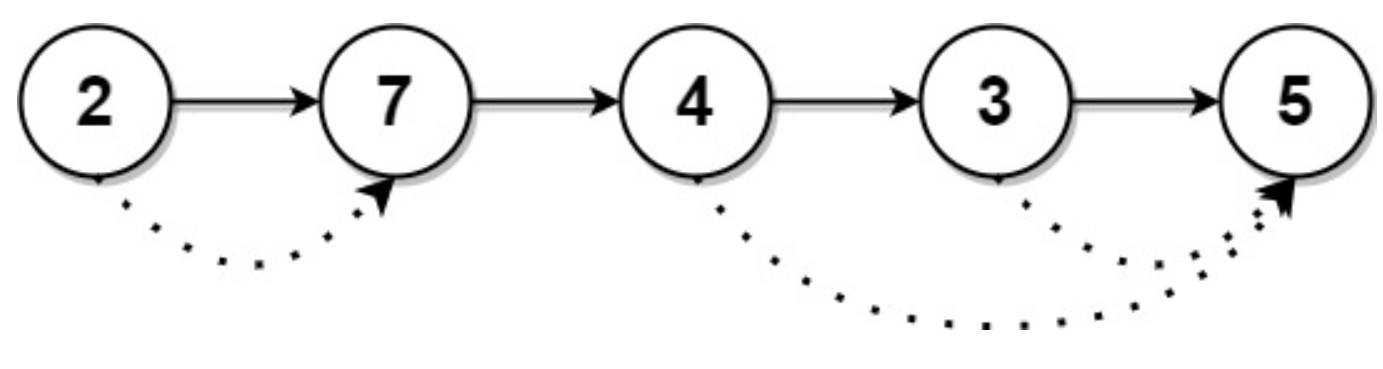
示例 1：



输入：head = [2,1,5]

输出：[5,5,0]

示例 2：



输入：head = [2,7,4,3,5]

输出：[7,0,5,5,0]

提示：

链表中节点数为 n

1 <= n <= 104

1 <= Node.val <= 109

# 分析

## 方法一：栈

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode() : val(0), next(nullptr) {}

\* ListNode(int x) : val(x), next(nullptr) {}

\* ListNode(int x, ListNode \*next) : val(x), next(next) {}

\* };

\*/

class Solution {

public:

// 反转链表

ListNode \*reverseList(ListNode \*head) {

ListNode \*pre = nullptr, \*cur = head;

while (cur) {

ListNode \*nxt = cur->next;

cur->next = pre;

pre = cur;

cur = nxt;

}

return pre;

}

vector<int> nextLargerNodes(ListNode\* head) {

head = reverseList(head);

vector<int> ans;

stack<int> st; // 单调栈（节点值）

for (auto cur = head; cur; cur = cur->next) {

while (!st.empty() && st.top() <= cur->val)

st.pop(); // 弹出无用数据

// 栈顶就是第 i 个节点的下一个更大元素

ans.push\_back(st.empty() ? 0 : st.top());

st.push(cur->val);

}

// 由于是倒着记录答案的，返回前要把答案反转

reverse(ans.begin(), ans.end());

return ans;

}

};