Assignment #6: 回溯、树、双向链表和哈希表

Updated 1526 GMT+8 Mar 22, 2025

2025 spring, Complied by 李振硕、信息管理系

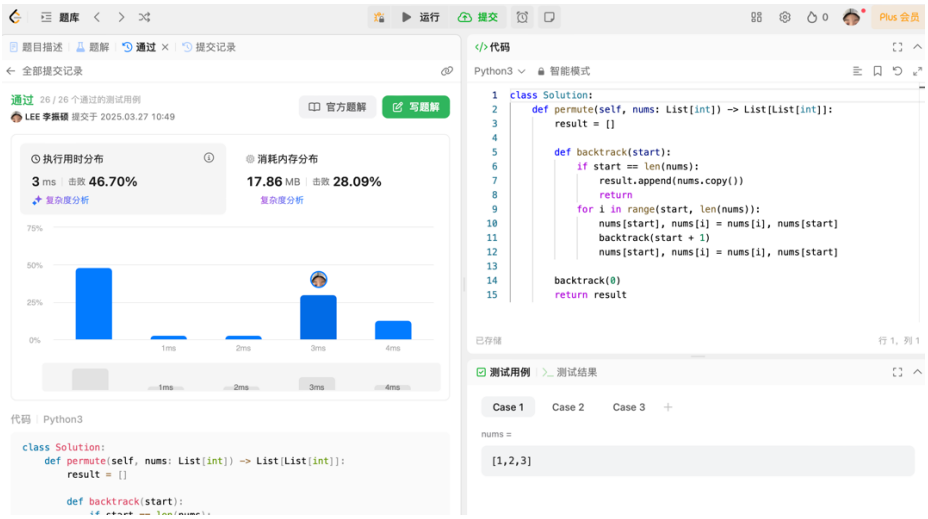
1. 题目

LC46.全排列

backtracking, https://leetcode.cn/problems/permutations/

思路:

代码:



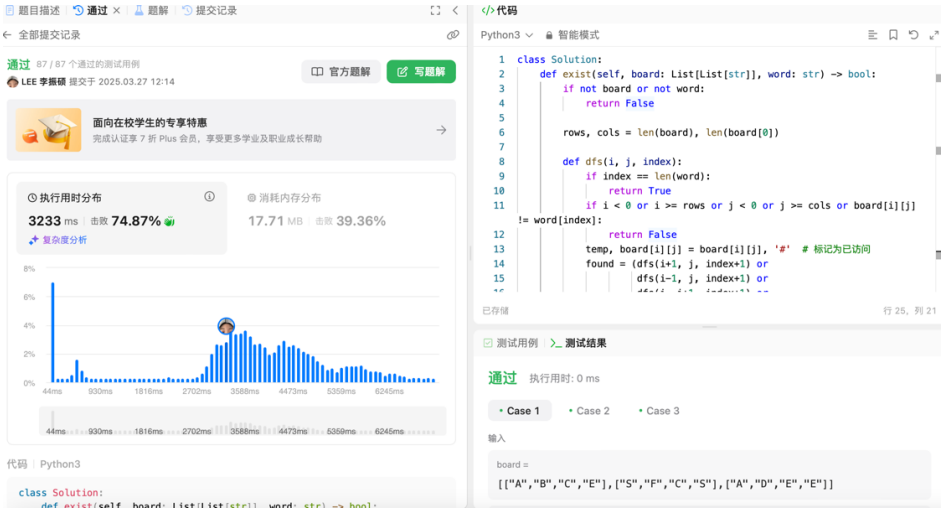
代码运行截图 <mark>（至少包含有"Accepted"）</mark>

LC79: 单词搜索

backtracking, https://leetcode.cn/problems/word-search/

思路:

代码:



LC94.二叉树的中序遍历

dfs, https://leetcode.cn/problems/binary-tree-inorder-traversal/

思路：

代码：

LC102.二叉树的层序遍历

bfs, https://leetcode.cn/problems/binary-tree-level-order-traversal/

思路：

代码：

LC131.分割回文串

dp, backtracking, https://leetcode.cn/problems/palindrome-partitioning/

思路：

代码:


```

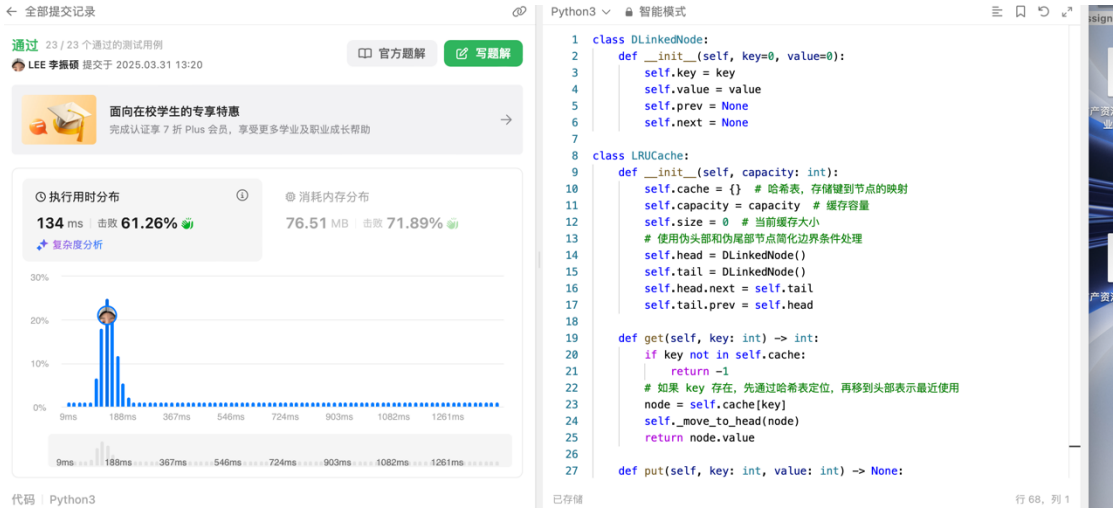
代码运行截图 <mark>（至少包含有"Accepted"）</mark>

### LC146.LRU 缓存

hash table, doubly-linked list, https://leetcode.cn/problems/lru-cache/

思路:

代码:


```

```python
def put(self, key: int, value: int) -> None:
    if key in self.cache:
        # 如果 key 存在，更新值并移到头部
        node = self.cache[key]
        node.value = value
        self._move_to_head(node)
    else:
        # 如果 key 不存在，创建新节点
        node = DLinkedNode(key, value)
        self.cache[key] = node
        self._add_to_head(node)
        self.size += 1
        # 如果超出容量，删除最久未使用的节点（尾部节点）
        if self.size > self.capacity:
            removed = self._remove_tail()
            del self.cache[removed.key]
            self.size -= 1

def _add_to_head(self, node):
    """将节点添加到双向链表头部"""
    node.prev = self.head
    node.next = self.head.next
    self.head.next.prev = node
    self.head.next = node

def _remove_node(self, node):
    """从双向链表中移除指定节点"""
    node.prev.next = node.next
    node.next.prev = node.prev


def _move_to_head(self, node):
    """将节点移到头部（先移除再添加到头部）"""
    self._remove_node(node)
    self._add_to_head(node)

def _remove_tail(self):
    """移除并返回双向链表的尾部节点"""
    node = self.tail.prev
    self._remove_node(node)
    return node
```

2. 学习总结和收获

这次作业中学习到了 backtrack 原理以及用法，还学到了二叉树的中序遍历、层序遍历。这些新的概念还是不太会，感觉需要复习。