Assignment #6: 回溯、树、双向链表和哈希表

Updated 1526 GMT+8 Mar 22, 2025

2025 spring, Complied by 颜鼎堃 工学院

> **说明：**
>
> 1. **解题与记录：**
>
>   对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge，Codeforces，LeetCode等平台上获得Accepted）。请将这些信息连同显示“Accepted”的截图一起填写到下方的作业模板中。（推荐使用Typora https://typoraio.cn 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。
>
> 2. **提交安排：**提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的“作业评论”区。确保你的Canvas账户有一个清晰可见的头像，提交的文件为PDF格式，并且“作业评论”区包含上传的.md或.doc附件。
>
> 3. **延迟提交：**如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。
>
> 请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

1. 题目

LC46.全排列

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

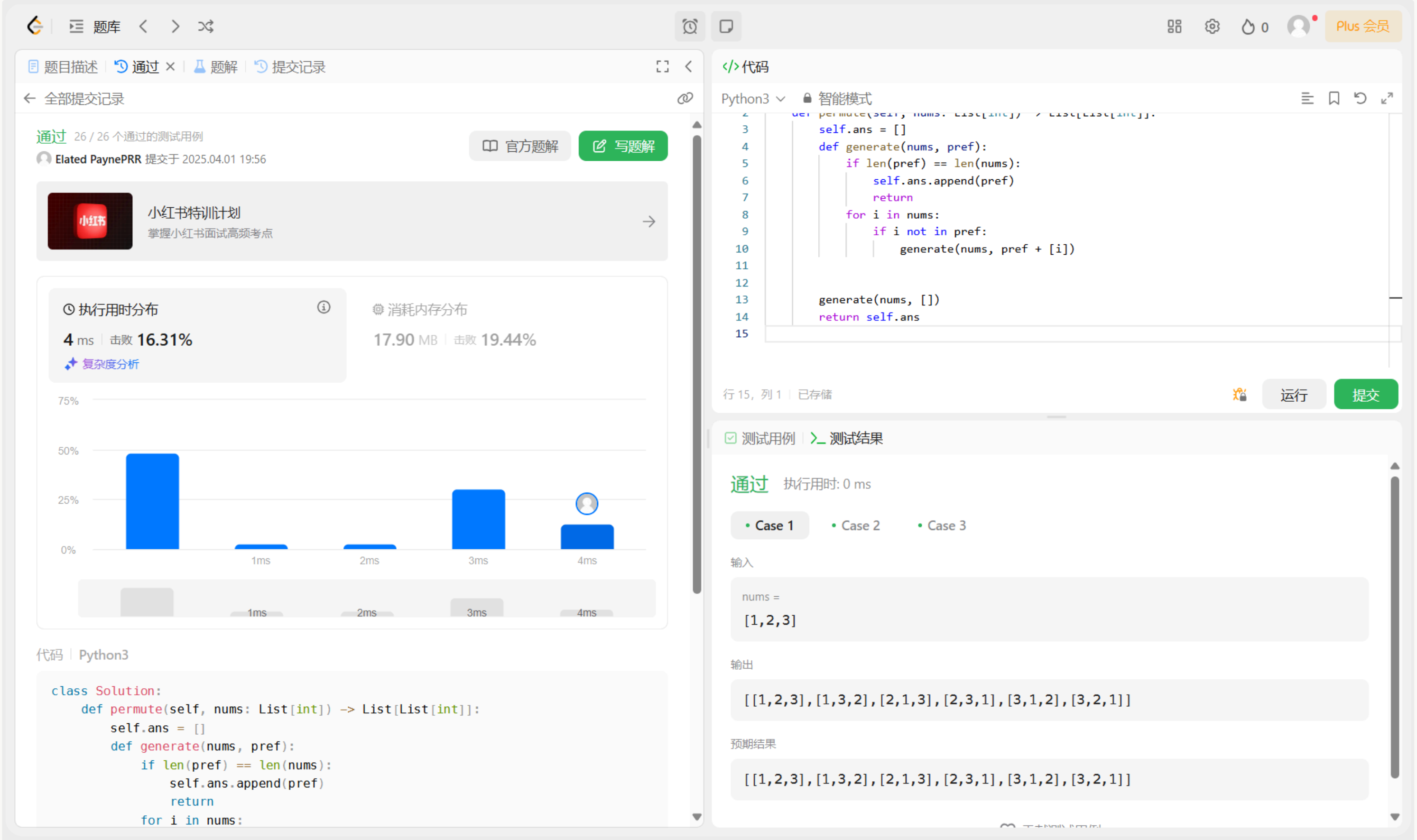
思路：

- 上学期在晴问写过

代码：

```python
from typing import *
class Solution:
    def permute(self, nums: List[int]) -> List[List[int]]:
        self.ans = []
        def generate(nums, pref):
            if len(pref) == len(nums):
                self.ans.append(pref)
                return
            for i in nums:
                if i not in pref:
                    generate(nums, pref + [i])


        generate(nums, [])
        return self.ans

if __name__ == '__main__':
    sol = Solution()
    print(*sol.permute(list(range(1, 6))), sep="\n")
```

代码运行截图 （至少包含有"Accepted"）



通过 26 / 26 个通过的测试用例

Elated PaynePRR 提交于 2025.04.01 19:56

执行用时分布

4 ms | 击败 16.31%

消耗内存分布

17.90 MB | 击败 19.44%

代码 | Python3

```python
class Solution:
    def permute(self, nums: List[int]) -> List[List[int]]:
        self.ans = []
        def generate(nums, pref):
            if len(pref) == len(nums):
                self.ans.append(pref)
                return
            for i in nums:
                if i not in pref:
                    generate(nums, pref + [i])


        generate(nums, [])
        return self.ans
```

行 15，列 1 | 已存储

通过 执行用时: 0 ms

输入

```
nums =
[1,2,3]
```

输出

```
[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

预期结果

```
[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

LC-46

LC79: 单词搜索

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

思路：

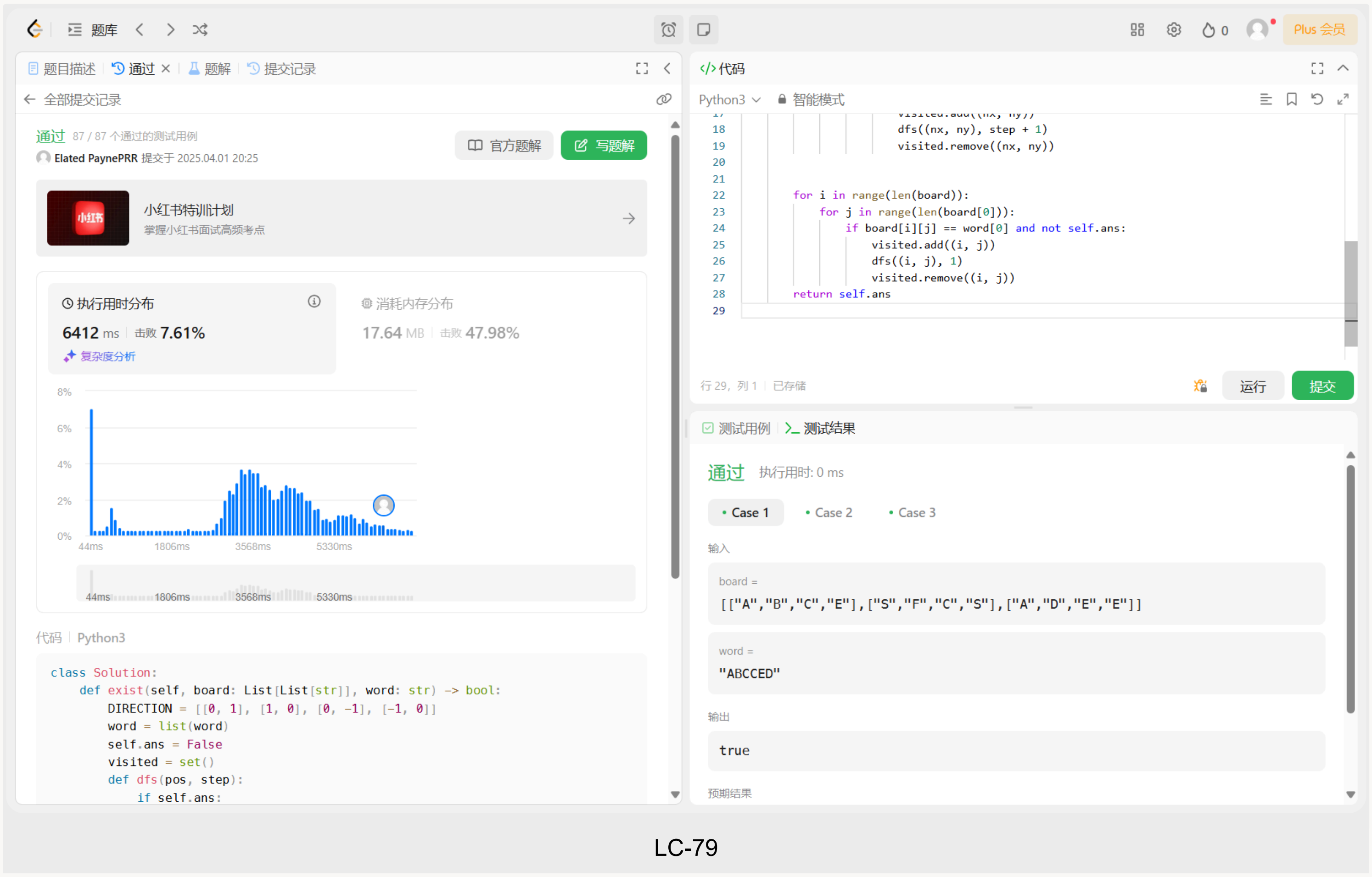
- dfs搜索

代码：

```python
from typing import *
class Solution:
    def exist(self, board: List[List[str]], word: str) -> bool:
        DIRECTION = [[0, 1], [1, 0], [0, -1], [-1, 0]]
        word = list(word)
        self.ans = False
        visited = set()
        def dfs(pos, step):
            if self.ans:
                return
            if step == len(word):
                self.ans = True
                return
            for dx, dy in DIRECTION:
                nx, ny = pos[0] + dx, pos[1] + dy
                if 0 <= nx < len(board) and 0 <= ny < len(board[0]) and (nx, ny) not in visited:
                    if board[nx][ny] == word[step]:
                        visited.add((nx, ny))
                        dfs((nx, ny), step + 1)
                        visited.remove((nx, ny))


        for i in range(len(board)):
            for j in range(len(board[0])):
                if board[i][j] == word[0] and not self.ans:
                    visited.add((i, j))
                    dfs((i, j), 1)
```

```python
                        visited.remove((i, j))
        return self.ans

if __name__ == '__main__':
    sol = Solution()
    print(sol.exist(board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "SEE"))
```

代码运行截图 （至少包含有"Accepted"）



通过 87 / 87 个通过的测试用例

Elated PaynePRR 提交于 2025.04.01 20:25

执行用时分布 6412 ms | 击败 7.61%

消耗内存分布 17.64 MB | 击败 47.98%

代码 | Python3

```python
class Solution:
    def exist(self, board: List[List[str]], word: str) -> bool:
        DIRECTION = [[0, 1], [1, 0], [0, -1], [-1, 0]]
        word = list(word)
        self.ans = False
        visited = set()
        def dfs(pos, step):
            if self.ans:
```

```python
                    dfs((nx, ny), step + 1)
                    visited.remove((nx, ny))


        for i in range(len(board)):
            for j in range(len(board[0])):
                if board[i][j] == word[0] and not self.ans:
                    visited.add((i, j))
                    dfs((i, j), 1)
                    visited.remove((i, j))
        return self.ans

```

通过 执行用时: 0 ms

输入

board =
[["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]]

word =
"ABCCED"

输出

true

LC-79

LC94.二叉树的中序遍历

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

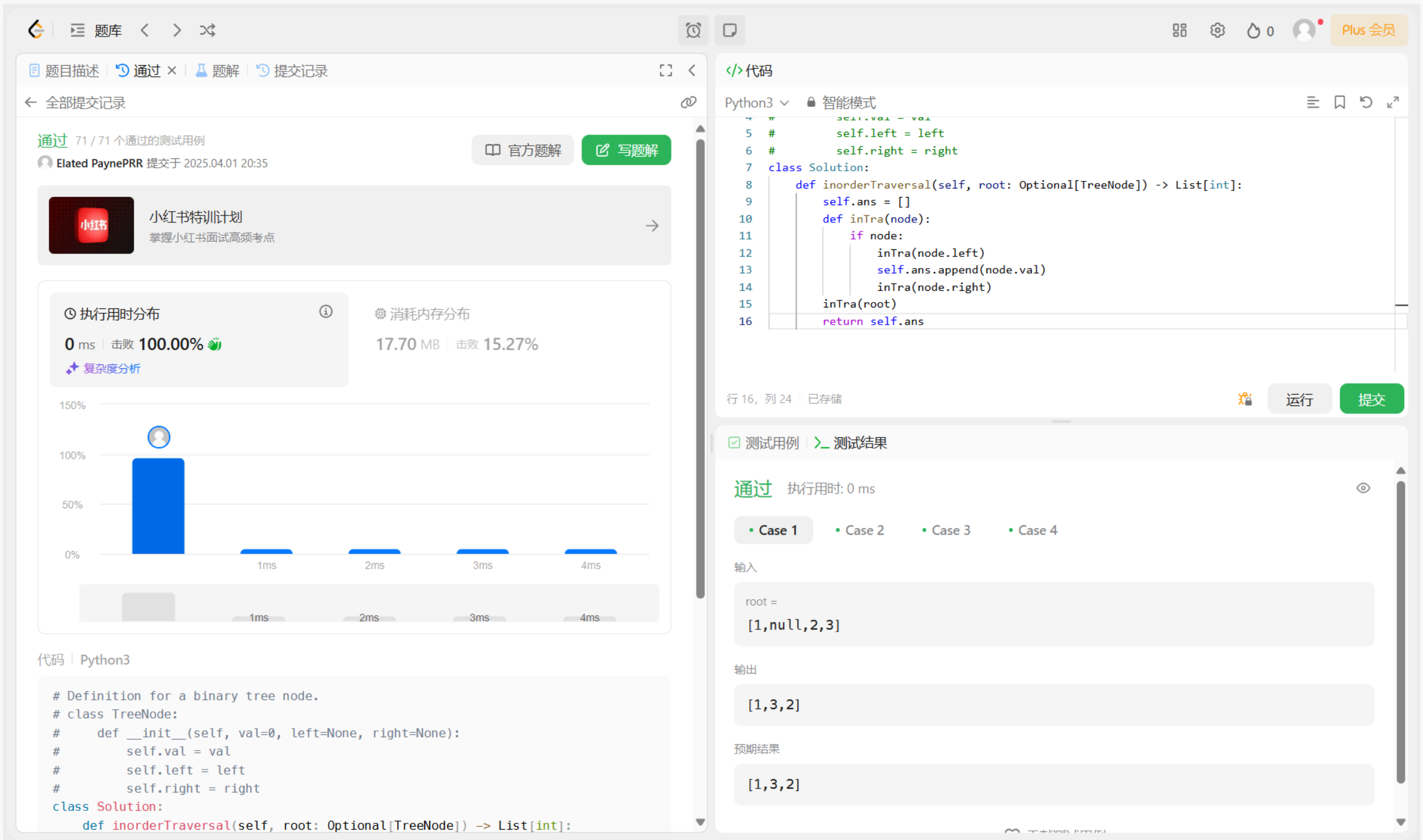
思路：

- 树上的dfs

代码：

```python
# Definition for a binary tree node.
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def inorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        self.ans = []
        def inTra(node):
            if node:
                inTra(node.left)
                self.ans.append(node.val)
                inTra(node.right)
        inTra(root)
        return self.ans

```

代码运行截图 （至少包含有"Accepted"）



题库

题目描述 | 通过 × | 题解 | 提交记录

全部提交记录

通过 71 / 71 个通过的测试用例

Elated PaynePRR 提交于 2025.04.01 20:35

官方题解 写题解

小红书特训计划

掌握小红书面试高频考点

执行用时分布

0 ms | 击败 100.00%

复杂度分析

消耗内存分布

17.70 MB | 击败 15.27%

代码 | Python3

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def inorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
```

代码

Python3 智能模式

```python
#         self.left = left
#         self.right = right
class Solution:
    def inorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        self.ans = []
        def inTra(node):
            if node:
                inTra(node.left)
                self.ans.append(node.val)
                inTra(node.right)
        inTra(root)
        return self.ans
```

行 16，列 24 | 已存储

运行 提交

测试用例 | 测试结果

通过 执行用时: 0 ms

Case 1 Case 2 Case 3 Case 4

输入

root =

```
[1,null,2,3]
```

输出

```
[1,3,2]
```

预期结果

```
[1,3,2]
```

LC-94

LC102.二叉树的层序遍历

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

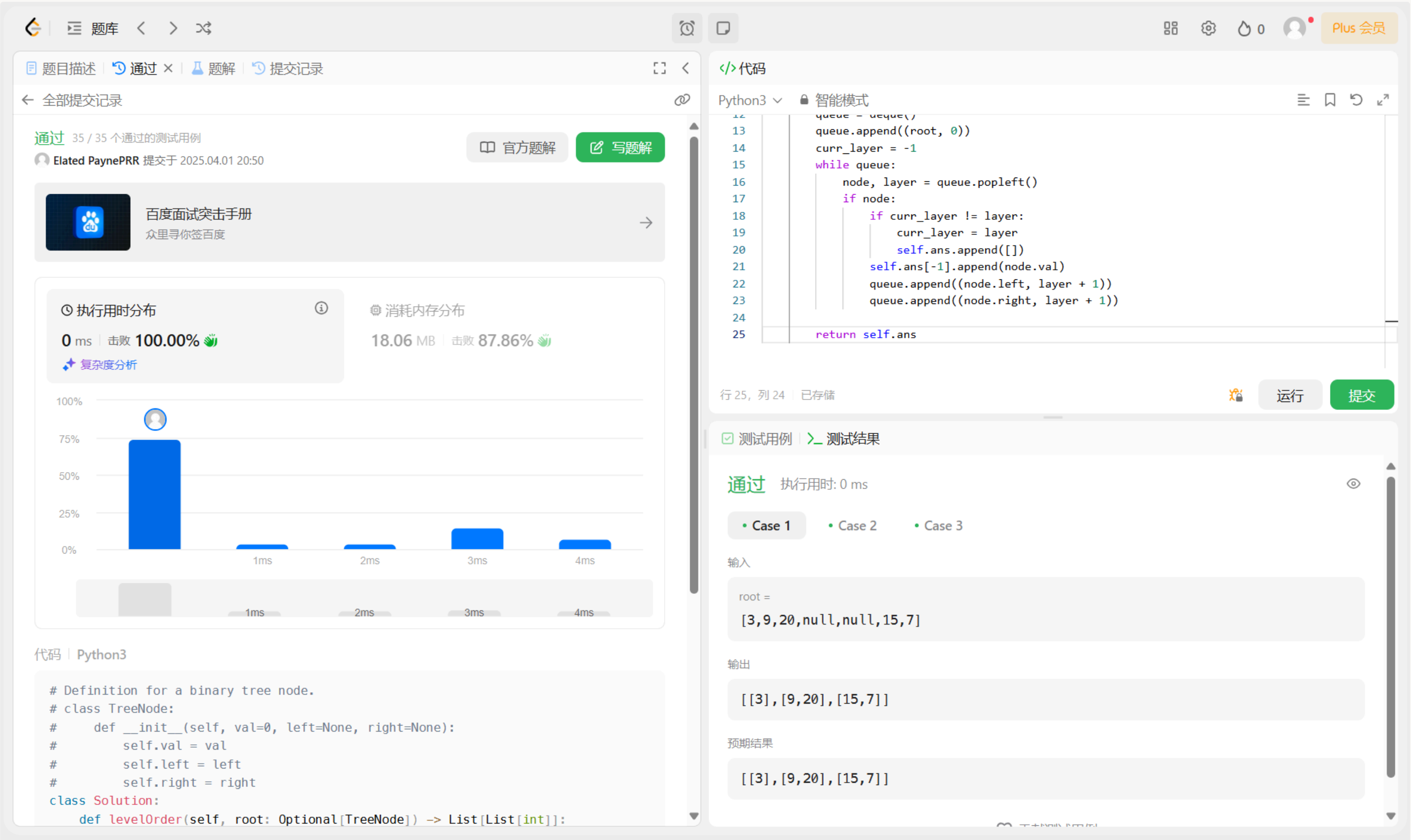
思路：

- 树上的bfs
- 记录layer决定要不要开数组

代码：

```python
# Definition for a binary tree node.
from collections import deque
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
class Solution:
    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        if not root:
            return []
        self.ans = []
        queue = deque()
        queue.append((root, 0))
        curr_layer = -1
        while queue:
            node, layer = queue.popleft()
            if node:
                if curr_layer != layer:
                    curr_layer = layer
                    self.ans.append([])
                self.ans[-1].append(node.val)
                queue.append((node.left, layer + 1))
                queue.append((node.right, layer + 1))
        return self.ans
```

代码运行截图（至少包含有"Accepted"）



题库

题目描述 | 通过 | 题解 | 提交记录

全部提交记录

通过 35 / 35 个通过的测试用例

Elated PaynePRR 提交于 2025.04.01 20:50

官方题解 写题解

百度面试突击手册

众里寻你签百度

执行用时分布

0 ms | 击败 100.00%

复杂度分析

消耗内存分布

18.06 MB | 击败 87.86%

代码 | Python3

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
```

代码

Python3 智能模式

```
        queue = deque()
        queue.append((root, 0))
        curr_layer = -1
        while queue:
            node, layer = queue.popleft()
            if node:
                if curr_layer != layer:
                    curr_layer = layer
                    self.ans.append([])
                self.ans[-1].append(node.val)
                queue.append((node.left, layer + 1))
                queue.append((node.right, layer + 1))

        return self.ans
```

行 25，列 24 | 已存储

运行 提交

测试用例 | 测试结果

通过 执行用时: 0 ms

Case 1 Case 2 Case 3

输入

root =

```
[3,9,20,null,null,15,7]
```

输出

```
[[3],[9,20],[15,7]]
```

预期结果

```
[[3],[9,20],[15,7]]
```

LC-102

LC131.分割回文串

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

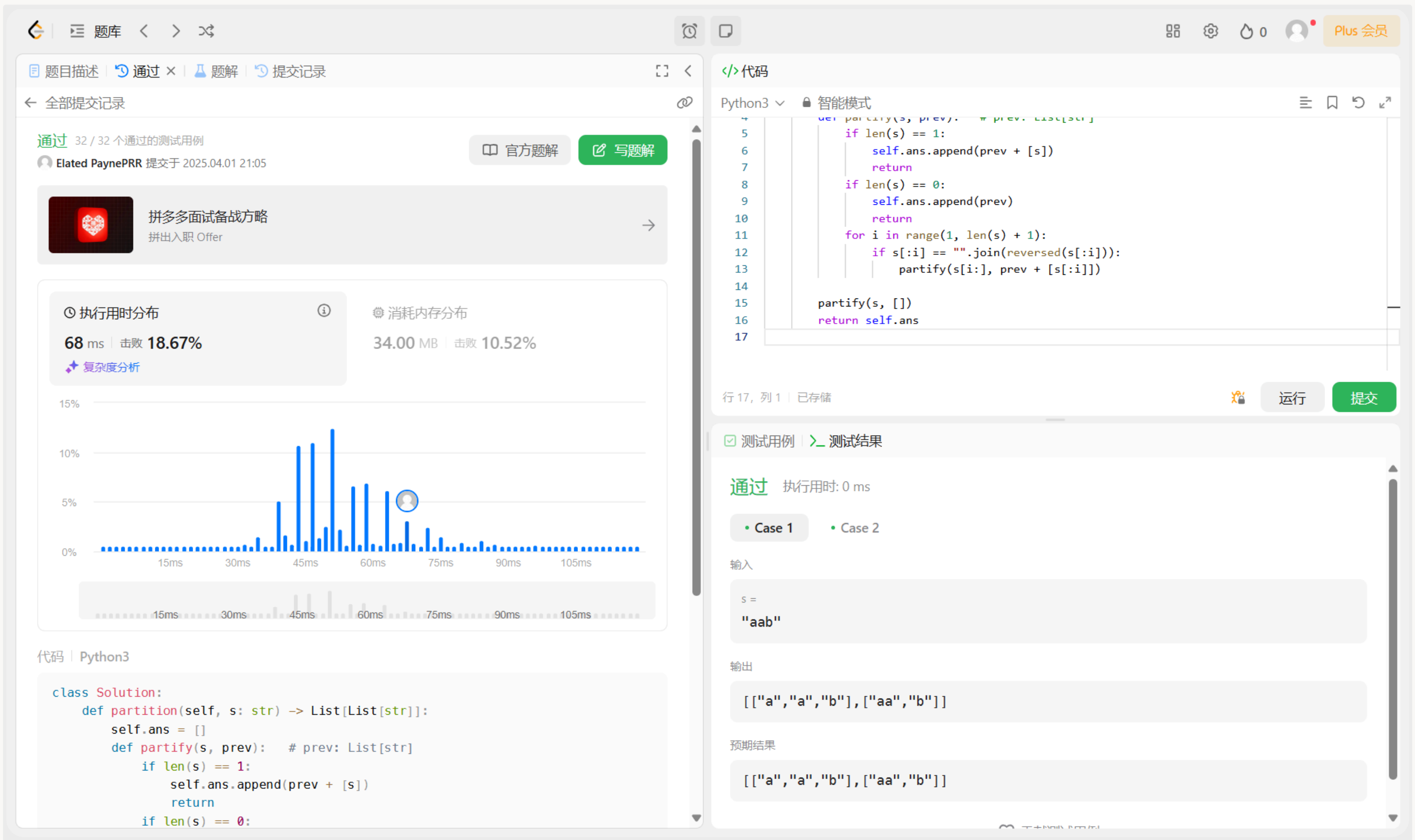
思路：

- 递归可做

代码：

```python
from typing import *
class Solution:
    def partition(self, s: str) -> List[List[str]]:
        self.ans = []
        def partify(s, prev):   # prev: List[str]
            if len(s) == 1:
                self.ans.append(prev + [s])
                return
            if len(s) == 0:
                self.ans.append(prev)
                return
            for i in range(1, len(s) + 1):
                if s[:i] == "".join(reversed(s[:i])):
                    partify(s[i:], prev + [s[:i]])

        partify(s, [])
        return self.ans

if __name__ == '__main__':
    sol = Solution()
    print(sol.partition("aababba"))
```

代码运行截图 （至少包含有"Accepted"）



通过 32 / 32 个通过的测试用例

Elated PaynePRR 提交于 2025.04.01 21:05

拼多多面试备战方略
拼出入职 Offer

执行用时分布
68 ms | 击败 18.67%

消耗内存分布
34.00 MB | 击败 10.52%

代码 | Python3

```python
class Solution:
    def partition(self, s: str) -> List[List[str]]:
        self.ans = []
        def partify(s, prev):    # prev: List[str]
            if len(s) == 1:
                self.ans.append(prev + [s])
                return
            if len(s) == 0:
```

Python3

```python
            if len(s) == 1:
                self.ans.append(prev + [s])
                return
            if len(s) == 0:
                self.ans.append(prev)
                return
            for i in range(1, len(s) + 1):
                if s[:i] == "".join(reversed(s[:i])):
                    partify(s[i:], prev + [s[:i]])

        partify(s, [])
        return self.ans
```

行 17，列 1 | 已存储

通过 执行用时: 0 ms

输入

s =
"aab"

输出

```
[["a","a","b"],["aa","b"]]
```

预期结果

```
[["a","a","b"],["aa","b"]]
```

LC-131

LC146.LRU缓存

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

思路：

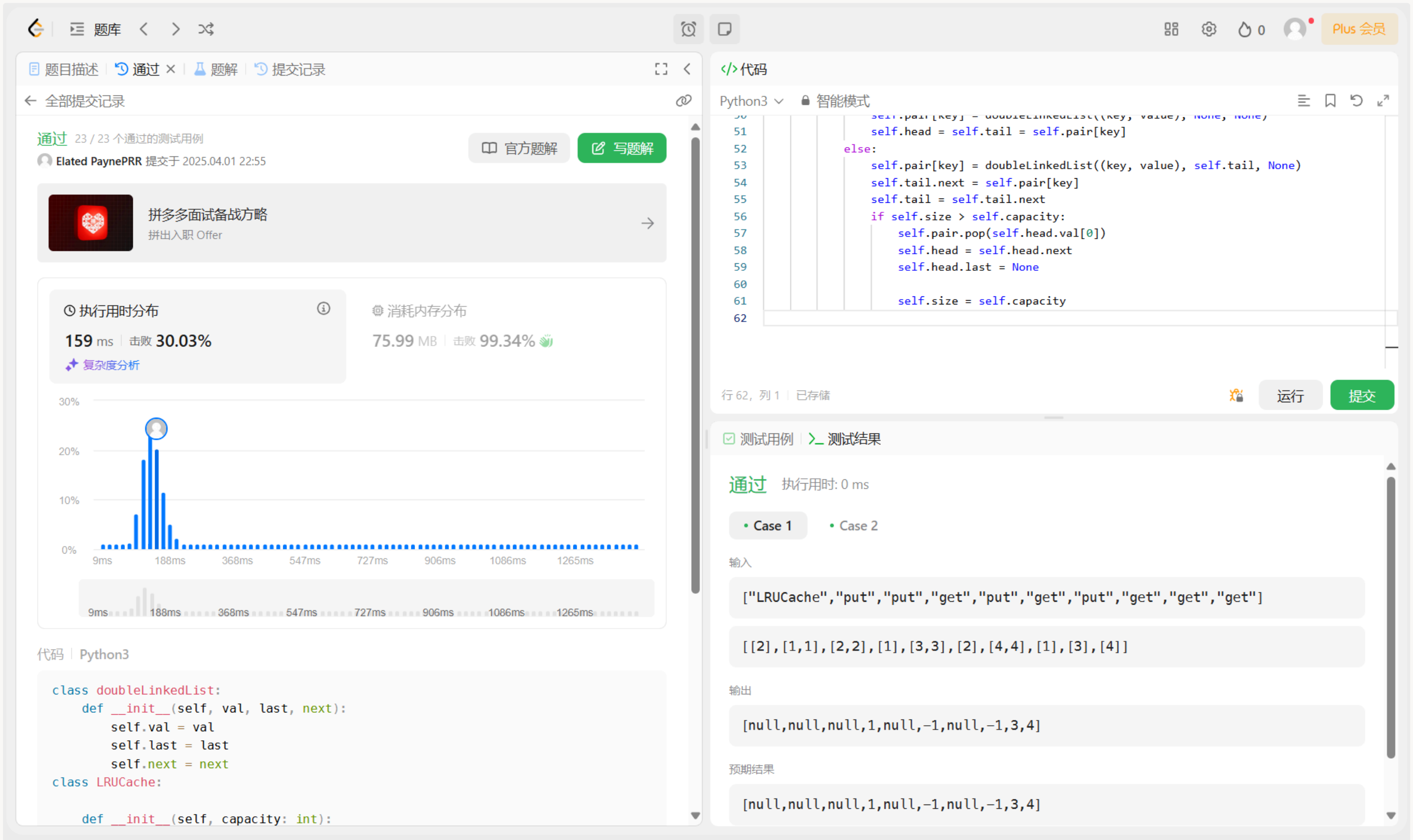
- 挺麻烦的
- 把双向链表的节点作为字典的值，在双向链表中记录
- 改错改了好久，总会忘记一些东西，然后根据结果慢慢找原因再修改
- 还是喜欢本地调试，起码快一些

代码：

```Python
class doubleLinkedList:
    def __init__(self, val, last, next):
        self.val = val
        self.last = last
        self.next = next
class LRUCache:

    def __init__(self, capacity: int):
        self.capacity = capacity
        self.pair = {}
        self.head = None
        self.tail = None
        self.size = 0

    def get(self, key: int) → int:
        if key in self.pair:
            if self.pair[key].next:
                if not self.pair[key].last:
                    self.head = self.head.next
                    self.head.last = None
                else:
```

```python
                    self.pair[key].last.next = self.pair[key].next
                    self.pair[key].next.last = self.pair[key].last
                self.pair[key].last = self.tail
                self.pair[key].next = None
                self.tail.next = self.pair[key]
                self.tail = self.tail.next
            return self.pair[key].val[1]
        else:
            return -1

    def put(self, key: int, value: int) -> None:
        if key in self.pair:
            self.pair[key].val = (key, value)
            if self.pair[key].next:
                if not self.pair[key].last:
                    self.head = self.head.next
                    self.head.last = None
                else:
                    self.pair[key].last.next = self.pair[key].next
                    self.pair[key].next.last = self.pair[key].last
                self.pair[key].last = self.tail
                self.pair[key].next = None
                self.tail.next = self.pair[key]
                self.tail = self.tail.next
        else:
            self.size += 1
            if self.size == 1:

                self.pair[key] = doubleLinkedList((key, value), None, None)
                self.head = self.tail = self.pair[key]
            else:
                self.pair[key] = doubleLinkedList((key, value), self.tail, None)
                self.tail.next = self.pair[key]
                self.tail = self.tail.next
                if self.size > self.capacity:
                    self.pair.pop(self.head.val[0])
                    self.head = self.head.next
                    self.head.last = None

                    self.size = self.capacity


if __name__ == '__main__':
    input = [["LRUCache","put","put","put","put","put","get","put","get","get","put","get","put","put","put","get","put","get","get","get","get","put","put","get","get","get","put","put","get","put","get","put","get","get","get","put","put","put","get","put","get","get","put","put","get","put","put","put","put","get","put","put","get","put","put","get","put","put","put","put","put","get","put","put","get","put","get","get","get","put","get","get","put","put","put","put","get","put","put","put","put","get","get","get","put","put","put","get","put","put","put","get","put","put","put","get","get","get","put","put","put","put","get","put","put","put","put","put","put","put"], [[10],[10,13],[3,17],[6,11],[10,5],[9,10],[13],[2,19],[2],[3],[5,25],[8],[9,22],[5,5],[1,30],[11],[9,12],[7],[5],[8],[9],[4,30],[9,3],[9],[10],[10],[6,14],[3,1],[3],[10,11],[8],[2,14],[1],[5],[4],[11,4],[12,24],[5,18],[13],[7,23],[8],[12],[3,27],[2,12],[5],[2,9],[13,4],[8,18],[1,7],[6],[9,29],[8,21],[5],[6,30],[1,12],[10],[4,15],[7,22],[11,26],[8,17],[9,29],[5],[3,4],[11,30],[12],[4,29],[3],[9],[6],[3,4],[1],[10],[3,29],[10,28],[1,20],[11,13],[3],[3,12],[3,8],[10,9],[3,26],[8],[7],[5],[13,17],[2,27],[11,15],[12],[9,19],[2,15],[3,16],[1],[12,17],[9,1],[6,19],[4],[5],[5],[8,1],[11,7],[5,2],[9,28],[1],[2,2],[7,4],[4,22],[7,24],[9,26],[13,28],[11,26]]]
    lru = None
    for i in range(len(input[0])):
        if input[0][i] == "LRUCache":
            lru = LRUCache(input[1][0][0])
        elif input[0][i] == "put":
            print(f"put({input[1][i][0]}, {input[1][i][1]})")
            lru.put(input[1][i][0], input[1][i][1])
        elif input[0][i] == "get":
            print(f"get({input[1][i][0]})")
            print(lru.get(input[1][i][0]))

```

代码运行截图 （至少包含有"Accepted"）



通过 23 / 23 个通过的测试用例

Elated PaynePRR 提交于 2025.04.01 22:55

执行用时分布 159 ms | 击败 30.03%

消耗内存分布 75.99 MB | 击败 99.34%

代码 | Python3

```python
class doubleLinkedList:
    def __init__(self, val, last, next):
        self.val = val
        self.last = last
        self.next = next
class LRUCache:

    def __init__(self, capacity: int):
```

```python
                self.head = self.tail = self.pair[key]
            else:
                self.pair[key] = doubleLinkedList((key, value), self.tail, None)
                self.tail.next = self.pair[key]
                self.tail = self.tail.next
                if self.size > self.capacity:
                    self.pair.pop(self.head.val[0])
                    self.head = self.head.next
                    self.head.last = None

                    self.size = self.capacity

```

测试结果

通过 执行用时: 0 ms

Case 1 Case 2

输入

```
["LRUCache","put","put","get","put","get","put","get","get","get"]
[[2],[1,1],[2,2],[1],[3,3],[2],[4,4],[1],[3],[4]]
```

输出

```
[null,null,null,1,null,-1,null,-1,3,4]
```

预期结果

```
[null,null,null,1,null,-1,null,-1,3,4]
```

LC-146

2. 学习总结和收获

如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算2025spring每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。

为了让自己抽出时间补一补每日选做决定先做每日选做再做作业

于是我以为作业星期三交，在星期二的晚上开始补，幸好这次作业相对而言比较简单，就最后一题麻烦点