# Assignment #6: 回溯、树、双向链表和哈希表

Updated 1526 GMT+8 Mar 22, 2025

2025 spring, Complied by <mark>同学的姓名、院系</mark>

**姓名：李彦臻**
**学号：2300010821**
**学院：数学科学学院**

> **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/

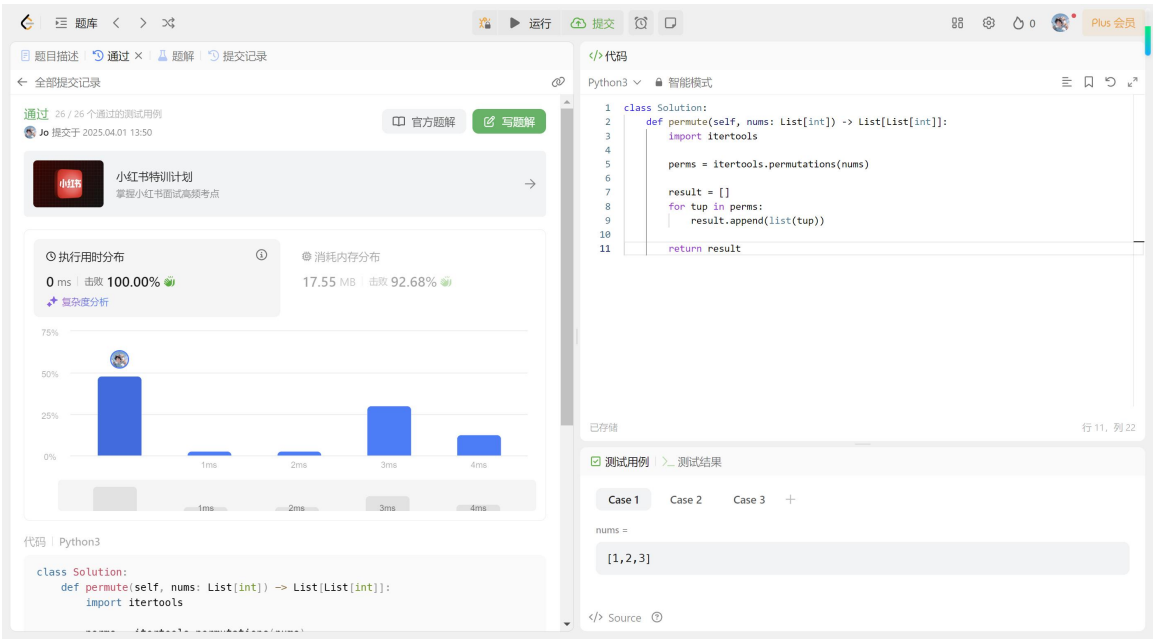
思路：使用 permutations 函数即可；需要注意的是，permutations 函数返回的是一个迭代器，不能直接 return 或者 print！

用时：5mins

代码：

```python

```
```
class Solution:
    def permute(self, nums: List[int]) -> List[List[int]]:
        import itertools

        perms = itertools.permutations(nums)

        result = []
        for tup in perms:
            result.append(list(tup))

        return result
```

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/

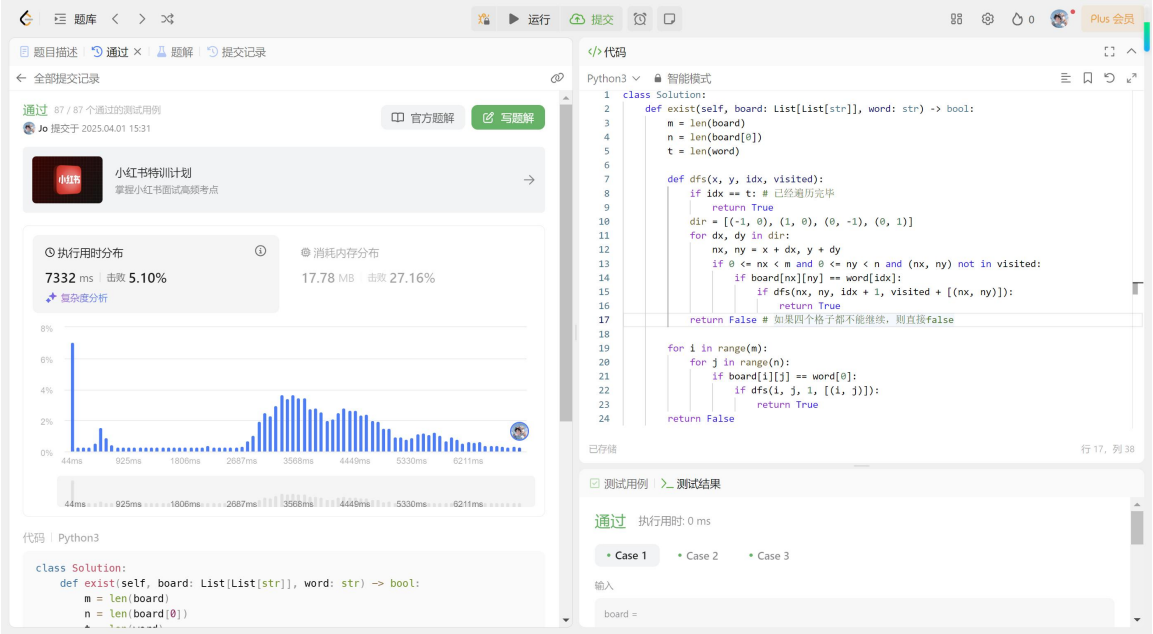
思路：使用常规的 dfs 思路来做就行：对每一个与第一个字母相同的格子，都去寻找它周围是否有与第二个字母相同的格子；接着，在第二个字母周围找是否有与第三个字母相同的格子；第三个也同理。如此循环找下去，便可找到最终符合题意的路径。

时间：20mins

代码：

```python
```
```
class Solution:
    def exist(self, board: List[List[str]], word: str) -> bool:
        m = len(board)
        n = len(board[0])
        t = len(word)

        def dfs(x, y, idx, visited):
            if idx == t: # 已经遍历完毕
                return True
            dir = [(-1, 0), (1, 0), (0, -1), (0, 1)]
            for dx, dy in dir:
                nx, ny = x + dx, y + dy
                if 0 <= nx < m and 0 <= ny < n and (nx, ny) not in visited:
                    if board[nx][ny] == word[idx]:
                        if dfs(nx, ny, idx + 1, visited + [(nx, ny)]):
                            return True
            return False # 如果四个格子都不能继续，则直接 false

        for i in range(m):
            for j in range(n):
                if board[i][j] == word[0]:
                    if dfs(i, j, 1, [(i, j)]):
                        return True
        return False
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>




### LC94.二叉树的中序遍历

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

思路：先用 while 循环不断深入左子树，并将路径上的节点放进栈中，直到没有左子树为止；当无法继续向左时，我们从栈中 pop 出一个节点，将它的值加入 result 中；随后，访问完根节点后，进入其右子树，重复以上过程即可。

时间：20mins

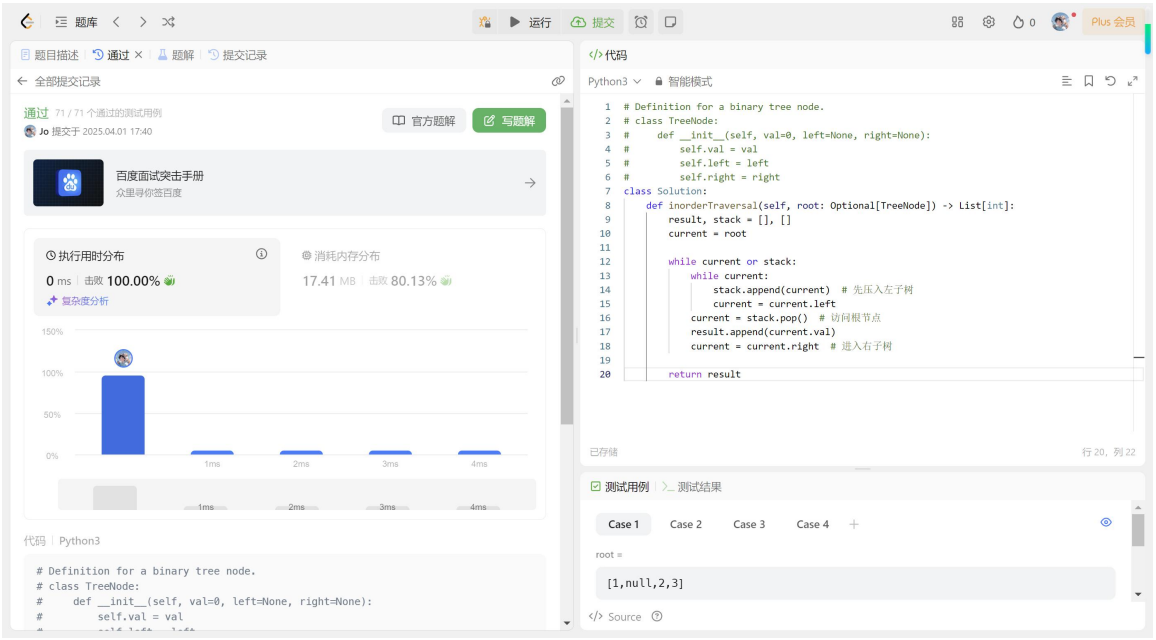
代码：

```python
```

```
class Solution:
    def inorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        result, stack = [], []
        current = root

        while current or stack:
            while current:
                stack.append(current)  # 先压入左子树
                current = current.left
            current = stack.pop()  # 访问根节点
```

```
            result.append(current.val)
            current = current.right  # 进入右子树

        return result
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>




### LC102.二叉树的层序遍历

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

思路：先将 root 加入队列，并依次处理队列中的每一层节点；取出当前层所有节点，并将它们的左右子节点加入队列，直到遍历完整棵树；
对于每一层可以创建一个子列表 level_nodes 来存储该层所有节点的值，并在遍历当前层的所有节点后，将 level_nodes 加入结果列表 results 中即可。

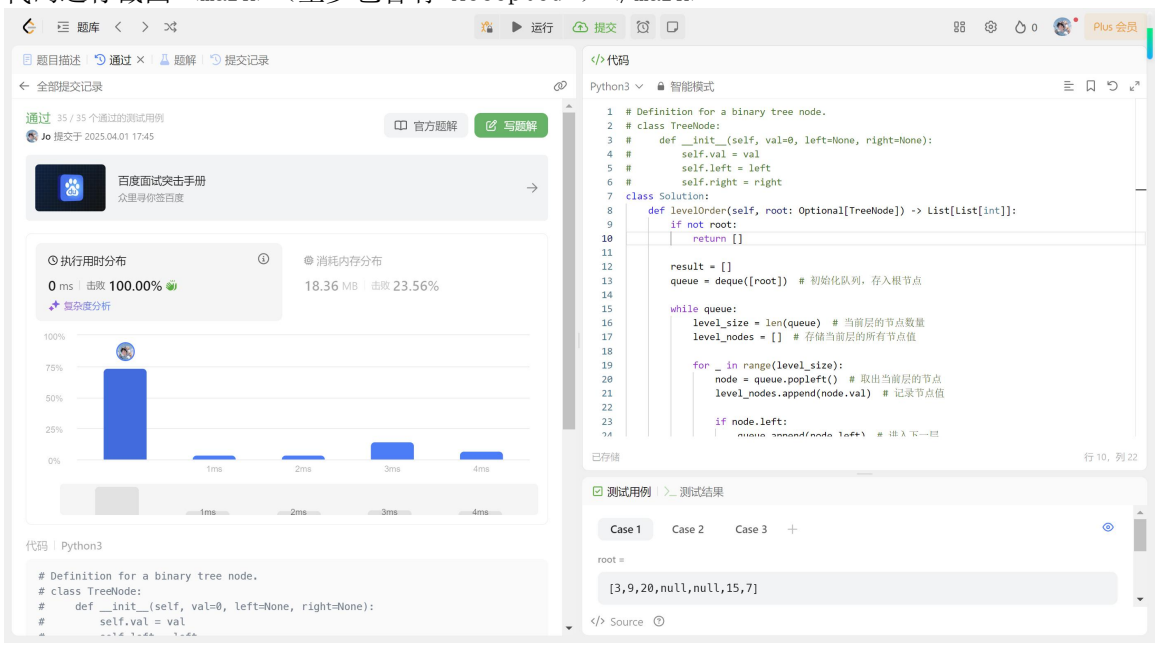
时间：25mins

代码：

```python

```

```python
class Solution:
    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        if not root:
            return []

        result = []
        queue = deque([root])  # 初始化队列，存入根节点

        while queue:
            level_size = len(queue)  # 当前层的节点数量
            level_nodes = []  # 存储当前层的所有节点值

            for _ in range(level_size):
                node = queue.popleft()  # 取出当前层的节点
                level_nodes.append(node.val)  # 记录节点值

                if node.left:
                    queue.append(node.left)  # 进入下一层
                if node.right:
                    queue.append(node.right)

            result.append(level_nodes)  # 存储当前层结果

        return result
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



### LC131.分割回文串

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

思路：

代码：

```python

```

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

```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

## 2. 学习总结和收获

<mark>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。</mark>

第一题就是对 permutations 函数的一个小应用，很简单，唯一需要注意的点是 permutations 函数返回的是一个迭代器，不能直接 print，也不能直接 return。

```
import itertools

nums = list(map(int,input().split()))
perms = itertools.permutations(nums)

print(perms)
```

（比如像这样就是错的，只会返回迭代器所在的位置，已经犯了两次一样的错了。。）

第二题就是一个方法很经典的 dfs 问题，感觉代码已经比较套路化了，没什么太多好说的；但是因为好久没怎么敲 dfs 的代码了，所以代码真正写起来的时候还是出了不少 bug，用时还是有点久；我这个代码的方法也不是特别好，几乎超时，题解的思路会更好一些。第三题只要能想到用 stack 的方法辅助来做就不难。第四题则是一个方法比较经典的 bfs 问题，用 deque 来辅助处理即可，代码也是比较模板化，算是比较常规的一道题。