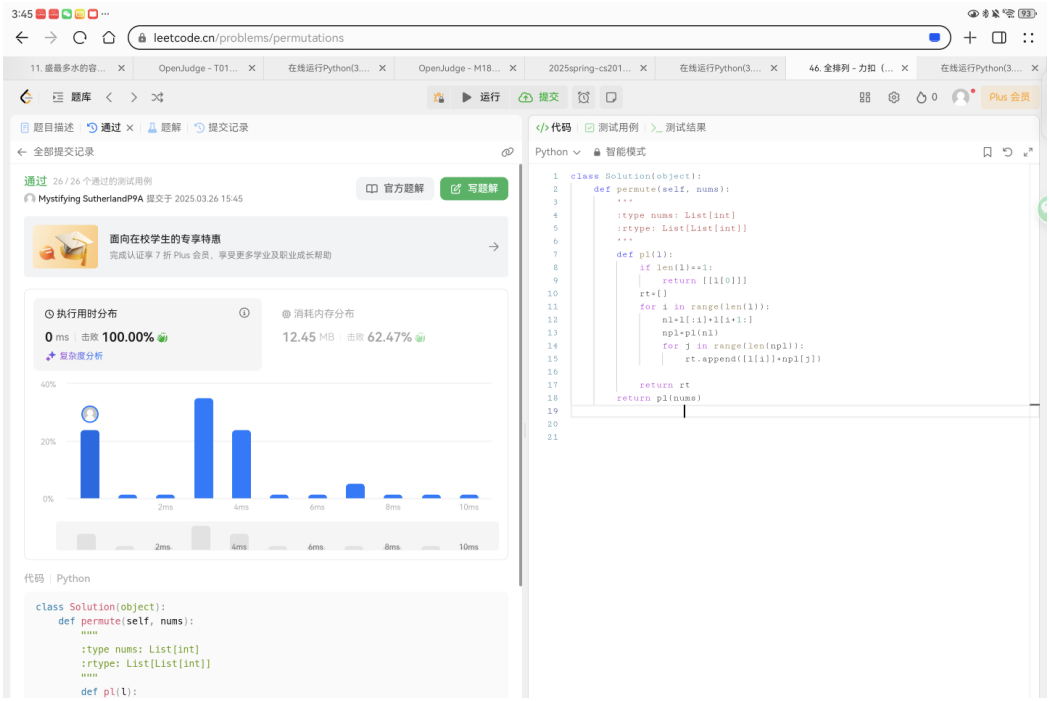
1. 题目
LC46.全排列
backtracking, https://leetcode.cn/problems/permutations/

思路:

代码:

```
class Solution(object):

    def permute(self, nums):

        """

        :type nums: List[int]

        :rtype: List[List[int]]

        """

        def pl(l):

            if len(l)==1:

                return [[l[0]]]

            rt=[]

            for i in range(len(l)):

                nl=l[:i]+l[i+1:]

                npl=pl(nl)

                for j in range(len(npl)):

                    rt.append([l[i]]+npl[j])



            return rt

        return pl(nums)
```

代码运行截图 （至少包含有"Accepted"）



LC79: 单词搜索

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

思路:

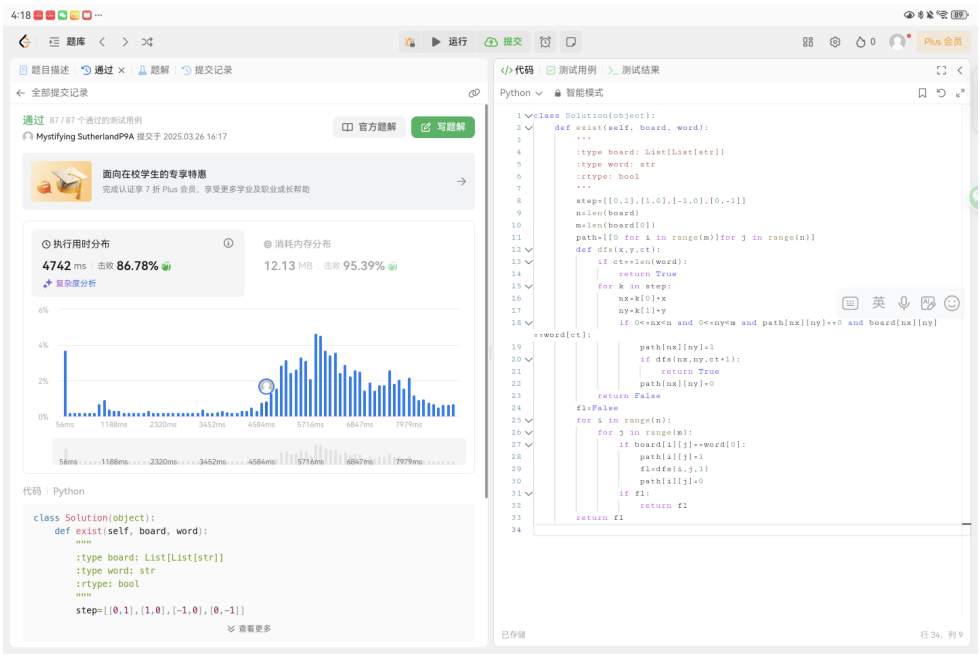
代码:

```
class Solution(object):

    def exist(self, board, word):

        """

        :type board: List[List[str]]

        :type word: str

        :rtype: bool

        """

        step=[[0,1],[1,0],[-1,0],[0,-1]]

        n=len(board)

        m=len(board[0])
```

```
        path=[[0 for i in range(m)]for j in range(n)]
        def dfs(x,y,ct):
            if ct==len(word):
                return True
            for k in step:
                nx=k[0]+x
                ny=k[1]+y
                if 0<=nx<n and 0<=ny<m and path[nx][ny]==0 and board[nx][ny]
==word[ct]:
                    path[nx][ny]=1
                    if dfs(nx,ny,ct+1):
                        return True
                    path[nx][ny]=0
            return False
        fl=False
        for i in range(n):
            for j in range(m):
                if board[i][j]==word[0]:
                    path[i][j]=1
                    fl=dfs(i,j,1)
                    path[i][j]=0
                if fl:
                    return fl
```

return fl

代码运行截图 （至少包含有"Accepted"）



LC94.二叉树的中序遍历

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

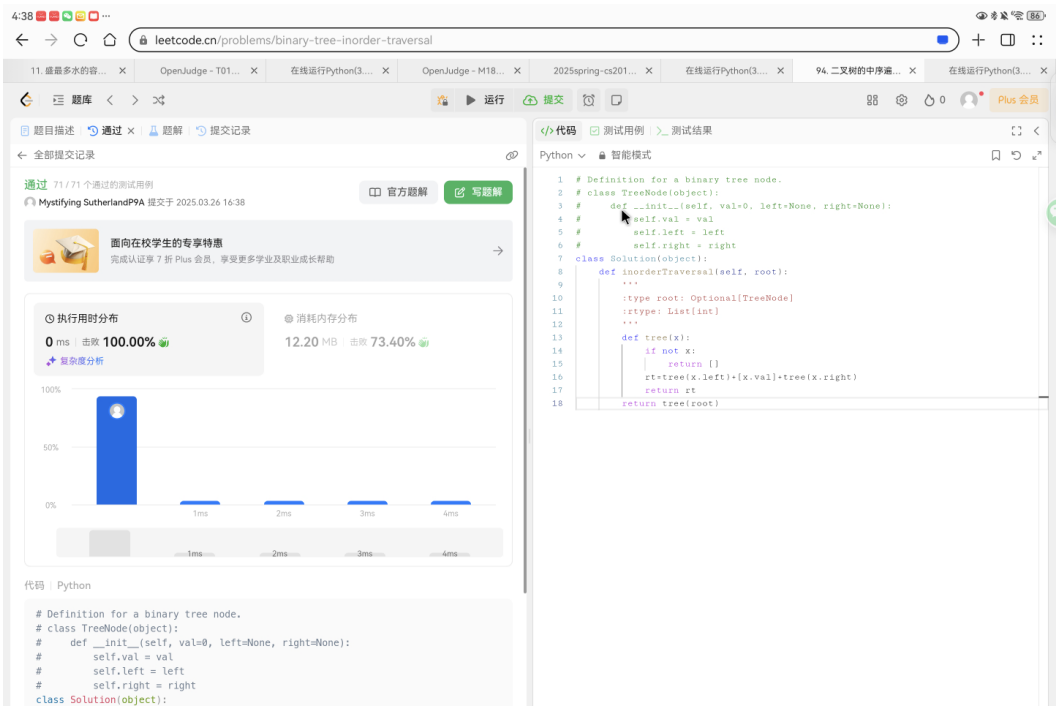
思路:

代码:

```
# Definition for a binary tree node.

# class TreeNode(object):

#     def __init__(self, val=0, left=None, right=None):

#         self.val = val

#         self.left = left

#         self.right = right

class Solution(object):

    def inorderTraversal(self, root):

        """

        :type root: Optional[TreeNode]
```

```
        :rtype: List[int]
        """
        def tree(x):
            if not x:
                return []
            rt=tree(x.left)+[x.val]+tree(x.right)
            return rt
        return tree(root)
```

代码运行截图 （至少包含有"Accepted"）



LC102.二叉树的层序遍历

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

思路：

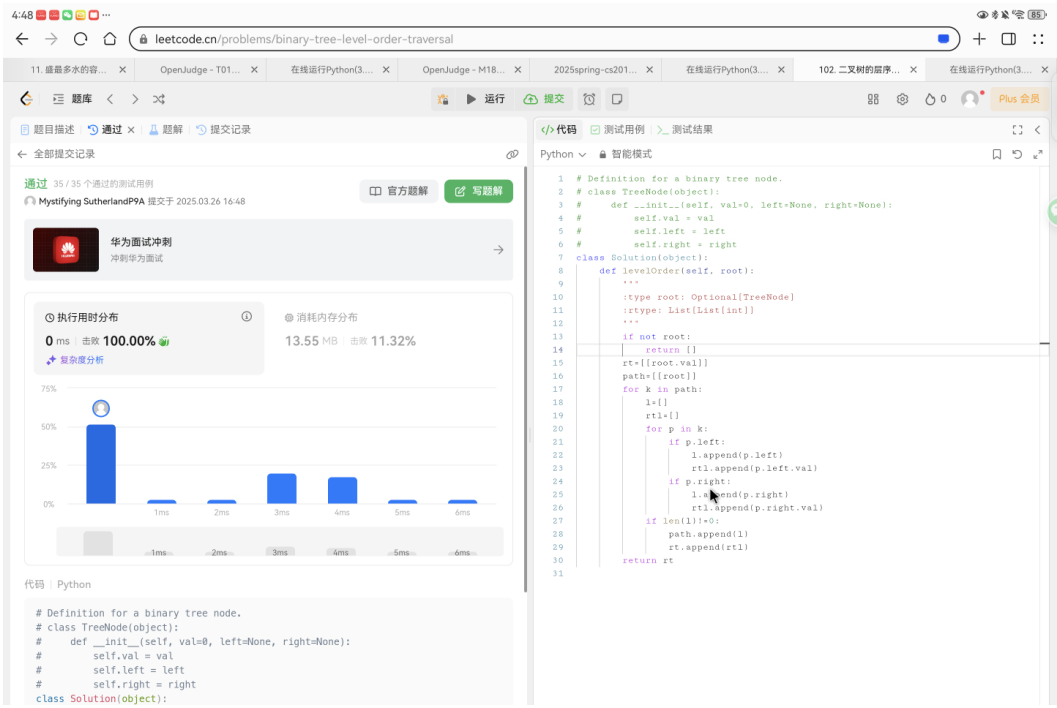
代码：

```
# Definition for a binary tree node.

# class TreeNode(object):
```

```
#         def __init__(self, val=0, left=None, right=None):
#             self.val = val
#             self.left = left
#             self.right = right
class Solution(object):
    def levelOrder(self, root):
        """
        :type root: Optional[TreeNode]
        :rtype: List[List[int]]
        """
        if not root:
            return []
        rt=[[root.val]]
        path=[[root]]
        for k in path:
            l=[]
            rtl=[]
            for p in k:
                if p.left:
                    l.append(p.left)
                    rtl.append(p.left.val)
                if p.right:
```

```
                l.append(p.right)
                rtl.append(p.right.val)
            if len(l)!=0:
                path.append(l)
                rt.append(rtl)
        return rt
```

代码运行截图 （至少包含有"Accepted"）



LC131.分割回文串

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

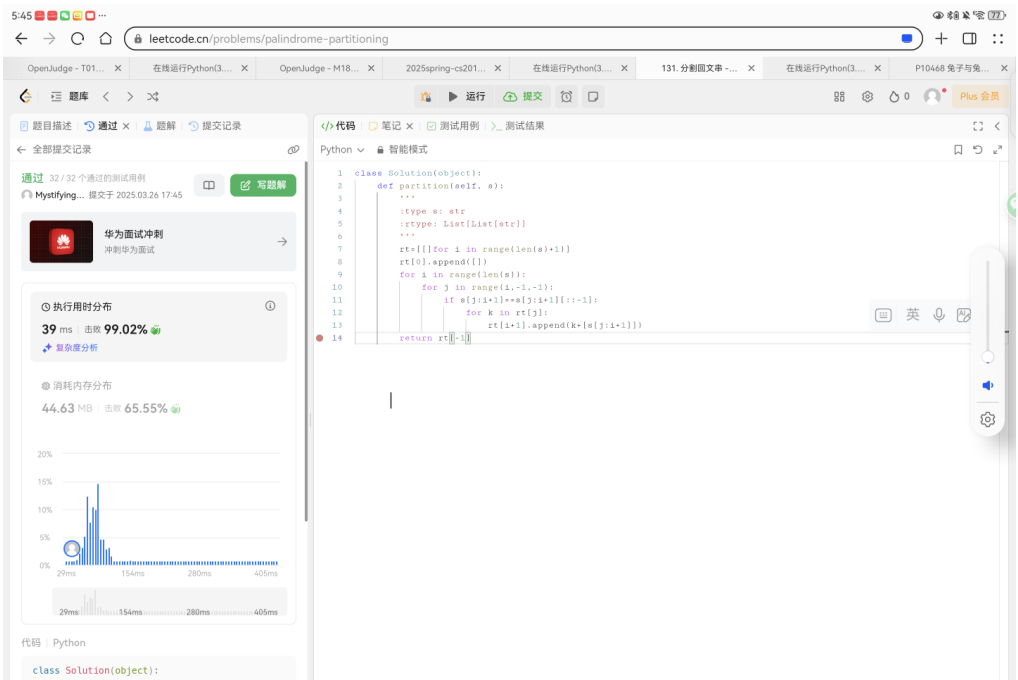
思路：

代码：

```
class Solution(object):
    def partition(self, s):
        """
        :type s: str
```

```
        :rtype: List[List[str]]

        """

        rt=[[]for i in range(len(s)+1)]

        rt[0].append([])

        for i in range(len(s)):

            for j in range(i,-1,-1):

                if s[j:i+1]==s[j:i+1][::-1]:

                    for k in rt[j]:

                        rt[i+1].append(k+[s[j:i+1]])

        return rt[-1]
```

代码运行截图 （至少包含有"Accepted"）



LC146.LRU 缓存

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

思路:

代码:

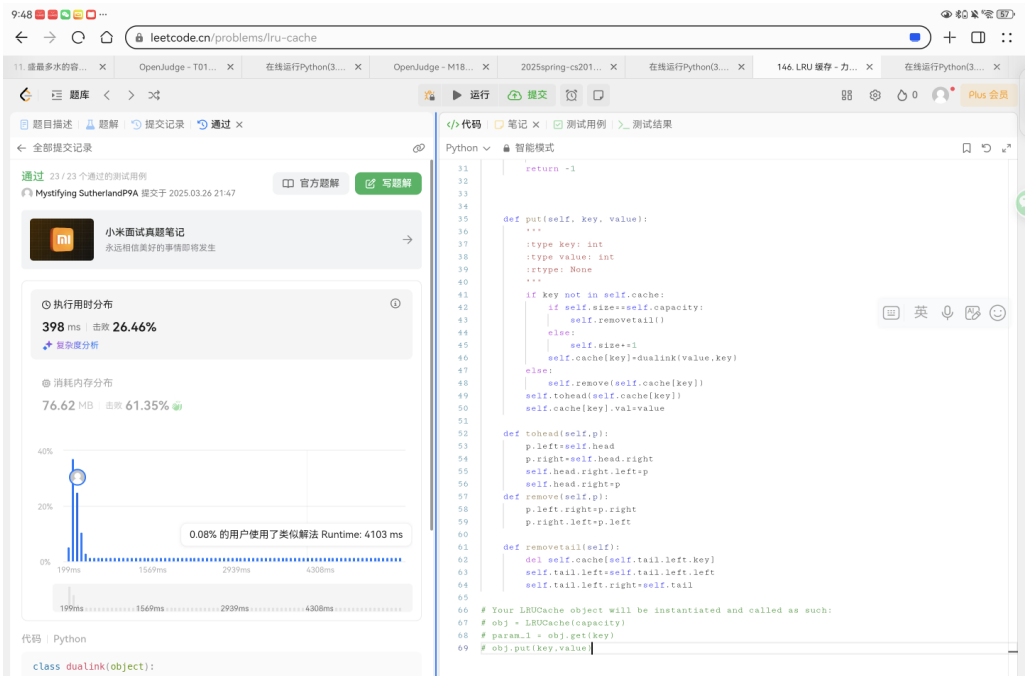
```
class dualink(object):
    def __init__(self,value,key):
        self.left=None
        self.right=None
        self.val=value
        self.key=key
class LRUCache(object):
    def __init__(self, capacity):
        """
        :type capacity: int
        """
        self.cache={}
        self.capacity=capacity
        self.head=dualink(0,0)
        self.tail=dualink(0,0)
        self.head.right=self.tail
        self.tail.left=self.head
        self.size=0




    def get(self, key):
```

```
        """
        :type key: int
        :rtype: int
        """
        if key in self.cache:
            self.remove(self.cache[key])
            self.tohead(self.cache[key])
            return self.cache[key].val
        return -1




def put(self, key, value):
        """
        :type key: int
        :type value: int
        :rtype: None
        """
        if key not in self.cache:
            if self.size==self.capacity:
                self.removetail()
            else:
```

```
                self.size+=1
            self.cache[key]=dualink(value,key)
        else:
            self.remove(self.cache[key])
        self.tohead(self.cache[key])
        self.cache[key].val=value

    def tohead(self,p):
        p.left=self.head
        p.right=self.head.right
        self.head.right.left=p
        self.head.right=p
    def remove(self,p):
        p.left.right=p.right
        p.right.left=p.left

    def removetail(self):
        del self.cache[self.tail.left.key]
        self.tail.left=self.tail.left.left
        self.tail.left.right=self.tail

# Your LRUCache object will be instantiated and called as such:
```

```
# obj = LRUCache(capacity)

# param_1 = obj.get(key)

# obj.put(key,value)
```

代码运行截图 （至少包含有"Accepted"）



2. 学习总结和收获

最后一题较为复杂，需要自己定义多个操作。但这么做会使得思路比较清晰，同时类的操作虽然代码长一点，但可以通过简单功能的叠加完成复杂操作，这应该是类的存在意义。

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。

日常跟进 OJ 每日选做