Assignment #6: 回溯、树、双向链表和哈希表

Updated 1526 GMT+8 Mar 22, 2025

2025 spring, Complied by 同学的姓名、院系

> **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：

代码：

```
class Solution(object):
    def permute(self, nums):
        """
        :type nums: List[int]
        :rtype: List[List[int]]
        """
        nums.sort()
        ans=[]
        if len(nums)==0:
            return [[]]
        for i in range(len(nums)):
            cur=nums[i]
            remain=nums[:i]+nums[i+1:]
            repermute=self.permute(remain)
            for a in repermute:
```

```
                ans.append([cur]+a)
        return ans
```

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：

```
def dfs(x,y,board,word,d,m,n,visited):
    if word=='':
        return True
    for a,b in d:
        nx,ny=x+a,y+b
        if 0<=nx<m and 0<=ny<n:
            if board[nx][ny]==word[0] and visited[nx][ny]:
                visited[nx][ny]=False
                if dfs(nx,ny,board,word[1:],d,m,n,visited):
                    return True
                visited[nx][ny]=True
    return False

class Solution(object):
    def exist(self, board, word):
        """
        :type board: List[List[str]]
        :type word: str
        :rtype: bool
        """
        d=[(0,1),(0,-1),(1,0),(-1,0)]
        m=len(board)
```

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

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/

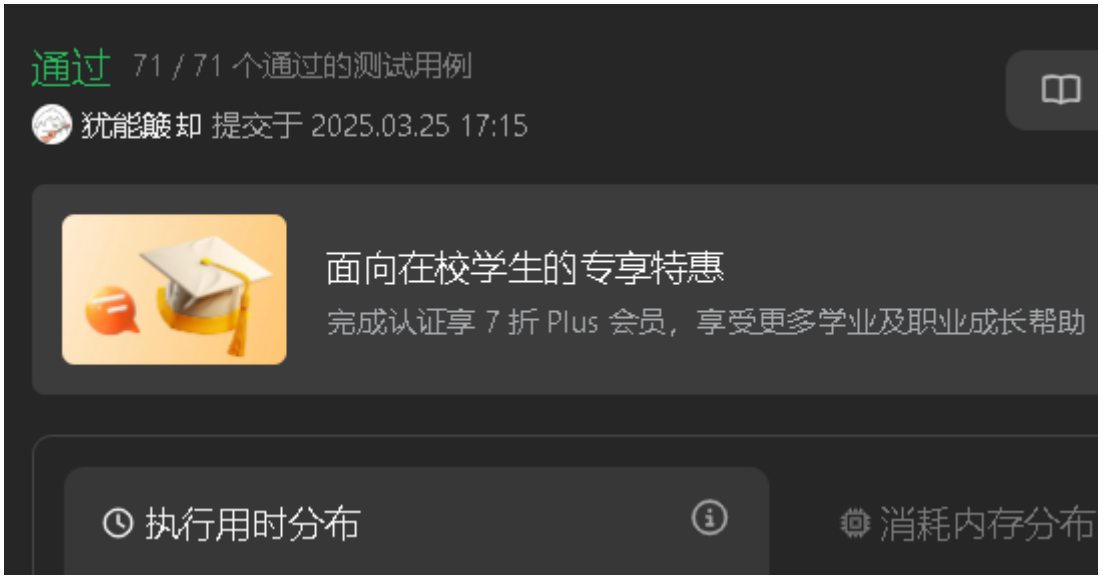
思路：

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]
        """
        ans=[]
```

```
        def dfs(root):
            if root:
                dfs(root.left)
                ans.append(root.val)
                dfs(root.right)
        dfs(root)
        return ans
```

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
from collections import deque
class Solution(object):
    def levelOrder(self, root):
        """
        :type root: Optional[TreeNode]
        :rtype: List[List[int]]
        """
        inq=set()
        if not root:
            return []
        ans=[[root.val]]
```

```
        inq.add(root.val)
        q=deque([(root,0)])
        while q:
            node,cnt=q.popleft()
            if node.left:
                q.append((node.left,cnt+1))
                if len(ans)>cnt+1:
                    ans[cnt+1].append(node.left.val)
                else:
                    ans.append([node.left.val])
            if node.right:
                q.append((node.right,cnt+1))
                if len(ans)>cnt+1:
                    ans[cnt+1].append(node.right.val)
                else:
                    ans.append([node.right.val])
        return ans
```

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：

```python
class Solution(object):
    def partition(self, s):
        """
        :type s: str
        :rtype: List[List[str]]
        """
        n=len(s)
        ans=[]
        record=[[True]*n for i in range(n)]
        for i in range(n-1,-1,-1):
            for j in range(i+1,n):
                if s[i]!=s[j] or record[i+1][j-1]==False:
                    record[i][j]=False
        arr=[]
        def dfs(i,arr):
            for j in range(i,n):
                if record[i][j]:
                    dfs(j+1,arr+[s[i:j+1]])
            if i==n:
                ans.append(arr)
        dfs(0,[])
        return ans
```

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：

```python
class LRUCache(object):

    def __init__(self, capacity):
        """
        :type capacity: int
        """
        self.capacity=capacity
        self.cache={}
        self.record=[]

    def get(self, key):
        """
        :type key: int
        :rtype: int
        """
        if key in self.cache:
            self.record.remove(key)
            self.record.append(key)
            return self.cache[key]
        return -1

    def put(self, key, value):
        """
        :type key: int
        :type value: int
        :rtype: None
        """
        self.cache[key]=value
        if key in self.record:
            self.record.remove(key)
        self.record.append(key)
        while len(self.cache)>self.capacity:
            self.cache.pop(self.record.pop(0))
```

代码运行截图 （至少包含有"Accepted"）

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。

树好抽象啊，感觉和链表相关的都有点抽象，不过有关bfs、dsf的题目靠着上学期攒下来的底子还是能比较顺利地写出来的，但dp就不行了……

这周作业完成比较早，争取多做点每日选做