

assignment7

Assignment #7: bfs、

Updated 0851 GMT+8 Oct 21, 2025

2025 fall, Complied by 杨知进 物理学院

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本人头像，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

M23555: 节省存储的矩阵乘法

implementation, matrices, <http://cs101.openjudge.cn/practice/23555>

要求用节省内存的方式实现，不能还原矩阵的方式实现。

思路：

代码：

代码运行截图 (至少包含有"Accepted")

M102.二叉树的层序遍历

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

思路：

代码：

```
class Solution:
    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        if not root:
            return []
        queue = [root]
        res = []
        while queue:
            res.append([node.val for node in queue])
            lis = []
            for node in queue:
                if node.left:
                    lis.append(node.left)
                if node.right:
                    lis.append(node.right)
            queue = lis
        return res
```

代码运行截图 (至少包含有"Accepted")

The screenshot shows a LeetCode submission page for a Python3 solution. The code is identical to the one above. The submission was stored and accepted. The runtime was 17 ms. The status bar at the bottom right shows '通过' (Accepted) and '写题解' (Write Solution).

```
Python3 ▾ 智能模式
1 # Definition for a binary tree node.
2 # class TreeNode:
3 #     def __init__(self, val=0, left=None, right=None):
4 #         self.val = val
5 #         self.left = left
6 #         self.right = right
7 class Solution:
8     def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
9         if not root:
10             return []
11         queue = [root]
12         res = []
13         while queue:
14             res.append([node.val for node in queue])
15             lis = []
16             for node in queue:
17                 if node.left:
18                     lis.append(node.left)
19                 if node.right:
20                     lis.append(node.right)
21             queue = lis
22         return res
```

已存储

题目描述 | 题解 | 通过 × | 提交记录

← 全部提交记录

通过 35 / 35 个通过的测试用例 用时: 17 m 1 s

官方题解 写题解

用时：14min左右

M131. 分割回文串

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

思路：

代码：

```
class Solution:
    def partition(self, s: str) -> List[List[str]]:
        self.check = lambda s: s == s[::-1]
        ans = []
        self.trace(s, ans, [])
        return ans
    def trace(self, s, ans, path):
        if not s:
            ans.append(path)
            return
        for i in range(1, len(s) + 1):
            if self.check(s[:i]):
                self.trace(s[i:], ans, path + [s[:i]])
```

代码运行截图 (至少包含有"Accepted")

The screenshot shows a Python3 code editor with the provided solution. Below the code, a message indicates it has been stored. At the bottom, there are navigation links: 题目描述 (Description), 题解 (Solution), 通过 (Accepted), 提交记录 (Submission History), and 全部提交记录 (All Submission History). A status bar at the bottom shows 通过 32 / 32 个通过的测试用例 (Accepted 32 / 32 test cases) and two buttons: 官方题解 (Official Solution) and 写题解 (Write Solution).

```
1 class Solution:
2     def partition(self, s: str) -> List[List[str]]:
3         self.check = lambda s: s == s[::-1]
4         ans = []
5         self.trace(s, ans, [])
6         return ans
7     def trace(self, s, ans, path):
8         if not s:
9             ans.append(path)
10            return
11        for i in range(1, len(s) + 1):
12            if self.check(s[:i]):
13                self.trace(s[i:], ans, path + [s[:i]])
```

用时：20min左右

M200.岛屿数量

dfs, bfs, <https://leetcode.cn/problems/number-of-islands/>

思路：

找到1就count+=1，并把这个1变成0，连在一起的也变成0；继续往后找1。

代码：

```
class Solution:
    def numIslands(self, grid: List[List[str]]) -> int:
        r = len(grid)
        if r == 0:
            return 0
        c = len(grid[0])
        count=0
        for i in range(r):
            for j in range(c):
                if grid[i][j] == "1":
                    count += 1
                    self.dfs(grid, i, j)
        return count
    def dfs(self,grid,i,j):
        grid[i][j] = 0
        r, c = len(grid), len(grid[0])
        for nx, ny in [(i - 1, j), (i + 1, j), (i, j - 1), (i, j + 1)]:
            if 0 <= nx < r and 0 <= ny < c and grid[nx][ny] == "1":
                self.dfs(grid, nx, ny)
```

(至少包含有"Accepted")

```
Python3 ▾ 智能模式
1 class Solution:
2     def numIslands(self, grid: List[List[str]]) -> int:
3         r = len(grid)
4         if r == 0:
5             return 0
6         c = len(grid[0])
7         count=0
8         for i in range(r):
9             for j in range(c):
10                 if grid[i][j] == "1":
11                     count += 1
12                     self.dfs(grid, i, j)
13         return count
14     def dfs(self,grid,i,j):
15         grid[i][j] = 0
16         r, c = len(grid), len(grid[0])
17         for nx, ny in [(i - 1, j), (i + 1, j), (i, j - 1), (i, j + 1)]:
18             if 0 <= nx < r and 0 <= ny < c and grid[nx][ny] == "1":
19                 self.dfs(grid, nx, ny)

已存储
```

题目描述 | 通过 × | 题解 | 提交记录

← 全部提交记录

通过 49 / 49 个通过的测试用例

官方题解

写题解

25fall Algo/week7/image 1.png

用时：18min左右

1123.最深叶节点的最近公共祖先

dfs, <https://leetcode.cn/problems/lowest-common-ancestor-of-deepest-leaves/>

思路：

对每个节点来说，更深的那边是局部最优解；从下往上搜索

代码

```
class Solution:
    def lcaDeepestLeaves(self, root: Optional[TreeNode]) ->
Optional[TreeNode]:
    return self.dfs(root)[1]
def dfs(self, node):
    if node is None:
        return 0, None
    lh, lnode = self.dfs(node.left)
    rh, rnode = self.dfs(node.right)
    if lh > rh:
        return lh + 1, lnode
    elif lh < rh:
```

```
        return rh + 1, rnode
    else:
        return lh + 1, node
```

(至少包含有"Accepted")

Python3 ✓ 智能模式

```
3     def __init__(self, val=0, left=None, right=None):
4         self.val = val
5         self.left = left
6         self.right = right
7     class Solution:
8         def lcaDeepestLeaves(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
9             return self.dfs(root)[1]
10        def dfs(self, node):
11            if node is None:
12                return 0, None
13            lh, lnode = self.dfs(node.left)
14            rh, rnode = self.dfs(node.right)
15            if lh > rh:
16                return lh + 1, lnode
17            elif lh < rh:
18                return rh + 1, rnode
19            else:
20                return lh + 1, node
```

已存储

题目描述 | 题解 | 通过 × | 提交记录

← 全部提交记录

通过 81 / 81 个通过的测试用例

□ 官方题解 □ 官方题解

image.png

用时：30min左右

M79. 单词搜索

回溯，<https://leetcode.cn/problems/word-search/>

思路：

代码：

代码运行截图 (至少包含有"Accepted")

2. 学习总结和个人收获

写dfs、bfs一定程度上可以先套框架，不去纠结细节。程式化一些，然后再针对问题修改。