Assignment #7: bfs、🌲

Updated 0851 GMT+8 Oct 21, 2025

2025 fall, Complied by 胡孝齐 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： 对于最终结果的r行c列，查找符合要求的元素并乘起来

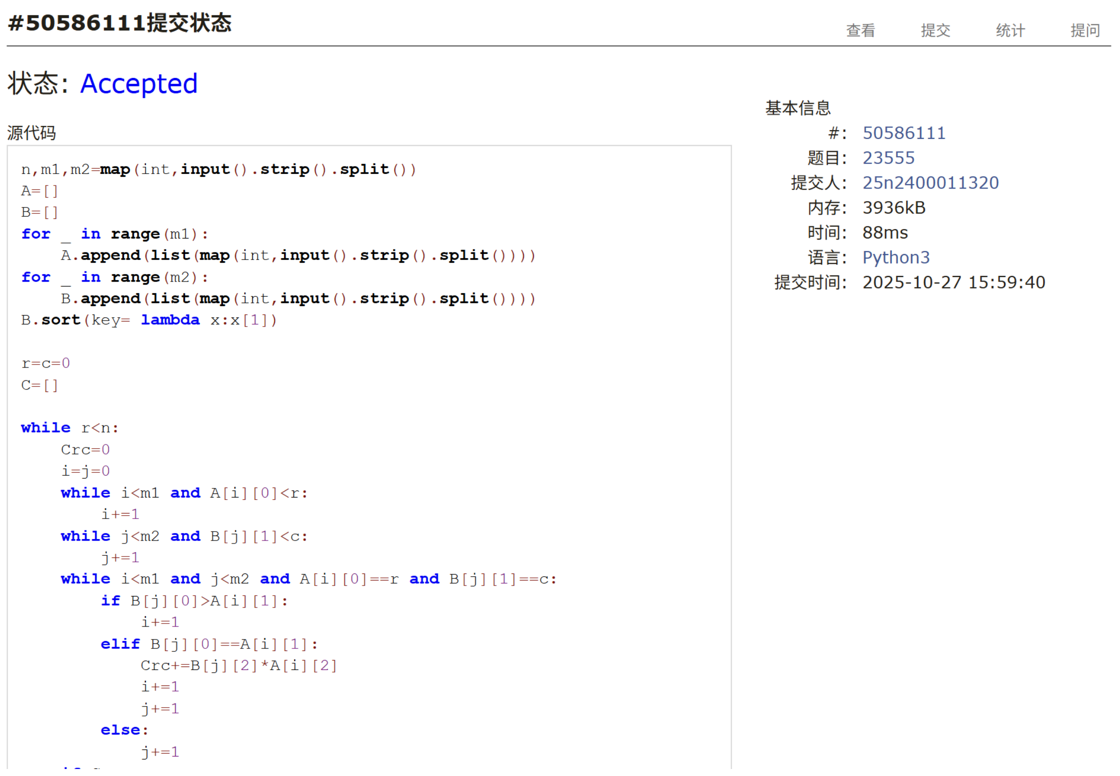
代码：

```python
n,m1,m2=map(int,input().strip().split())
A=[]
B=[]
for _ in range(m1):
    A.append(list(map(int,input().strip().split())))
for _ in range(m2):
    B.append(list(map(int,input().strip().split())))
B.sort(key= lambda x:x[1])

r=c=0
C=[]

while r<n:
    Crc=0
    i=j=0
    while i<m1 and A[i][0]<r:
        i+=1
    while j<m2 and B[j][1]<c:
        j+=1
    while i<m1 and j<m2 and A[i][0]==r and B[j][1]==c:
        if B[j][0]>A[i][1]:
            i+=1
        elif B[j][0]==A[i][1]:
            Crc+=B[j][2]*A[i][2]
            i+=1
            j+=1
        else:
            j+=1
    if Crc:
        C.append([r,c,Crc])
    if c<n-1:
        c+=1
    else:
        r+=1
        c=0
for num in C:
    print(*num)
```

代码运行截图 （至少包含有"Accepted"）



#50586111提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#:	50586111
题目:	23555
提交人:	25n2400011320
内存:	3936kB
时间:	88ms
语言:	Python3
提交时间:	2025-10-27 15:59:40

源代码

```
n,m1,m2=map(int,input().strip().split())
A=[]
B=[]
for _ in range(m1):
    A.append(list(map(int,input().strip().split())))
for _ in range(m2):
    B.append(list(map(int,input().strip().split())))
B.sort(key= lambda x:x[1])

r=c=0
C=[]

while r<n:
    Crc=0
    i=j=0
    while i<m1 and A[i][0]<r:
        i+=1
    while j<m2 and B[j][1]<c:
        j+=1
    while i<m1 and j<m2 and A[i][0]==r and B[j][1]==c:
        if B[j][0]>A[i][1]:
            i+=1
        elif B[j][0]==A[i][1]:
            Crc+=B[j][2]*A[i][2]
            i+=1
            j+=1
        else:
            j+=1
```

M102.二叉树的层序遍历

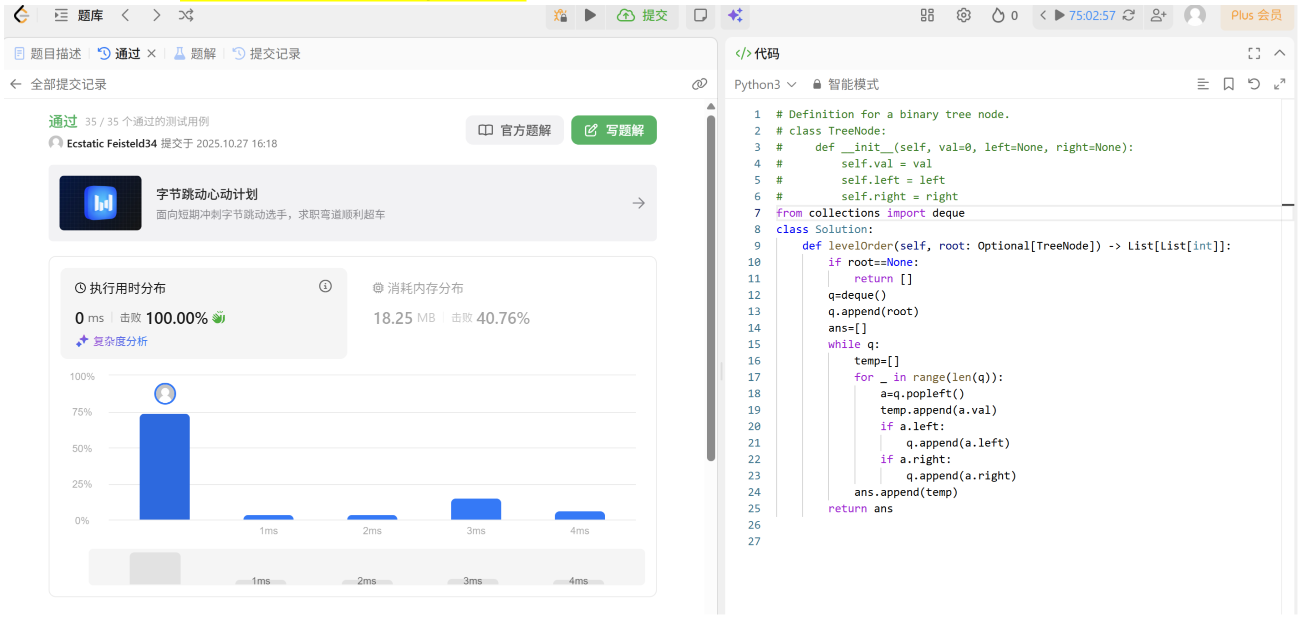
bfs, https://leetcode.cn/problems/binary-tree-level-order-traversal/

思路： 树通过left和right进行访问下一层，直接使用bfs即可

代码：

```
from collections import deque
class Solution:
    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        if root==None:
            return []
        q=deque()
        q.append(root)
        ans=[]
        while q:
            temp=[]
            for _ in range(len(q)):
                a=q.popleft()
                temp.append(a.val)
                if a.left:
                    q.append(a.left)
                if a.right:
                    q.append(a.right)
            ans.append(temp)
        return ans
```

代码运行截图 （至少包含有"Accepted"）



通过 35 / 35 个通过的测试用例

Ecstatic Feisteld34 提交于 2025.10.27 16:18

执行用时分布
0 ms | 击败 100.00%

消耗内存分布
18.25 MB | 击败 40.76%

```
# 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
from collections import deque
class Solution:
    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        if root==None:
            return []
        q=deque()
        q.append(root)
        ans=[]
        while q:
            temp=[]
            for _ in range(len(q)):
                a=q.popleft()
                temp.append(a.val)
                if a.left:
                    q.append(a.left)
                if a.right:
                    q.append(a.right)
            ans.append(temp)
        return ans
```

M131.分割回文串

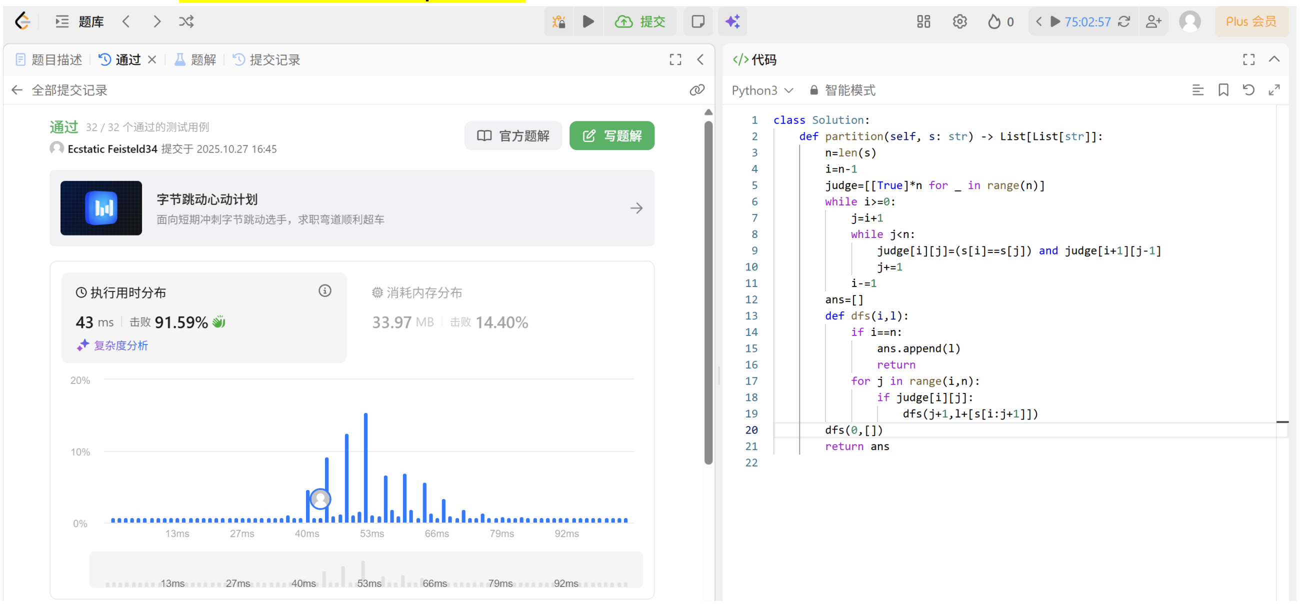
dp, backtracking, https://leetcode.cn/problems/palindrome-partitioning/

思路： 判断第i个元素到第j个元素是否为回文串的方法比较新颖，可以采用动态规划，比直接判断快

代码：

```
class Solution:
    def partition(self, s: str) -> List[List[str]]:
        n=len(s)
        i=n-1
        judge=[[True]*n for _ in range(n)]
        while i>=0:
            j=i+1
            while j<n:
                judge[i][j]=(s[i]==s[j]) and judge[i+1][j-1]
                j+=1
            i-=1
        ans=[]
        def dfs(i,l):
            if i==n:
                ans.append(l)
                return
            for j in range(i,n):
                if judge[i][j]:
                    dfs(j+1,l+[s[i:j+1]])
        dfs(0,[])
        return ans
```

代码运行截图 （至少包含有"Accepted"）



题库 提交 通过 32 / 32 个通过的测试用例

Ecstatic Feisteld34 提交于 2025.10.27 16:45

字节跳动心动计划

面向短期冲刺字节跳动选手，求职弯道顺利超车

执行用时分布 43 ms | 击败 91.59%

消耗内存分布 33.97 MB | 击败 14.40%

Python3 智能模式

```
class Solution:
    def partition(self, s: str) -> List[List[str]]:
        n=len(s)
        i=n-1
        judge=[[True]*n for _ in range(n)]
        while i>=0:
            j=i+1
            while j<n:
                judge[i][j]=(s[i]==s[j]) and judge[i+1][j-1]
                j+=1
            i-=1
        ans=[]
        def dfs(i,l):
            if i==n:
                ans.append(l)
                return
            for j in range(i,n):
                if judge[i][j]:
                    dfs(j+1,l+[s[i:j+1]])
        dfs(0,[])
        return ans
```

M200.岛屿数量

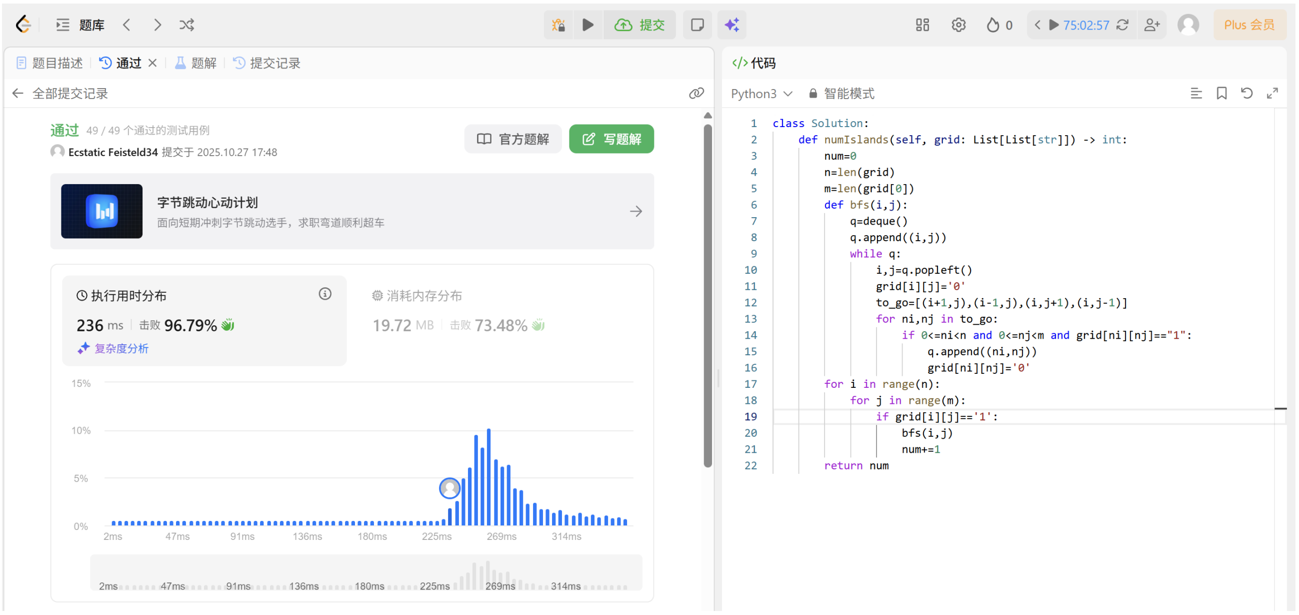
dfs, bfs, https://leetcode.cn/problems/number-of-islands/

思路： 使用bfs解决单个岛屿，然后直接将已经计数的岛屿抹除即可

代码

```
class Solution:
    def numIslands(self, grid: List[List[str]]) -> int:
        num=0
        n=len(grid)
        m=len(grid[0])
        def bfs(i,j):
            q=deque()
            q.append((i,j))
            while q:
                i,j=q.popleft()
                grid[i][j]='0'
                to_go=[(i+1,j),(i-1,j),(i,j+1),(i,j-1)]
                for ni,nj in to_go:
                    if 0<=ni<n and 0<=nj<m and grid[ni][nj]=="1":
                        q.append((ni,nj))
                        grid[ni][nj]='0'
        for i in range(n):
            for j in range(m):
                if grid[i][j]=='1':
                    bfs(i,j)
                    num+=1
        return num
```

（至少包含有"Accepted"）



通过 49 / 49 个通过的测试用例

Ecstatic Feisteld34 提交于 2025.10.27 17:48

执行用时分布 236 ms 击败 96.79%

消耗内存分布 19.72 MB 击败 73.48%

Python3 智能模式

```python
class Solution:
    def numIslands(self, grid: List[List[str]]) -> int:
        num=0
        n=len(grid)
        m=len(grid[0])
        def bfs(i,j):
            q=deque()
            q.append((i,j))
            while q:
                i,j=q.popleft()
                grid[i][j]='0'
                to_go=[(i+1,j),(i-1,j),(i,j+1),(i,j-1)]
                for ni,nj in to_go:
                    if 0<=ni<n and 0<=nj<m and grid[ni][nj]=="1":
                        q.append((ni,nj))
                        grid[ni][nj]='0'
        for i in range(n):
            for j in range(m):
                if grid[i][j]=='1':
                    bfs(i,j)
                    num+=1
        return num
```

1123.最深叶节点的最近公共祖先

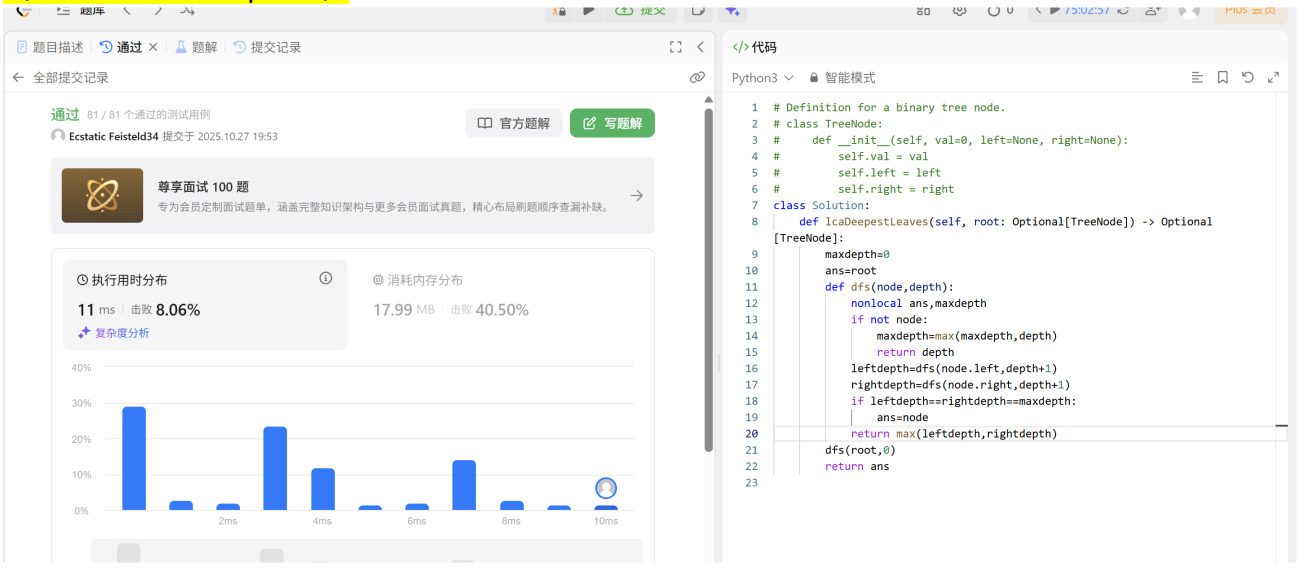
dfs, https://leetcode.cn/problems/lowest-common-ancestor-of-deepest-leaves/

思路：

代码

```python
class Solution:
    def lcaDeepestLeaves(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
        maxdepth=0
        ans=root
        def dfs(node,depth):
            nonlocal ans,maxdepth
            if not node:
                maxdepth=max(maxdepth,depth)
                return depth
            leftdepth=dfs(node.left,depth+1)
            rightdepth=dfs(node.right,depth+1)
            if leftdepth==rightdepth==maxdepth:
                ans=node
            return max(leftdepth,rightdepth)
        dfs(root,0)
        return ans
```

（至少包含有"Accepted"）



通过 81 / 81 个通过的测试用例

Ecstatic Feisteld34 提交于 2025.10.27 19:53

尊享面试 100 题

专为会员定制面试题单，涵盖完整知识架构与更多会员面试真题，精心布局刷题顺序查漏补缺。

执行用时分布

11 ms | 击败 8.06%

复杂度分析

消耗内存分布

17.99 MB | 击败 40.50%

```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 lcaDeepestLeaves(self, root: Optional[TreeNode]) -> Optional
[TreeNode]:
        maxdepth=0
        ans=root
        def dfs(node,depth):
            nonlocal ans,maxdepth
            if not node:
                maxdepth=max(maxdepth,depth)
                return depth
            leftdepth=dfs(node.left,depth+1)
            rightdepth=dfs(node.right,depth+1)
            if leftdepth==rightdepth==maxdepth:
                ans=node
            return max(leftdepth,rightdepth)
        dfs(root,0)
        return ans
```

M79.单词搜索

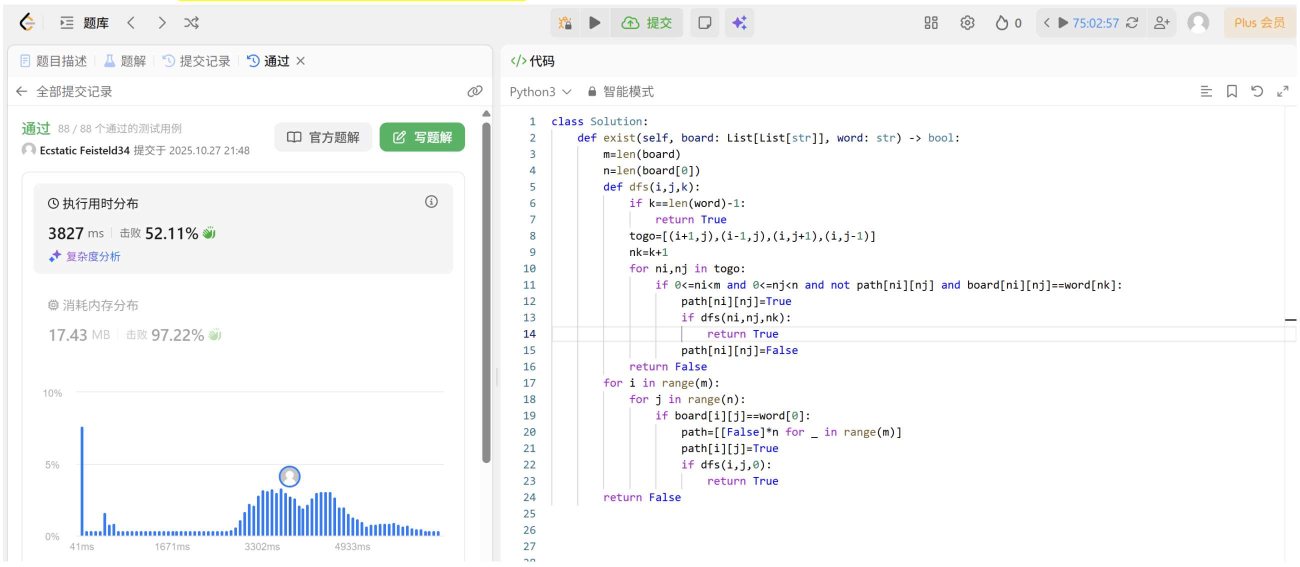
回溯，https://leetcode.cn/problems/word-search/

思路： 本题我一开始用的是bfs，bfs只能用三维列表，否则只能深拷贝了（深拷贝超时），索性就尝试了一下dfs，正好也挺久没用了。

代码：

```python
class Solution:
    def exist(self, board: List[List[str]], word: str) -> bool:
        m=len(board)
        n=len(board[0])
        def dfs(i,j,k):
            if k==len(word)-1:
                return True
            togo=[(i+1,j),(i-1,j),(i,j+1),(i,j-1)]
            nk=k+1
            for ni,nj in togo:
                if 0<=ni<m and 0<=nj<n and not path[ni][nj] and board[ni]
[nj]==word[nk]:
                    path[ni][nj]=True
                    if dfs(ni,nj,nk):
                        return True
                    path[ni][nj]=False
            return False
        for i in range(m):
            for j in range(n):
                if board[i][j]==word[0]:
                    path=[[False]*n for _ in range(m)]
                    path[i][j]=True
                    if dfs(i,j,0):
                        return True
        return False
```

代码运行截图 （至少包含有"Accepted"）



通过 88 / 88 个通过的测试用例

Ecstatic Feisteld34 提交于 2025.10.27 21:48

执行用时分布

3827 ms | 击败 52.11%

消耗内存分布

17.43 MB | 击败 97.22%

Python3 智能模式

```python
class Solution:
    def exist(self, board: List[List[str]], word: str) -> bool:
        m=len(board)
        n=len(board[0])
        def dfs(i,j,k):
            if k==len(word)-1:
                return True
            togo=[(i+1,j),(i-1,j),(i,j+1),(i,j-1)]
            nk=k+1
            for ni,nj in togo:
                if 0<=ni<m and 0<=nj<n and not path[ni][nj] and board[ni][nj]==word[nk]:
                    path[ni][nj]=True
                    if dfs(ni,nj,nk):
                        return True
                    path[ni][nj]=False
            return False
        for i in range(m):
            for j in range(n):
                if board[i][j]==word[0]:
                    path=[[False]*n for _ in range(m)]
                    path[i][j]=True
                    if dfs(i,j,0):
                        return True
        return False
```

2. 学习总结和个人收获

练习了bfs和dfs，学习了基本的树的一些操作