

Assignment #7: bfs、

Updated 0851 GMT+8 Oct 21, 2025

2025 fall, Complied by 杨浩、化院

1. 1. 题目

1.1 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：

- 两个矩阵一个按行存，一个按列存

代码：

```

1 n,m1,m2=map(int,input().split())
2 l1=[[] for _ in range(n)]
3 for i in range(m1):
4     shuru=tuple(map(int,input().split()))
5     l1[shuru[1]].append(shuru[:])
6 l2=[[] for _ in range(n)]
7 for i in range(m2):
8     shuru=tuple(map(int,input().split()))
9     l2[shuru[0]].append(shuru[:])
10 ans={}
11 for i in range(n):
12     for j in l1[i]:
13         for k in l2[i]:
14             ans.setdefault((j[0],k[1]),0)
15             ans[(j[0],k[1])] +=j[2]*k[2]
16 res=sorted(list(ans.items()),key=lambda x:(x[0][0],x[0][1]))
17 for i in res:
18     print(f'{i[0][0]} {i[0][1]} {i[1]}')

```

Fence 1

代码运行截图 (至少包含有"Accepted")

#50516924提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
n,m1,m2=map(int,input().split())
l1=[[]]
for _ in range(n):
    for i in range(m1):
        shuru=tuple(map(int,input().split()))
        l1[shuru[1]].append(shuru[:])
l2=[[]]
for _ in range(n):
    for i in range(m2):
        for j in range(m1):
            l2.append(l1[i][j])
for i in range(m2):
    print(*l2[i])
```

基本信息

#: 50516924
题目: 2355
提交人: 25n2400011769
内存: 3904kB
时间: 26ms
语言: Python3
提交时间: 2025-10-23 13:11:06

Figure 1

1.2 M102.二叉树的层序遍历

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

思路:

- bfs

代码:

```
1 class Solution:
2     def levelOrder(self, root: Optional[TreeNode]) ->
3         List[List[int]]:
4             l=deque([])
5             if root==None:
6                 return []
7             l.append((root,1))
8             res=[]
9             while l:
10                 t=l.popleft()
11                 if len(res)<t[1]:
12                     res.append([])
13                 res[-1].append(t[0].val)
14                 if t[0].left:
15                     l.append((t[0].left,t[1]+1))
16                 if t[0].right:
17                     l.append((t[0].right,t[1]+1))
18             return res
```

Fence 2

代码运行截图 (至少包含有"Accepted")

通过 35 / 35 个通过的测试用例

AND-Y 提交于 2025.10.13 16:09

官方题解

写题解

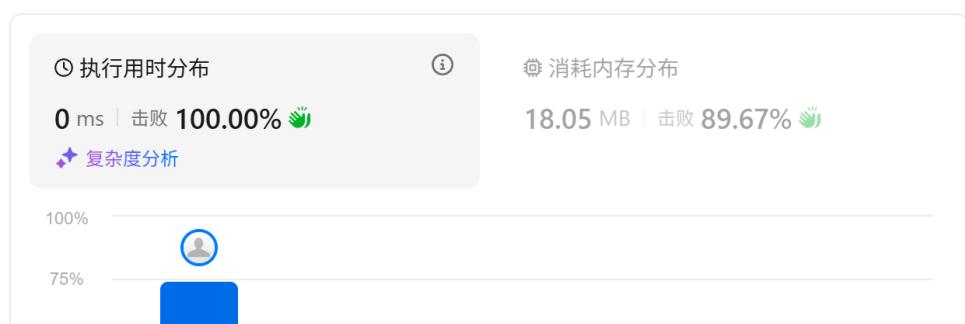


Figure 2

1.3 M131.分割回文串

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

思路:

- 回溯+dfs

代码:

```

1 class Solution:
2     def partition(self, s: str) -> List[List[str]]:
3         res=[]
4         def check(astr):
5             if astr[::-1]==astr:
6                 return True
7             else:
8                 return False
9         def dfs(s,res,path,deep,size):
10            if s=='':
11                res.append(path[:])
12                return
13            for i in range(1,size+1):
14                a=s[:i]
15                if check(a):
16                    path.append(a)
17                    dfs(s[i:],res,path,deep,len(s[i:]))
18                    path.pop()
19
20
21        dfs(s,res,[],0,len(s))
22        return res

```

Fence 3

代码运行截图 (至少包含有"Accepted")

通过 32 / 32 个通过的测试用例

AND-Y 提交于 2025.08.31 15:21

官方题解

写题解

① 执行用时分布

②

59 ms | 击败 37.75%

消耗内存分布

31.95 MB | 击败 94.13% 🌟

复杂度分析



Figure 3

1.4 M200.岛屿数量

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

思路:

- bfs

代码

```

1 class Solution:
2     def numIslands(self, grid: List[List[str]]) -> int:
3         def bfs(alist,m,n):
4             blist=[]
5             for x,y in alist:
6                 for dx,dy in delta:
7                     if 0<=x+dx<m and 0<=y+dy<n:
8                         if grid[x+dx][y+dy]=='1':
9                             blist.append((x+dx,y+dy))
10                            grid[x+dx][y+dy]='0'
11             if blist:
12                 bfs(blist,m,n)
13             num=0
14             m=len(grid)
15             n=len(grid[0])
16             delta=[(0,1),(0,-1),(1,0),(-1,0)]
17             for i in range(len(grid)):
18                 for j in range(len(grid[0])):
19                     if grid[i][j]=='1':
20                         num +=1
21                         grid[i][j]='0'
22                         bfs([(i,j)],m,n)
23             return num

```

Fence 4

(至少包含有"Accepted")

通过 49 / 49 个通过的测试用例

AND-Y 提交于 2025.10.23 19:02

官方题解

写题解



高频 SQL 50 题

面试考点全覆盖应对高阶数据库面试



① 执行用时分布



257 ms | 击败 67.52% 🏆

复杂度分析

消耗内存分布

19.70 MB | 击败 79.99% 🏆

15%



Figure 4

1.5 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：

- 先用bfs找到最深叶节点再遍历最近公共祖先

代码

```

1 class Solution:
2     def lcaDeepestLeaves(self, root: Optional[TreeNode]) ->
3         Optional[TreeNode]:
4             duilie=deque([[root]])
5             ans=[]
6             deep=0
7             while duilie:
8                 if deep<len(duilie[0]):
9                     deep+=1
10                    ans=list(duilie)
11                    l=duilie.popleft()
12                    pr=l[-1]
13                    if pr.left:
14                        duilie.append(pr.left)
15                    if pr.right:
16                        duilie.append(pr.right)
17            return ans[deep-1]

```

```

13         res=l[:]
14         res.append(pr.left)
15         duilie.append(res[:])
16     if pr.right:
17         res=l[:]
18         res.append(pr.right)
19         duilie.append(res[:])
20     for i in range(deep):
21         t=ans[0][i]
22         for j in ans:
23             if j[i]!=t:
24                 return j[i-1]
25

```

Fence 5

(至少包含有"Accepted")

通过 81 / 81 个通过的测试用例

AND-Y 提交于 2025.10.23 19:28

官方题解

写题解

① 执行用时分布

7 ms | 击败 26.43%

复杂度分析

消耗内存分布

17.93 MB | 击败 51.57%



Figure 5

1.6 M79.单词搜索

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

思路:

- dfs

代码:

```

1  class Solution:
2      def exist(self, board: List[List[str]], word: str) ->
3          bool:
4              def dfs(word,x,y,m,n,deep):
5                  if deep==len(word):
6                      return True
7                  else:
8                      for dx,dy in delta:
9                          if 0<=x+dx<m and 0<=y+dy<n:
10                             if hax[x+dx][y+dy] and board[x+dx]
11                                [y+dy]==word[deep]:
12                                 hax[x+dx][y+dy]=False
13                                 if
14                                     dfs(word,x+dx,y+dy,m,n,deep+1):
15                                         return True
16                                     hax[x+dx][y+dy]=True
17
18                                     m=len(board)
19                                     n=len(board[0])
20                                     hax=[[True for __ in range(n)] for _ in range(m)]
21                                     begin=[]
22                                     for i in range(m):
23                                         for j in range(n):
24                                             if board[i][j] not in word:
25                                                 hax[i][j]=False
26                                             if board[i][j]==word[0]:
27                                                 begin.append((i,j))
28                                             delta=[(0,1),(0,-1),(1,0),(-1,0)]
29                                             for i,j in begin:
30                                                 hax[i][j]=False
31                                                 if dfs(word,i,j,m,n,1):
32                                                     return True
33                                                 hax[i][j]=True
34
35                                     return False

```

Fence 6

代码运行截图 (至少包含有"Accepted")

通过 88 / 88 个通过的测试用例
AND-Y 提交于 2025.10.23 20:23

官方题解

写题解

① 执行用时分布

(i)

3230 ms | 击败 74.97% 🟢

复杂度分析

消耗内存分布

17.59 MB | 击败 58.81% 🟢

10%

Figure 6

2. 2. 学习总结和个人收获

学习了二叉树的前序、中序、后序遍历。实现不同树的不同功能需要使用这三种遍历方法。二叉搜索树使用中序遍历可以按大小顺序遍历，后序遍历可以用于找二叉树的直径。现阶段可以顺利地写出递归来解决问题，但自己写出来的递归通常很冗长，不够简洁优雅。