

# 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:

```
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

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
    r+=1
    C.append([r,c,Crc])
    c+=1

```

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

## 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")

The screenshot shows the LeetCode platform interface. On the left, there's a navigation bar with '题目描述', '通过', '题解', and '提交记录'. Below it, a card for '字节跳动心动计划' (ByteDance Heartbeat Plan) shows '面向短期冲刺字节跳动选手，求职弯道顺利超车'. On the right, the code editor displays the Python solution. The execution results show '通过 35 / 35 个通过的测试用例' and a timestamp 'Ecstatic Feisteld34 提交于 2025.10.27 16:18'. The execution time distribution chart indicates most operations are near 0ms, with some spikes at 1ms, 2ms, 3ms, and 4ms.

```

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 from collections import deque
8
9 class Solution:
10    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
11        if root==None:
12            return []
13        q=deque()
14        q.append(root)
15        ans=[]
16        while q:
17            temp=[]
18            for _ in range(len(q)):
19                a=q.popleft()
20                temp.append(a.val)
21                if a.left:
22                    q.append(a.left)
23                if a.right:
24                    q.append(a.right)
25            ans.append(temp)
26
27        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")

```
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，求职赛道顺利通车

① 执行用时分布 ② 消耗内存分布
43 ms | 击败 91.59% 33.97 MB | 击败 14.40%
复杂度分析

20%
10%
0%
13ms 27ms 40ms 53ms 66ms 79ms 92ms
```

```

1 class Solution:
2     def partition(self, s: str) -> List[List[str]]:
3         n=len(s)
4         i=n-1
5         judge=[[True]*n for _ in range(n)]
6         while i>=0:
7             j=i+1
8             while j<n:
9                 if judge[i][j] and judge[i+1][j-1]:
10                    i+=1
11                    j+=1
12             ans=[]
13             def dfs(i,j):
14                 if i==n:
15                     ans.append(j)
16                     return
17                 for j in range(i,n):
18                     if judge[i][j]:
19                         dfs(j+1,1+[s[i:j+1]])
20             dfs(0,[])
21             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")

The screenshot shows a LeetCode problem submission page for problem 1123. The code is a Python3 solution using Breadth-First Search (BFS) to find the lowest common ancestor of the deepest leaves in a binary tree. The code is as follows:

```
1 class Solution:
2     def lcaDeepestLeaves(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
3         maxdepth=0
4         ans=root
5         def dfs(node, depth):
6             nonlocal ans, maxdepth
7             if not node:
8                 maxdepth=max(maxdepth, depth)
9                 return depth
10            leftdepth=dfs(node.left, depth+1)
11            rightdepth=dfs(node.right, depth+1)
12            if leftdepth==rightdepth==maxdepth:
13                ans=node
14            return max(leftdepth, rightdepth)
15        dfs(root, 0)
16        return ans
```

The page also displays execution time distribution (236 ms, 96.79% accepted), memory usage (19.72 MB, 73.48% accepted), and complexity analysis.

## 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

```
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
```



## 代码运行截图 (至少包含有"Accepted")

The screenshot shows a LeetCode submission page. At the top, it says "题目描述 | 题解 | 提交记录 | 通过" and "通过 88 / 88 个通过的测试用例". Below this, there's a "官方题解" button and a "写题解" button. On the left, there's a "执行用时分布" section with a histogram showing execution times from 41ms to 4933ms, with a peak around 3302ms. There's also a "消耗内存分布" section showing memory usage of 17.43 MB at 97.22%. The right side shows the Python3 code for the solution:

```
1 class Solution:
2     def exist(self, board: List[List[str]], word: str) -> bool:
3         m=len(board)
4         n=len(board[0])
5         def dfs(i,j,k):
6             if k==len(word)-1:
7                 return True
8             togo=[(i+1,j),(i-1,j),(i,j+1),(i,j-1)]
9             nk=k+1
10            for ni,nj in togo:
11                if 0<=ni<m and 0<=nj<n and not path[ni][nj] and board[ni][nj]==word[nk]:
12                    if dfs(ni,nj,nk):
13                        return True
14                    path[ni][nj]=False
15            return False
16        for i in range(m):
17            for j in range(n):
18                if board[i][j]==word[0]:
19                    path=[[False]*n for _ in range(m)]
20                    path[i][j]=True
21                    if dfs(i,j,0):
22                        return True
23
24
25
26
27
```

## 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