

# Assignment #B: 图为主

Updated 2223 GMT+8 Apr 29, 2025

2025 spring, Compiled by 任宇桐 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

### E07218:献给阿尔吉侬的花束

bfs, <http://cs101.openjudge.cn/practice/07218/>

思路:

看起来是简单的bfs, 但是感觉时间卡的有点紧。。。如果出队列的时候再加入visited就不能通过了, 必须在加入队列的同时加入visited。但是这两种做法用时竟然相差十倍以上, 还是让我有点惊讶的。

代码:

```
from collections import deque

def scope(x, y):
    return 0 <= x < r and 0 <= y < c

def bfs(x, y):
    q = deque([(x, y)])
    visited = {(x, y)} # 就是这里
    step = 0
    while q:
        s = len(q)
        for _ in range(s):
            x, y = q.popleft()
            for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0)]:
                nx = x+dx
```

```

        ny = y+dy
        if scope(nx, ny) and maze[nx][ny] != '#' and (nx, ny) not in
visited:

            if maze[nx][ny] == 'E':
                return step+1
            q.append((nx, ny))
            visited.add((nx, ny))

        step += 1
    return 'oop!'
t = int(input())
for _ in range(t):
    r,c = map(int, input().split())
    maze = [input() for _ in range(r)]
    for i in range(r):
        for j in range(c):
            if maze[i][j] == 'S':
                print(bfs(i, j))

```

代码运行截图 (至少包含有"Accepted")

**状态: Accepted**

源代码

```

from collections import deque

def scope(x, y):
    return 0 <= x < r and 0 <= y < c

def bfs(x, y):
    q = deque([(x, y)])
    visited = {(x, y)}
    step = 0
    while q:
        s = len(q)
        for _ in range(s):
            x, y = q.popleft()
            for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0)]:
                nx = x+dx
                ny = y+dy
                if scope(nx, ny) and maze[nx][ny] != '#' and (nx, ny) not in visited:
                    if maze[nx][ny] == 'E':
                        return step+1
                    q.append((nx, ny))
                    visited.add((nx, ny))
            step += 1
        return 'oop!'
t = int(input())
for _ in range(t):
    r, c = map(int, input().split())
    maze = [input() for _ in range(r)]
    for i in range(r):

```

基本信息

#: 49049458  
 题目: 07218  
 提交人: 24n2400011498  
 内存: 5636kB  
 时间: 98ms  
 语言: Python3  
 提交时间: 2025-05-02 18:58:20

## M3532.针对图的路径存在性查询I

disjoint set, <https://leetcode.cn/problems/path-existence-queries-in-a-graph-i/>

思路:

其实感觉比一般的并查集要简单,但是一开始一直沉浸在普通并查集的思路中无法逃脱,代码写的也不漂亮,最后受到ai的提示才写出简单的代码。

代码:

```

class Solution:

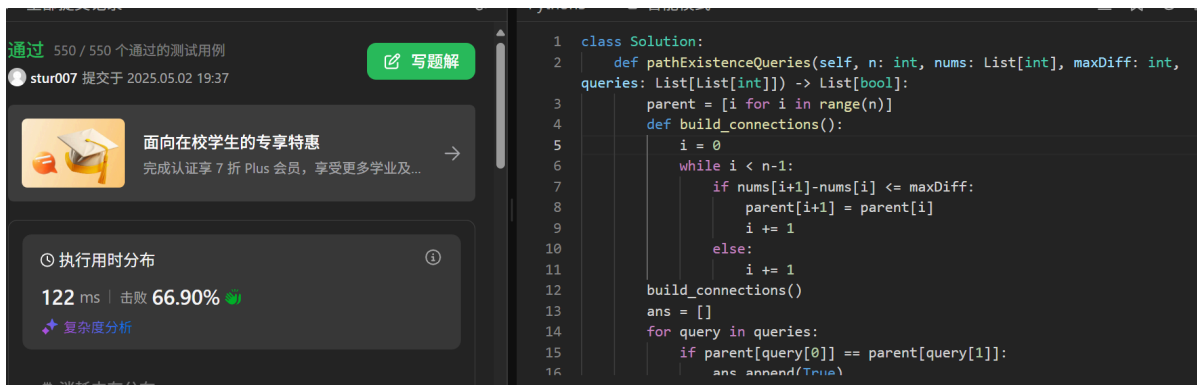
```

```

def pathExistenceQueries(self, n: int, nums: List[int], maxDiff: int,
queries: List[List[int]]) -> List[bool]:
    parent = [i for i in range(n)]
    def build_connections():
        i = 0
        while i < n-1:
            if nums[i+1]-nums[i] <= maxDiff:
                parent[i+1] = parent[i]
                i += 1
            else:
                i += 1
    build_connections()
    ans = []
    for query in queries:
        if parent[query[0]] == parent[query[1]]:
            ans.append(True)
        else:
            ans.append(False)
    return ans

```

代码运行截图 (至少包含有"Accepted")



## M22528:厚道的调分方法

binary search, <http://cs101.openjudge.cn/practice/22528/>

思路:

二分查找最小

代码:

```

s = [float(x) for x in input().split()]
s.sort()
i = int(0.4*len(s))
def check(a):
    a = a/1000000000
    return a*s[i]+1.1**(a*s[i]) >= 85
def binary_search():
    low = 0
    high = 1000000000
    while low <= high:

```

```

        mid = (low+high)//2
        if check(mid):
            high = mid - 1
        else:
            low = mid + 1

    return low
print(binary_search())

```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

```

s = [float(x) for x in input().split()]
s.sort()
i = int(0.4*len(s))
def check(a):
    a = a/1000000000
    return a*s[i]+1.1**(a*s[i]) >= 85
def binary_search():
    low = 0
    high = 1000000000
    while low <= high:
        mid = (low+high)//2
        if check(mid):
            high = mid - 1
        else:
            low = mid + 1

    return low
print(binary_search())

```

基本信息

#: 49049713

题目: 22528

提交人: 24n2400011498

内存: 18084kB

时间: 91ms

语言: Python3

提交时间: 2025-05-02 20:11:35

## Msy382: 有向图判环

dfs, <https://sunnywhy.com/sfbj/10/3/382>

思路:

基本和直接dfs差不多, 但是要分black, gray, white节点。

代码:

```

from collections import defaultdict
edges = defaultdict(set)
n, m = map(int, input().split())
for _ in range(m):
    u, v = map(int, input().split())
    edges[u].add(v)

visited = [-1]*n
def dfs(i):
    visited[i] = 0
    for vertex in edges[i]:
        if visited[vertex] == -1:
            if dfs(vertex):
                return True
        elif visited[vertex] == 0:
            return True
    visited[i] = 1
    return False

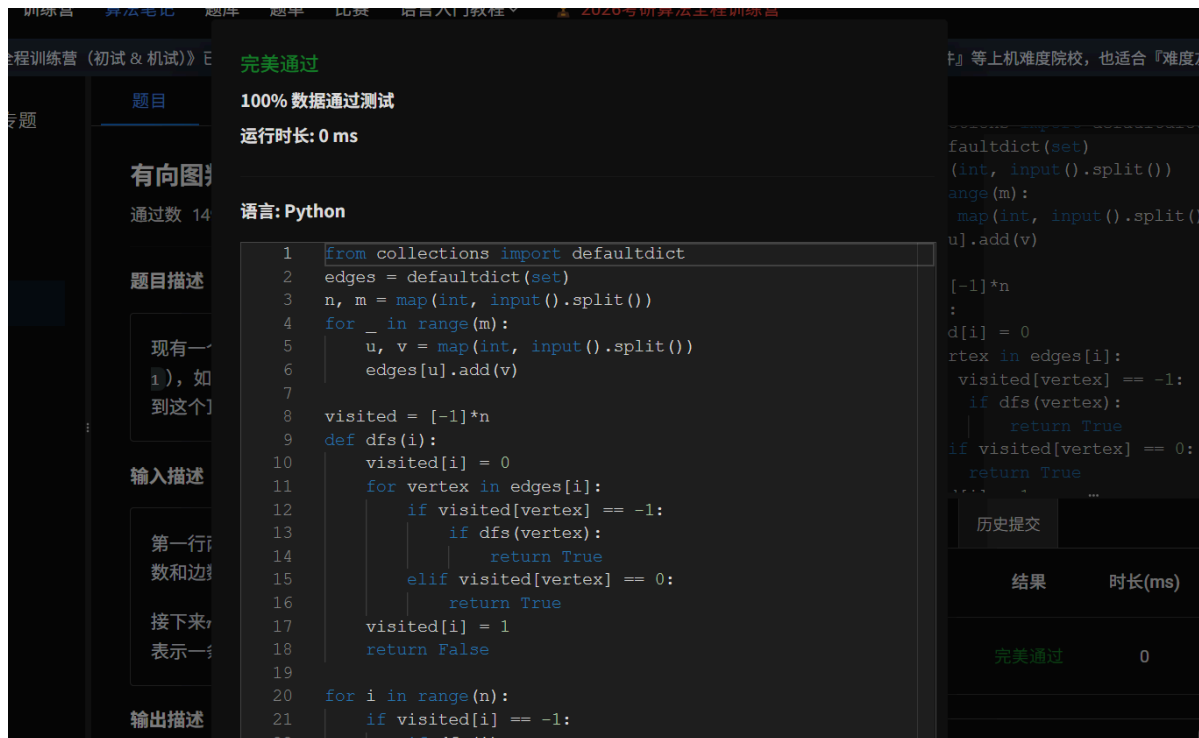
```

```

for i in range(n):
    if visited[i] == -1:
        if dfs(i):
            print('Yes')
            break
    else:
        print('No')

```

代码运行截图 (至少包含有"Accepted")



The screenshot shows a submission page for a problem titled '有向图' (Directed Graph). The solution is written in Python and uses a Depth-First Search (DFS) algorithm to determine if a path exists between two vertices in a directed graph. The code is as follows:

```

1 from collections import defaultdict
2 edges = defaultdict(set)
3 n, m = map(int, input().split())
4 for _ in range(m):
5     u, v = map(int, input().split())
6     edges[u].add(v)
7
8 visited = [-1]*n
9 def dfs(i):
10     visited[i] = 0
11     for vertex in edges[i]:
12         if visited[vertex] == -1:
13             if dfs(vertex):
14                 return True
15         elif visited[vertex] == 0:
16             return True
17     visited[i] = 1
18     return False
19
20 for i in range(n):
21     if visited[i] == -1:
22         if dfs(i):

```

The submission result is '完美通过' (Perfectly Passed) with a runtime of 0 ms. The interface also shows the problem description, input/output format, and a table of submission results.

## M05443:兔子与樱花

Dijkstra, <http://cs101.openjudge.cn/practice/05443/>

思路:

直接套用dijkstra的模板即可。

代码:

```

import heapq

p = int(input())
paths = dict()
for _ in range(p):
    paths[input()] = dict()
q = int(input())
for _ in range(q):
    u, v, c = input().split()
    c = int(c)
    paths[u][v] = c
    paths[v][u] = c

```

```

r = int(input())
for _ in range(r):
    a, b = input().split()
    visited = {a: 'end'}
    def dijkstra():
        q = [(0, a)]
        distance = dict()
        for place in paths:
            distance[place] = float('inf')
        distance[a] = 0
        while q:
            c_distance, c_place = heapq.heappop(q)
            if c_place == b:
                return
            for neighbor in paths[c_place]:
                if distance[neighbor] > c_distance + paths[c_place][neighbor]:
                    distance[neighbor] = c_distance + paths[c_place][neighbor]
                    heapq.heappush(q, (distance[neighbor], neighbor))
                    visited[neighbor] = c_place
    dijkstra()
    ans = []
    while visited[b] != 'end':
        ans.append(b)
        ans.append('(' + str(paths[b][visited[b]]) + ')')
        b = visited[b]
    ans.append(b)
    ans.reverse()
    print('->'.join(ans))

```

代码运行截图 (至少包含有"Accepted")

### #49049815提交状态

查看 提交 统计 提问

### 状态: Accepted

源代码

```

import heapq

p = int(input())
paths = dict()
for _ in range(p):
    paths[input()] = dict()
q = int(input())
for _ in range(q):
    u, v, c = input().split()
    c = int(c)
    paths[u][v] = c
    paths[v][u] = c
r = int(input())
for _ in range(r):
    a, b = input().split()
    visited = {a: 'end'}
    def dijkstra():
        q = [(0, a)]

```

#### 基本信息

#: 49049815  
 题目: 05443  
 提交人: 24n2400011498  
 内存: 3700kB  
 时间: 22ms  
 语言: Python3  
 提交时间: 2025-05-02 20:37:06

## T28050: 骑士周游

dfs, <http://cs101.openjudge.cn/practice/28050/>

思路:

使用Warnsdorff's Rule会快不少, 但是感觉也不是一下子就能想到。。。

代码:

```
import sys

sys.setrecursionlimit(1 << 30)
n = int(input())
a, b = map(int, input().split())
visited = [[False]*n for _ in range(n)]
visited[a][b] = True

def get_degree(x, y):
    cnt = 0
    moves = [(1, 2), (1, -2), (-1, 2), (-1, -2), (2, 1), (2, -1), (-2, 1), (-2, -1)]
    for dx, dy in moves:
        nx = x + dx
        ny = y + dy
        if 0 <= nx < n and 0 <= ny < n and not visited[nx][ny]:
            cnt += 1
    return cnt

def dfs(x, y, temp):
    if temp == n*n:
        return True
    moves = [(1, 2), (1, -2), (-1, 2), (-1, -2), (2, 1), (2, -1), (-2, 1), (-2, -1)]
    n_step = []
    for dx, dy in moves:
        nx = x + dx
        ny = y + dy
        if 0 <= nx < n and 0 <= ny < n and not visited[nx][ny]:
            n_step.append((get_degree(nx, ny), nx, ny))
    n_step.sort()
    for _, nx, ny in n_step:
        visited[nx][ny] = True
        temp += 1
        if dfs(nx, ny, temp):
            return True
        temp -= 1
        visited[nx][ny] = False

if dfs(a, b, 1):
    print('success')
else:
    print('fail')
```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

```
import sys

sys.setrecursionlimit(1 << 30)
n = int(input())
a, b = map(int, input().split())
visited = [[False]*n for _ in range(n)]
visited[a][b] = True

def get_degree(x, y):
    cnt = 0
    moves = [(1, 2), (1, -2), (-1, 2), (-1, -2), (2, 1), (2, -1), (-2, 1), (-2, -1)]
    for dx, dy in moves:
        nx = x + dx
        ny = y + dy
        if 0 <= nx < n and 0 <= ny < n and not visited[nx][ny]:
            cnt += 1
            visited[nx][ny] = True
    return cnt

print(get_degree(a, b))
```

基本信息

```
#: 49049857
题目: 28050
提交人: 24n2400011498
内存: 3880kB
时间: 28ms
语言: Python3
提交时间: 2025-05-02 20:46:47
```

## 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。

做题的感受都写在题目中了。

感觉小北探索的模型似乎不是很支持换行？（不知道是仅仅看不到还是不支持）利用了知识库中的检索，感觉确实一定程度上提高了研究题解的效率。

```
parent = [i for i in range(n)]
def find_children(x):
    if x >= n-1:
        return y = x+1 if nums[y] - nums[x] > maxDiff else find_children(y)
    return y if nums[y]-nums[x] <= maxDiff else parent[x]
y = parent[x] + 1
else:
    find_children(y-1)
return find_children(0)
```

这段代码我感觉写的不好看，帮我优化一下吧

 小北

DeepSeek-V3(火山API)上下文64K

这段代码的主要功能是通过递归和循环来构建一个父子关系数组 `parent`，其中 `parent[y]` 表示 `y` 的父节点是 `x`，条件是 `nums[y] - nums[x] <= maxDiff`。代码的逻辑是正确的，但可以通过以下方式进行优化，使其更加简洁和易读：

开始新对话

知识库 即 数算计概题解\_GMyhif cs101\_题解重切片 数算课件\_GMyhif 计概课件\_GMyhif