# Assignment #B: 图为主

Updated 2223 GMT+8 Apr 29, 2025

2025 spring, Complied by 李振硕、信息管理系

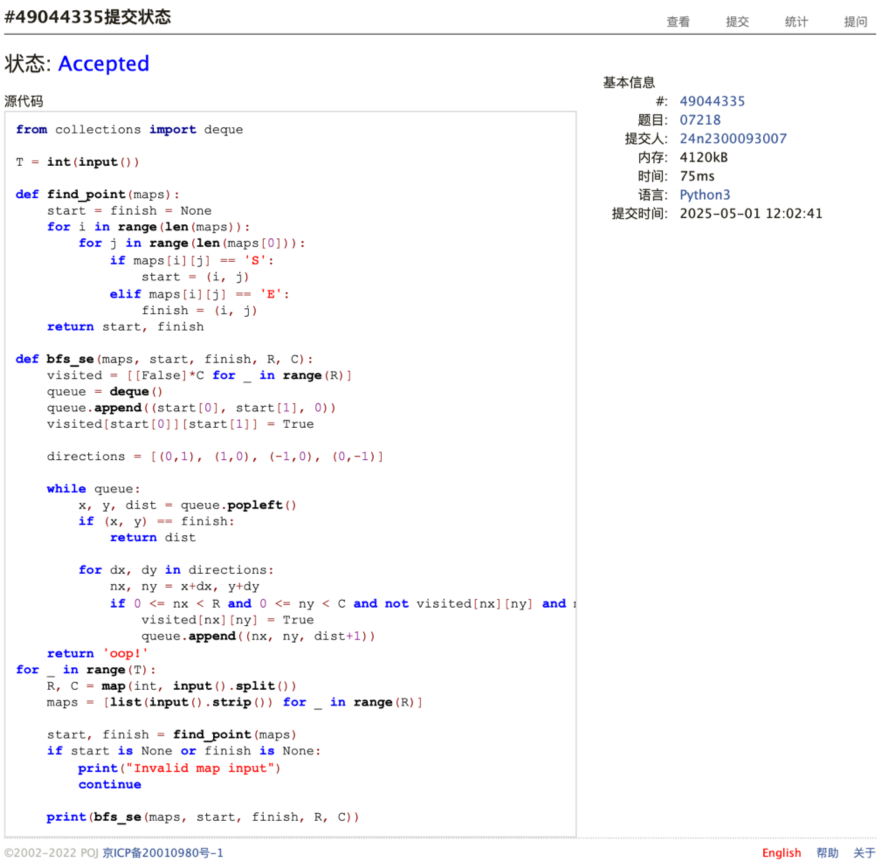
## 1. 题目

### E07218:献给阿尔吉侬的花束

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

思路:

代码:



#49044335提交状态

状态: Accepted

源代码

```python
from collections import deque

T = int(input())

def find_point(maps):
    start = finish = None
    for i in range(len(maps)):
        for j in range(len(maps[0])):
            if maps[i][j] == 'S':
                start = (i, j)
            elif maps[i][j] == 'E':
                finish = (i, j)
    return start, finish

def bfs_se(maps, start, finish, R, C):
    visited = [[False]*C for _ in range(R)]
    queue = deque()
    queue.append((start[0], start[1], 0))
    visited[start[0]][start[1]] = True

    directions = [(0,1), (1,0), (-1,0), (0,-1)]

    while queue:
        x, y, dist = queue.popleft()
        if (x, y) == finish:
            return dist

        for dx, dy in directions:
            nx, ny = x+dx, y+dy
            if 0 <= nx < R and 0 <= ny < C and not visited[nx][ny] and
                visited[nx][ny] = True
                queue.append((nx, ny, dist+1))
    return 'oop!'
for _ in range(T):
    R, C = map(int, input().split())
    maps = [list(input().strip()) for _ in range(R)]

    start, finish = find_point(maps)
    if start is None or finish is None:
        print("Invalid map input")
        continue

    print(bfs_se(maps, start, finish, R, C))
```

基本信息

#: 49044335
题目: 07218
提交人: 24n2300093007
内存: 4120kB
时间: 75ms
语言: Python3
提交时间: 2025-05-01 12:02:41

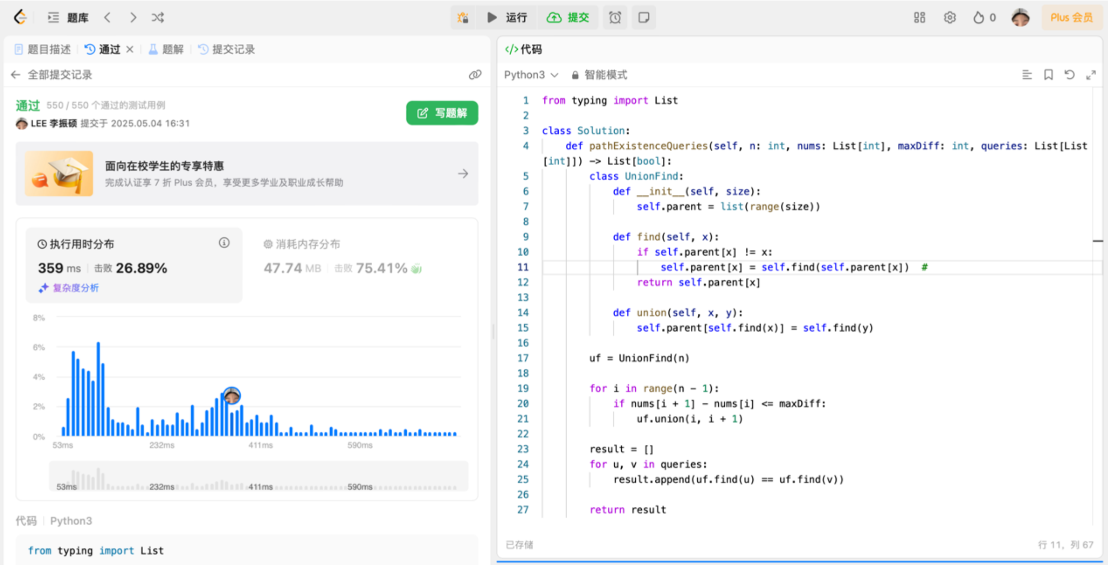
©2002-2022 POJ 京ICP备20010980号-1

### 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:

代码：



通过 550 / 550 个通过的测试用例

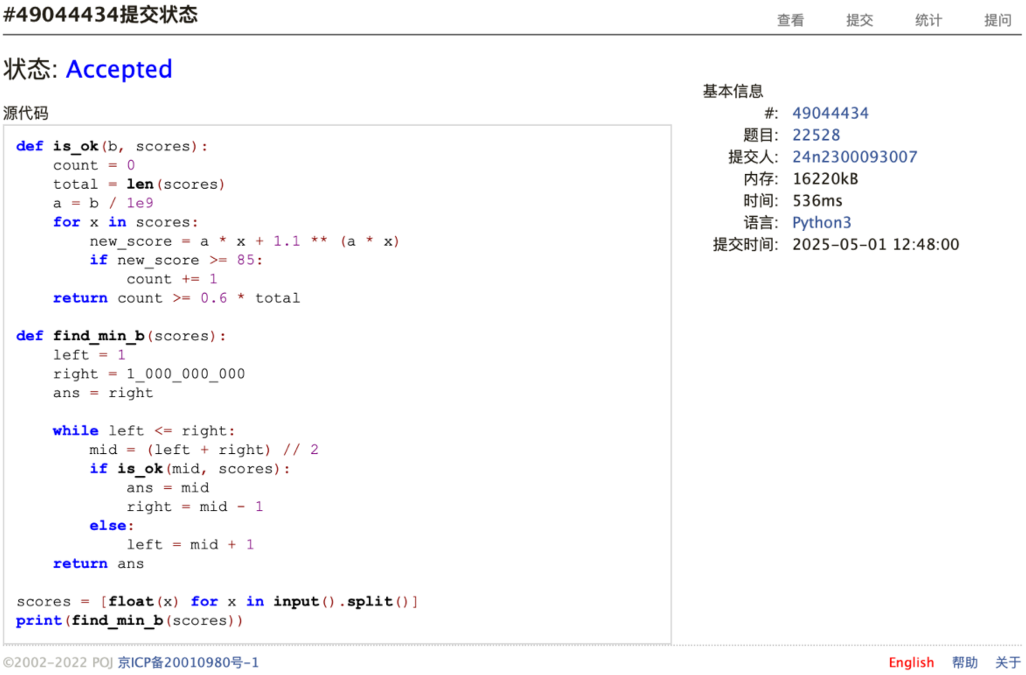
LEE 李振硕 提交于 2025.05.04 16:31

359 ms | 击败 26.89%

47.74 MB | 击败 75.41%

```python
from typing import List

class Solution:
    def pathExistenceQueries(self, n: int, nums: List[int], maxDiff: int, queries: List[List[int]]) -> List[bool]:
        class UnionFind:
            def __init__(self, size):
                self.parent = list(range(size))

            def find(self, x):
                if self.parent[x] != x:
                    self.parent[x] = self.find(self.parent[x])  #
                return self.parent[x]

            def union(self, x, y):
                self.parent[self.find(x)] = self.find(y)

        uf = UnionFind(n)

        for i in range(n - 1):
            if nums[i + 1] - nums[i] <= maxDiff:
                uf.union(i, i + 1)

        result = []
        for u, v in queries:
            result.append(uf.find(u) == uf.find(v))

        return result
```

### M22528:厚道的调分方法
binary search, http://cs101.openjudge.cn/practice/22528/
思路：
代码：



#49044434提交状态

状态: Accepted

基本信息

#:	49044434
题目:	22528
提交人:	24n2300093007
内存:	16220kB
时间:	536ms
语言:	Python3
提交时间:	2025-05-01 12:48:00

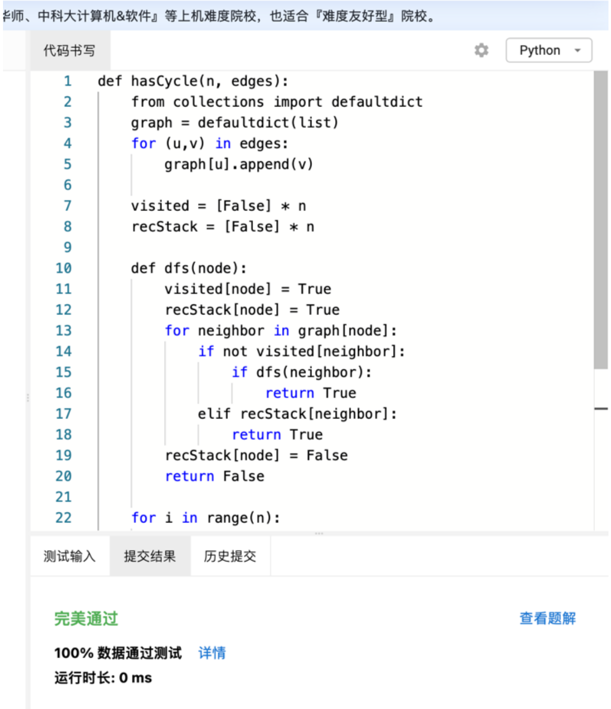
源代码

```python
def is_ok(b, scores):
    count = 0
    total = len(scores)
    a = b / 1e9
    for x in scores:
        new_score = a * x + 1.1 ** (a * x)
        if new_score >= 85:
            count += 1
    return count >= 0.6 * total

def find_min_b(scores):
    left = 1
    right = 1_000_000_000
    ans = right

    while left <= right:
        mid = (left + right) // 2
        if is_ok(mid, scores):
            ans = mid
            right = mid - 1
        else:
            left = mid + 1
    return ans

scores = [float(x) for x in input().split()]
print(find_min_b(scores))
```

©2002-2022 POJ 京ICP备20010980号-1

### Msy382: 有向图判环
dfs, https://sunnywhy.com/sfbj/10/3/382
思路：

代码：

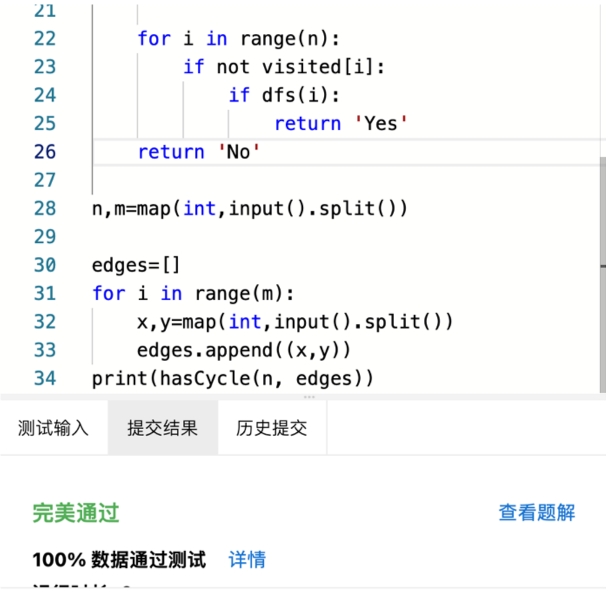


师、中科大计算机&软件』等上机难度院校，也适合『难度友好型』院校。

```python
def hasCycle(n, edges):
    from collections import defaultdict
    graph = defaultdict(list)
    for (u,v) in edges:
        graph[u].append(v)

    visited = [False] * n
    recStack = [False] * n

    def dfs(node):
        visited[node] = True
        recStack[node] = True
        for neighbor in graph[node]:
            if not visited[neighbor]:
                if dfs(neighbor):
                    return True
            elif recStack[neighbor]:
                return True
        recStack[node] = False
        return False

    for i in range(n):
```

100% 数据通过测试 详情

运行时长: 0 ms



```python

    for i in range(n):
        if not visited[i]:
            if dfs(i):
                return 'Yes'
    return 'No'

n,m=map(int,input().split())

edges=[]
for i in range(m):
    x,y=map(int,input().split())
    edges.append((x,y))
print(hasCycle(n, edges))
```

100% 数据通过测试 详情

### M05443:兔子与樱花

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

思路：

代码：

#49044685提交状态

查看 提交 统计 提问

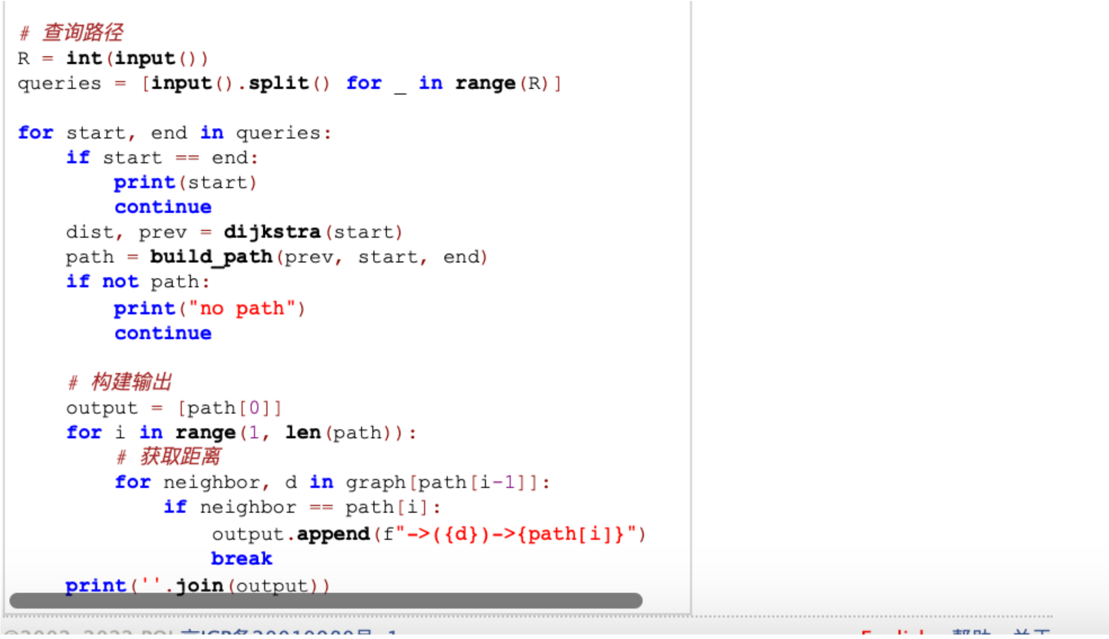
状态: Accepted

基本信息

#:	49044685
题目:	05443
提交人:	24n2300093007
内存:	3748kB
时间:	19ms
语言:	Python3
提交时间:	2025-05-01 14:02:36

源代码

```python
import heapq
from collections import defaultdict

# 输入地点
P = int(input())
places = []
place_index = dict()

for i in range(P):
    name = input().strip()
    places.append(name)
    place_index[name] = i

# 构建图（邻接表）
Q = int(input())
graph = defaultdict(list)

for _ in range(Q):
    a, b, d = input().split()
    d = int(d)
    graph[a].append((b, d))
    graph[b].append((a, d))  # 无向图

# Dijkstra 算法
def dijkstra(start):
    dist = {name: float('inf') for name in places}
    prev = {name: None for name in places}
    dist[start] = 0
    heap = [(0, start)]

    while heap:
        cur_dist, node = heapq.heappop(heap)
        if cur_dist > dist[node]:
            continue
        for neighbor, d in graph[node]:
            new_dist = cur_dist + d
            if new_dist < dist[neighbor]:
                dist[neighbor] = new_dist
                prev[neighbor] = node
                heapq.heappush(heap, (new_dist, neighbo
    return dist, prev

# 回溯路径
def build_path(prev, start, end):
    if start == end:
        return [start]
    path = []
    node = end
    while node != start:
        path.append(node)
        node = prev[node]
        if node is None:
            return []  # no path
    path.append(start)
    path.reverse()
    return path
```



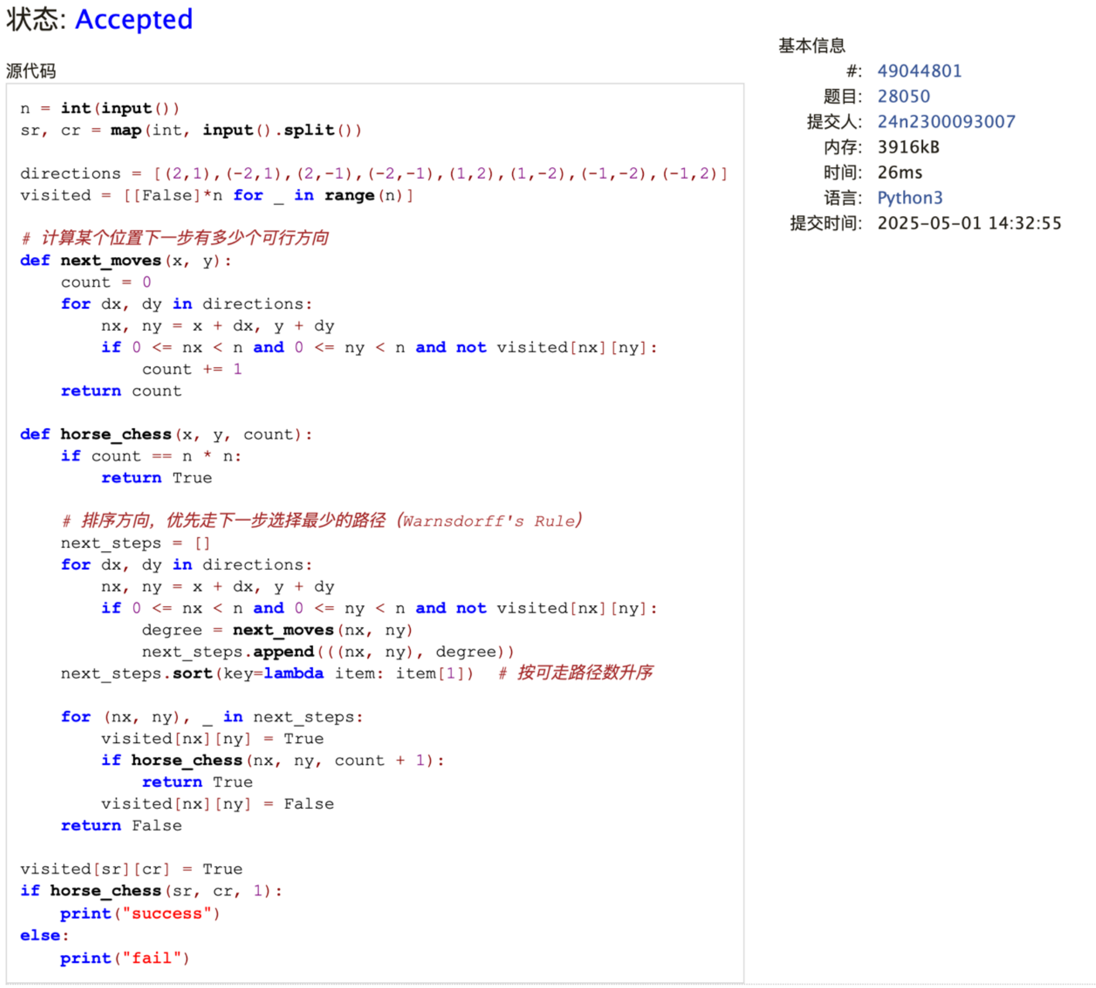
```
# 查询路径
R = int(input())
queries = [input().split() for _ in range(R)]

for start, end in queries:
    if start == end:
        print(start)
        continue
    dist, prev = dijkstra(start)
    path = build_path(prev, start, end)
    if not path:
        print("no path")
        continue

    # 构建输出
    output = [path[0]]
    for i in range(1, len(path)):
        # 获取距离
        for neighbor, d in graph[path[i-1]]:
            if neighbor == path[i]:
                output.append(f"->({d})->{path[i]}")
                break
    print(''.join(output))
```

### T28050: 骑士周游

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

思路:

代码:



状态: Accepted

源代码

```
n = int(input())
sr, cr = map(int, input().split())

directions = [(2,1),(-2,1),(2,-1),(-2,-1),(1,2),(1,-2),(-1,-2),(-1,2)]
visited = [[False]*n for _ in range(n)]

# 计算某个位置下一步有多少个可行方向
def next_moves(x, y):
    count = 0
    for dx, dy in directions:
        nx, ny = x + dx, y + dy
        if 0 <= nx < n and 0 <= ny < n and not visited[nx][ny]:
            count += 1
    return count

def horse_chess(x, y, count):
    if count == n * n:
        return True

    # 排序方向，优先走下一步选择最少的路径（Warnsdorff's Rule）
    next_steps = []
    for dx, dy in directions:
        nx, ny = x + dx, y + dy
        if 0 <= nx < n and 0 <= ny < n and not visited[nx][ny]:
            degree = next_moves(nx, ny)
            next_steps.append(((nx, ny), degree))
    next_steps.sort(key=lambda item: item[1])  # 按可走路径数升序

    for (nx, ny), _ in next_steps:
        visited[nx][ny] = True
        if horse_chess(nx, ny, count + 1):
            return True
        visited[nx][ny] = False
    return False

visited[sr][cr] = True
if horse_chess(sr, cr, 1):
    print("success")
else:
    print("fail")
```

基本信息

#:	49044801
题目:	28050
提交人:	24n2300093007
内存:	3916kB
时间:	26ms
语言:	Python3
提交时间:	2025-05-01 14:32:55

## 2．学习总结和收获、

去年学的 dfs 和 bfs 相关知识都忘记了。这次写作业的时候参考了之前写过的 cheetsheat。Binary search 部分还是不熟悉，感觉需要继续复习多多做题。