1. 题目

E07218:献给阿尔吉侬的花束

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

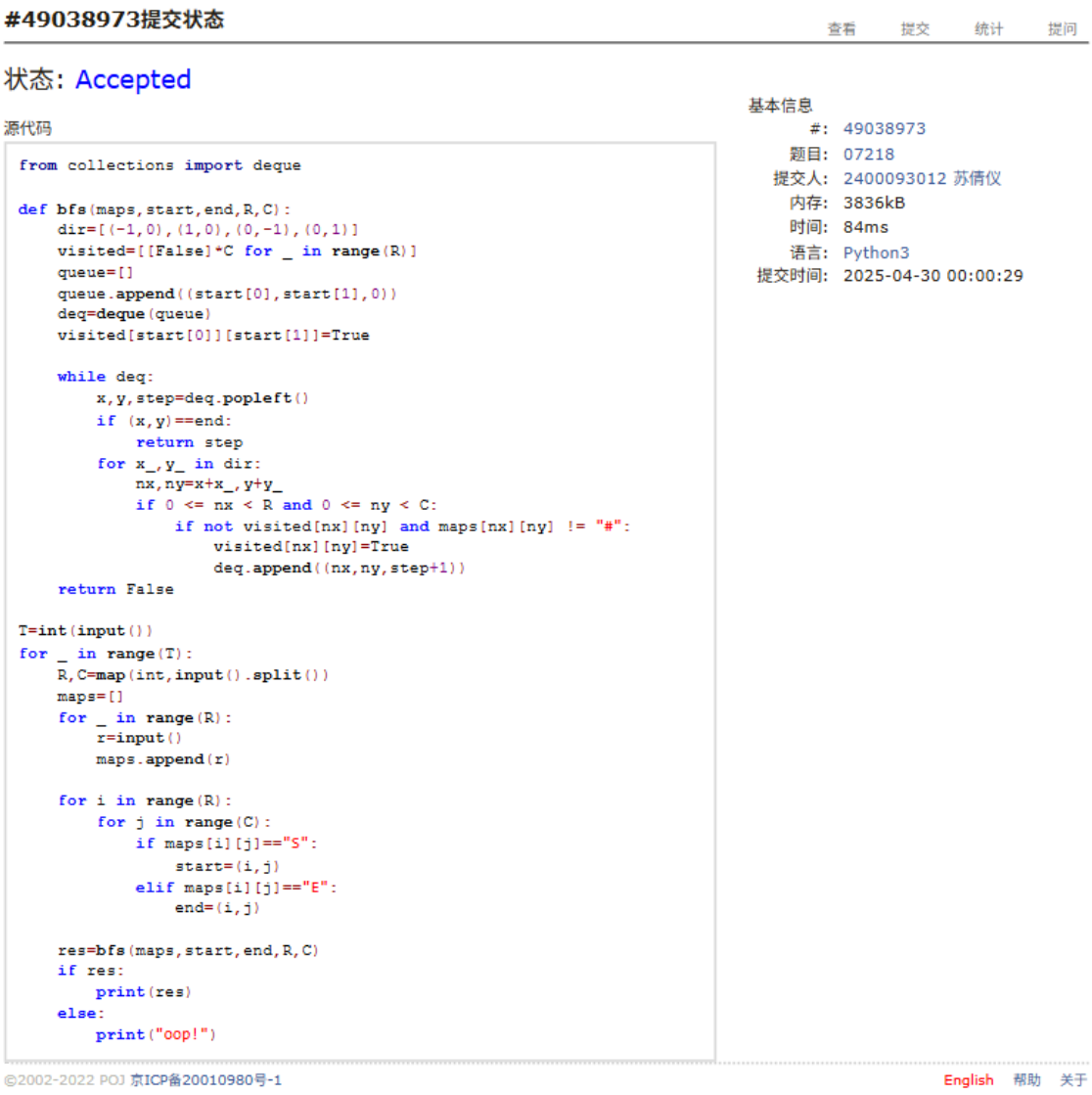
思路：读取 R 行 C 列的迷宫图并找到起点 S 和终点 E 的位置。然后使用 bfs，即从 S 出发，将它加入队列（因为需要频繁进行“入队”和“出队”操作，因此使用队列两端进行插入和删除的操作非常高效），标记为已访问，记录起始步数为 0；每次从队列中取出当前点 (x, y)，尝试走到上下左右位置；若新位置为 E，立即返回当前步数 + 1，若行动合法，则继续加入队列并标记，若搜索完都没找到 E，则输出 "oop!"

代码：

```
from collections import deque

def bfs(maps,start,end,R,C):
    dir=[(-1,0),(1,0),(0,-1),(0,1)]
    visited=[[False]*C for _ in range(R)]
    queue=[]
    queue.append((start[0],start[1],0))
    deq=deque(queue)
    visited[start[0]][start[1]]=True

    while deq:
        x,y,step=deq.popleft()
        if (x,y)==end:
            return step
        for x_,y_ in dir:
            nx,ny=x+x_,y+y_
```

```
            if 0 <= nx < R and 0 <= ny < C:
                if not visited[nx][ny] and maps[nx][ny] != "#":
                    visited[nx][ny]=True
                    deq.append((nx,ny,step+1))
    return False


T=int(input())
for _ in range(T):
    R,C=map(int,input().split())
    maps=[]
    for _ in range(R):
        r=input()
        maps.append(r)

    for i in range(R):
        for j in range(C):
            if maps[i][j]=="S":
                start=(i,j)
            elif maps[i][j]=="E":
                end=(i,j)

    res=bfs(maps,start,end,R,C)
    if res:
        print(res)
    else:
        print("oop!")
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



#49038973提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

def bfs(maps,start,end,R,C):
    dir=[(-1,0),(1,0),(0,-1),(0,1)]
    visited=[[False]*C for _ in range(R)]
    queue=[]
    queue.append((start[0],start[1],0))
    deq=deque(queue)
    visited[start[0]][start[1]]=True

    while deq:
        x,y,step=deq.popleft()
        if (x,y)==end:
            return step
        for x_,y_ in dir:
            nx,ny=x+x_,y+y_
            if 0 <= nx < R and 0 <= ny < C:
                if not visited[nx][ny] and maps[nx][ny] != "#":
                    visited[nx][ny]=True
                    deq.append((nx,ny,step+1))
    return False

T=int(input())
for _ in range(T):
    R,C=map(int,input().split())
    maps=[]
    for _ in range(R):
        r=input()
        maps.append(r)

    for i in range(R):
        for j in range(C):
            if maps[i][j]=="S":
                start=(i,j)
            elif maps[i][j]=="E":
                end=(i,j)

    res=bfs(maps,start,end,R,C)
    if res:
        print(res)
    else:
        print("oop!")
```

基本信息

#: 49038973
题目: 07218
提交人: 2400093012 苏倩仪
内存: 3836kB
时间: 84ms
语言: Python3
提交时间: 2025-04-30 00:00:29

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

M3532.针对图的路径存在性查询 I

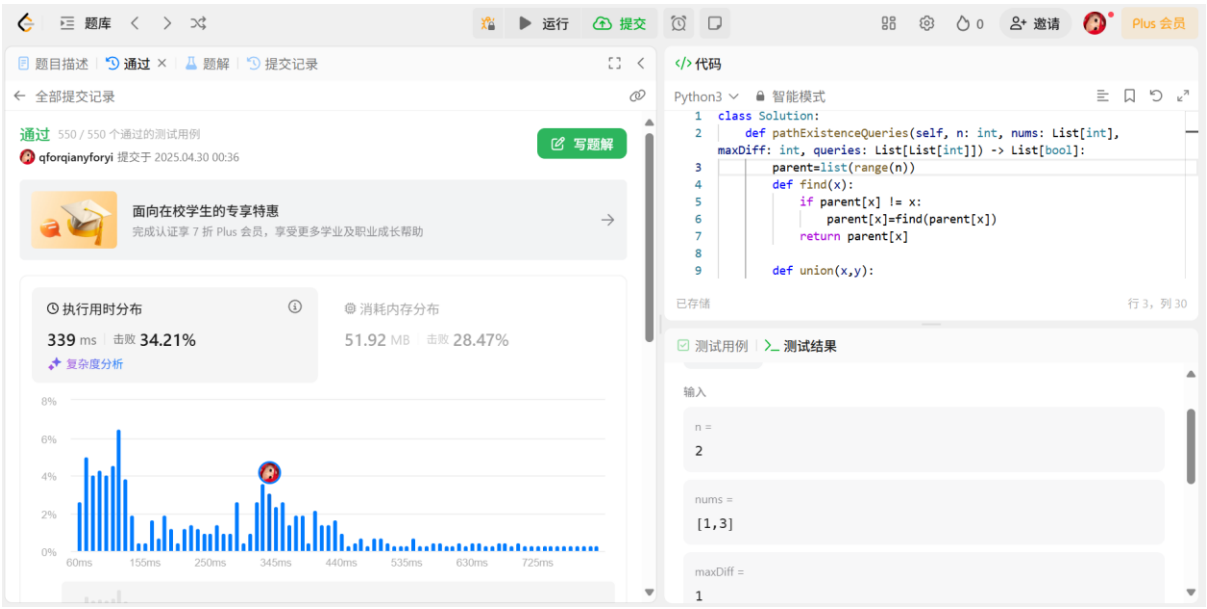
disjoint set, https://leetcode.cn/problems/path-existence-queries-in-a-graph-i/

思路：需要定义两个函数，find 和 union，find 用于找到每个节点的父节点，union 用于将两个父节点合并进同一个集合。首先 parent 列表代表开始时每个节点是一个单独集合，如果两个相邻节点的差值在指定范围内则在他们之间连一条无向边（union 函数），然后对每个 u，v 进行查询，判断是否在同一集合（find 函数），如果他们的父节点一样表示在同一个集合中，返回 True，否则返回 False。

代码：

```
class Solution:
    def pathExistenceQueries(self, n: int, nums: List[int], maxDiff: int, queries: List[List[int]]) -> List[bool]:
        parent=list(range(n))
        def find(x):
            if parent[x] != x:
                parent[x]=find(parent[x])
            return parent[x]

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

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

        res=[]
        for u,v in queries:
            res.append(find(u)==find(v))
        return res
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



M22528:厚道的调分方法

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

思路：初始化二分上下界，每轮二分尝试一个 b 值，算出对应的 a=b / 1e9，即对每个学生调整后的分数，并看有多少人达标；如果人数>=60%，则记录这个 b，并尝试更小的范围，如果人数<60%，说明调得不够，b 要更大，则向右缩小范围，直到二分结束后输出最小满足条件的 b。

代码：

```
score=list(map(float,input().split()))

left,right=1,1000000000
res=right

while left <= right:
    mid=(left+right)//2
```

```
    a=mid/1e9
    count=0
    for i in score:
        ax=a*i
        new_score=ax+1.1**ax
        if new_score >= 85:
            count+=1

    if count >= len(score) * 0.6:
        res=mid
        right=mid-1
    else:
        left=mid+1

print(res)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49039018提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
score=list(map(float,input().split()))

left,right=1,1000000000
res=right

while left <= right:
    mid=(left+right)//2
    a=mid/1e9
    count=0
    for i in score:
        ax=a*i
        new_score=ax+1.1**ax
        if new_score >= 85:
            count+=1

    if count >= len(score) * 0.6:
        res=mid
        right=mid-1
    else:
        left=mid+1

print(res)
```

基本信息

#: 49039018
题目: 22528
提交人: 2400093012 苏倩仪
内存: 16276kB
时间: 896ms
语言: Python3
提交时间: 2025-04-30 00:51:43

English 帮助 关于

Msy382: 有向图判环

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

思路：用一个列表 graph 来建图，将每个 u 对应到 v，然后使用 visited 来定义访问状态（未访问=0，访问中=1，访问完毕=2），并用递归来遍历每组 uv 检查图是否连通（每次将节点标记为正在访问，同时检查 u 对应的 v，如果未访问则递归对其进行访问，如果发现环（回到==1 仍正在访问的节点）则将 cycle 标记为 True，否则标记访问完毕=2），最后如果检查到 cycle 为 True 则代表有换，输出 yes，否则输出 no。

代码：

```
n,m=map(int,input().split())
graph=[[] for _ in range(n)]

for _ in range(m):
```

```
    u,v=map(int,input().split())
    graph[u].append(v)

def dfs(u):
    global cycle
    visited[u]=1 # 正在访问
    for v in graph[u]:
        if visited[v]==0: # 未访问
            dfs(v)
        elif visited[v]==1:
            cycle=True
    visited[u]=2 # 访问完成

visited=[0]*n
cycle=False

for i in range(n):
    if visited[i]==0:
        dfs(i)

# print(graph)
if cycle:
    print("Yes")
else:
    print("No")
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M05443:兔子与樱花

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

思路：用一个 graph 字典来建图，各点之间是无向边。然后使用 Dijkstra 算法求两个地点之间的最短路径和距离，将起点的距离设为 0，然后使用最小堆来储存当前距离最短的节点，并每次取出距离最短的节点进行处理，每次遍历更新当前节点的相邻节点的最短路径，如果发现通过当前节点到达某个相邻节点比原来的距离更短，则更新该邻居的距离，当 heap 为空说明已达到终点（如果起点和终点相同，直接输出起点），最后按照指定格式输出最短距离和路径。

代码：

```
import heapq

def dijkstra(graph,start,end):
    heap=[(0,start,[])] # 当前距离,当前点,当前路径
```

```
    while heap:
        dist,node,path=heapq.heappop(heap)
        path=path+[node]

        if node == end:
            return dist,path

        for place,dis in graph.get(node,[]):
            heapq.heappush(heap,(dist+dis,place,path))
    return False

P=int(input())
places=[input() for _ in range(P)]

Q=int(input())
graph={}
for _ in range(Q):
    a,b,d=input().split()
    d=int(d)
    if a not in graph:
        graph[a]=[]
    if b not in graph:
        graph[b]=[]
    graph[a].append((b,d))
    graph[b].append((a,d))

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

```
for start,end in queries:
    if start == end:
        print(start)
    else:
        distance,path=dijkstra(graph,start,end)
        if path:
            res=path[0]
            for i in range(1,len(path)):
                pre,cur=path[i-1],path[i]
                for place,dis in graph[pre]:
                    if place == cur:
                        res += f"->({dis})->{cur}"
                        break
            print(res)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49040301提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 49040301
题目: 05443
提交人: 2400093012 苏倩仪
内存: 3660kB
时间: 22ms
语言: Python3
提交时间: 2025-04-30 14:36:45

源代码

```
import heapq

def dijkstra(graph,start,end):
    heap=[(0,start,[])] # 当前距离,当前点,当前路径

    while heap:
        dist,node,path=heapq.heappop(heap)
        path=path+[node]

        if node == end:
            return dist,path

        for place,dis in graph.get(node,[]):
            heapq.heappush(heap,(dist+dis,place,path))
    return False

P=int(input())
places=[input() for _ in range(P)]

Q=int(input())
graph={}
for _ in range(Q):
    a,b,d=input().split()
    d=int(d)
    if a not in graph:
        graph[a]=[]
    if b not in graph:
        graph[b]=[]
    graph[a].append((b,d))
    graph[b].append((a,d))

R=int(input())
queries=[input().split() for _ in range(R)]

for start,end in queries:
    if start == end:
        print(start)
    else:
        distance,path=dijkstra(graph,start,end)
        if path:
            res=path[0]
            for i in range(1,len(path)):
                pre,cur=path[i-1],path[i]
                for place,dis in graph[pre]:
                    if place == cur:
                        res += f"->({dis})->{cur}"
                        break
            print(res)
```


English 帮助 关于

T28050: 骑士周游

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

思路：暴力递归会导致 TLE，用 Warnsdorff's Rule（优先走下一步出路最少的格子，减少搜索空间，比如：从一个点出发，有 8 个可跳点，优先跳到下一步只有 1~2 个出路的点，而不是选择 8 个出路的点，避免走进死角），定义另一个函数 next 用于计算一

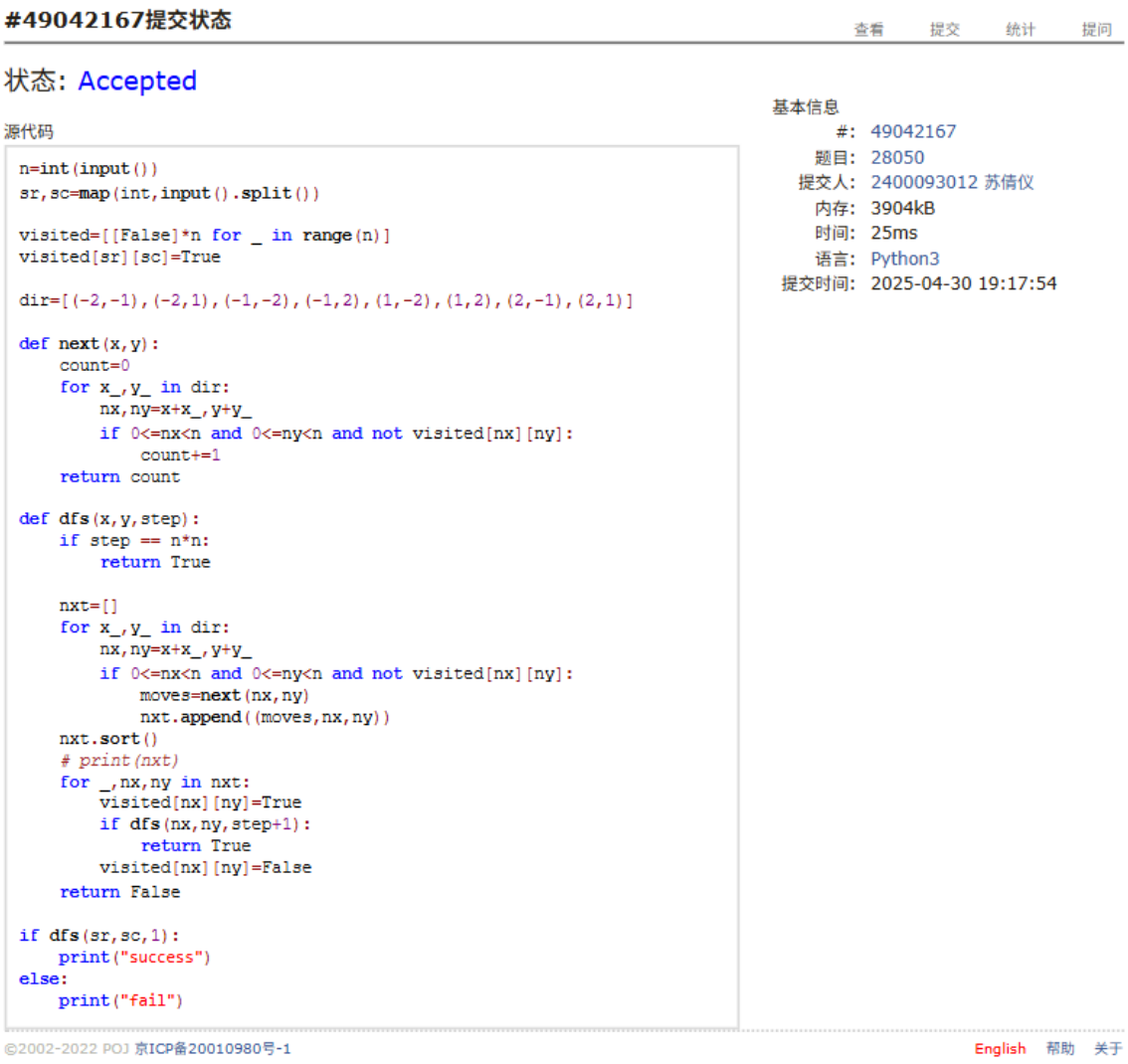
个点有多少出路，并进行排序，并在递归时每次选择这个最少出路的点，对它的可跳点进行标记，如果失败则进行回溯（如果某一步没有合法跳点，就回到上一步换条路继续尝试），当 step==n*n（棋盘大小）则表示走完了，输出 success，否则 fail。

代码：

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

visited=[[False]*n for _ in range(n)]
visited[sr][sc]=True

dir=[(-2,-1),(-2,1),(-1,-2),(-1,2),(1,-2),(1,2),(2,-1),(2,1)]

def next(x,y):
    count=0
    for x_,y_ in dir:
        nx,ny=x+x_,y+y_
        if 0<=nx<n and 0<=ny<n and not visited[nx][ny]:
            count+=1
    return count

def dfs(x,y,step):
    if step == n*n:
        return True

    nxt=[]
    for x_,y_ in dir:
        nx,ny=x+x_,y+y_
```

```
        if 0<=nx<n and 0<=ny<n and not visited[nx][ny]:
            moves=next(nx,ny)
            nxt.append((moves,nx,ny))
    nxt.sort()
    # print(nxt)
    for _,nx,ny in nxt:
        visited[nx][ny]=True
        if dfs(nx,ny,step+1):
            return True
        visited[nx][ny]=False
    return False

if dfs(sr,sc,1):
    print("success")
else:
    print("fail")
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



#49042167提交状态

查看 提交 统计 提问

状态: Accepted

源代码

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

visited=[[False]*n for _ in range(n)]
visited[sr][sc]=True

dir=[(-2,-1),(-2,1),(-1,-2),(-1,2),(1,-2),(1,2),(2,-1),(2,1)]

def next(x,y):
    count=0
    for x_,y_ in dir:
        nx,ny=x+x_,y+y_
        if 0<=nx<n and 0<=ny<n and not visited[nx][ny]:
            count+=1
    return count

def dfs(x,y,step):
    if step == n*n:
        return True

    nxt=[]
    for x_,y_ in dir:
        nx,ny=x+x_,y+y_
        if 0<=nx<n and 0<=ny<n and not visited[nx][ny]:
            moves=next(nx,ny)
            nxt.append((moves,nx,ny))
    nxt.sort()
    # print(nxt)
    for _,nx,ny in nxt:
        visited[nx][ny]=True
        if dfs(nx,ny,step+1):
            return True
        visited[nx][ny]=False
    return False

if dfs(sr,sc,1):
    print("success")
else:
    print("fail")
```

基本信息

#: 49042167
题目: 28050
提交人: 2400093012 苏倩仪
内存: 3904kB
时间: 25ms
语言: Python3
提交时间: 2025-04-30 19:17:54

©2002-2022 POJ 京ICP备20010980号-1

English 帮助 关于

2. 学习总结和收获

<mark>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。</mark>

感觉第一二三题还好，第一题在去年的计概有学过一些，但是都忘了，就当复习了；第二题花了点时间找什么是并查集，也挺好理解的；最近做了很多完全超出理解范围的题，看到第三题的二分查找瞬间好亲切（，虽然写公式的部分写错好几次。第四题一开始没什么思路，上网找了资料学到了“染色法”的概念，理解起来花了好一会时

间，再加上递归对我来说就有点费力了，但是感觉是很好用的方法。第五题用了 AI，但还是看不太明白 Dijkstra 的核心算法是什么，但是用 heap 的方法是很便捷。第六题感觉这类题目做了挺多次，但是一直 get 不到诀窍，这次做的时候豁然开朗结果 TLE... 然后就 AI 了，最后学到了 Warndorff's Rule！