Assignment #B: 图为主

Updated 2223 GMT+8 Apr 29, 2025

2025 spring, Complied by <mark>汤伟杰，信息管理系</mark>

> **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：

纯正的 bfs，把 step 保存到 deque 中。

代码：

```
from collections import deque

def bfs(s,sx,sy,ex,ey):
    q=deque()
    q.append((sx,sy,0))
    inq=set()
    inq.add((sx,sy))

    while q:
        for _ in range(len(q)):
            x,y,step=q.popleft()
            for dx,dy in (0,1),(0,-1),(1,0),(-1,0):
                nx,ny=x+dx,y+dy
                if (nx,ny)==(ex,ey):
                    return step+1
                if 0<=nx<n and 0<=ny<m and s[nx][ny]=='.' and (nx,ny) not in
```

```
inq:
                    q.append((nx,ny,step+1))
                    inq.add((nx,ny))
    return 'oop!'

for _ in range(int(input())):
    n,m=map(int,input().split())
    s=[]
    for i in range(n):
        l=input()
        if 'S' in l:
            sx,sy=i,l.index('S')
        if 'E' in l:
            ex,ey=i,l.index('E')
        s.append(l)
    print(bfs(s,sx,sy,ex,ey))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49039028提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

def bfs(s,sx,sy,ex,ey):
    q=deque()
    q.append((sx,sy,0))
    inq=set()
    inq.add((sx,sy))

    while q:
        for _ in range(len(q)):
            x,y,step=q.popleft()
            for dx,dy in (0,1),(0,-1),(1,0),(-1,0):
                nx,ny=x+dx,y+dy
                if (nx,ny)==(ex,ey):
                    return step+1
                if 0<=nx<n and 0<=ny<m and s[nx][ny]=='.' and (nx,ny) nc
                    q.append((nx,ny,step+1))
                    inq.add((nx,ny))
    return 'oop!'

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

基本信息

#: 49039028
题目: 07218
提交人: 24n2400016635
内存: 5124kB
时间: 92ms
语言: Python3
提交时间: 2025-04-30 01:00:36

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/

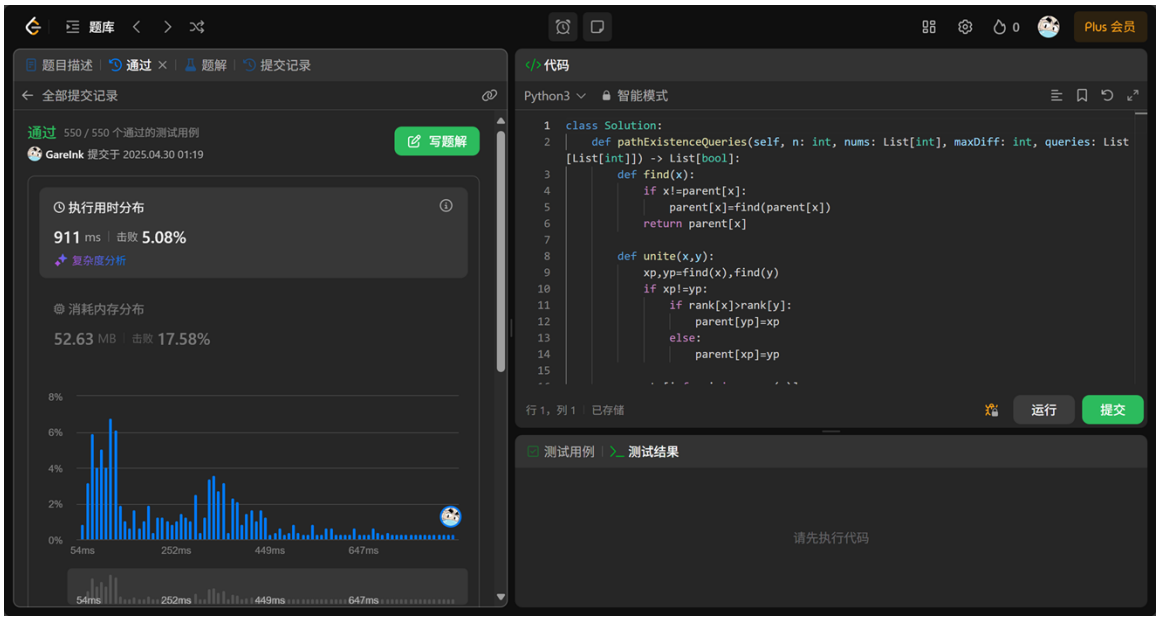
思路：

这道题一开始做超时了，是看了别人的题解才想出来怎么在连接节点的二重循环中优化的，因为数组是已经排好序的，假如 1 和 2、3、4 都已经连上了，就不需要再让 2 和 3、2 和 4、3 和 4 连一遍了，因为对它们用 find 函数之后的根节点已经是相同的了，这样可以维护一个 j 的下界，让遍历的每个 i 对应的要连接的节点能够从靠右的合适位置开始连接，这样能降低一些时间复杂度。

代码：

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

        def unite(x,y):
            xp,yp=find(x),find(y)
            if xp!=yp:
                if rank[x]>rank[y]:
                    parent[yp]=xp
                else:
                    parent[xp]=yp

        parent=[i for i in range(n)]
        rank=[0]*n
        bound=0
        for i in range(n):
            for j in
range(max(bound,i+1),min(n,(upper:=bisect_right(nums,nums[i]+maxDiff)))):
                unite(i,j)
            bound=max(bound,upper)

        ans=[]
        for a,b in queries:
            if find(a)==find(b):
                ans.append(True)
            else:
                ans.append(False)

        return ans
```

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：

二分，代码居然一遍过了好开心。。对 b 进行二分，然后 success 函数的部分只需要提取对成绩倒序排序，检查前 60%的人在调分后是不是都大于等于 85，一旦有小于 85 的就直接返回 False。感觉逻辑还算比较清晰易想的。

代码：

```
from math import ceil
def success(mid,n):
    cnt=ceil(n*0.6)
    a=mid/1000000000
    for i in range(cnt):
        converted=a*s[i]+1.1**(a*s[i])
        if converted<85:
            return False
    return True

s=list(map(float,input().split()))
n=len(s)
s.sort(reverse=True)
left,right=0,1000000000
while left<=right:
    mid=(left+right)//2
    if success(mid,n):
        right=mid-1
    else:
        left=mid+1
print(left)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49039041提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from math import ceil
def success(mid,n):
    cnt=ceil(n*0.6)
    a=mid/1000000000
    for i in range(cnt):
        converted=a*s[i]+1.1**(a*s[i])
        if converted<85:
            return False
    return True

s=list(map(float,input().split()))
n=len(s)
s.sort(reverse=True)
left,right=0,1000000000
while left<=right:
    mid=(left+right)//2
    if success(mid,n):
        right=mid-1
    else:
        left=mid+1
print(left)
```

基本信息

#: 49039041
题目: 22528
提交人: 24n2400016635
内存: 16464kB
时间: 402ms
语言: Python3
提交时间: 2025-04-30 01:37:01

Msy382: 有向图判环

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

思路：

从课件学习的 dfs 方法，0 表示为搜索过，1 表示正在搜索的路上，2 表示之前的路径上已经搜索过这个节点。判环的逻辑就是在每一次 dfs 的时候，遇到新节点置为 1，接下来一旦发现在这条路径上又出现了 1 就说明存在一个闭环。

从力扣学习的 bfs 方法，类似于拓扑排序的思路，记录有向图各个节点的入度，每次把入度为 0 的节点放入 deque（叶子节点），然后扩散到下一层，让下一层的入度减一，如果为 0 了就接着入队列。最后检查是不是所有的节点都经历了一层入队出队，因为一旦有环势必会有一些节点无法入队。

代码：

```
### DFS ###
from collections import defaultdict

def dfs(i,g,s):
    if s[i]==1:
        return True
    if s[i]==2:
        return False
    s[i]=1
    for next in g[i]:
        if dfs(next,g,s):
            return True
    s[i]=2
    return False

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

```
g=defaultdict(list)
for _ in range(m):
    a,b=map(int,input().split())
    g[a].append(b)
s=[0]*n
for i in range(n):
    if dfs(i,g,s):
        print('Yes')
        break
else:
    print('No')

### BFS ###
from collections import deque,defaultdict

n,m=map(int,input().split())
g=defaultdict(list)
indeg=[0]*n
for _ in range(m):
    a,b=map(int,input().split())
    g[a].append(b)
    indeg[b]+=1

q=deque()
for i in range(n):
    if indeg[i]==0:
        q.append(i)

while q:
    for _ in range(len(q)):
        node=q.popleft()
        n-=1
        for next in g[node]:
            indeg[next]-=1
            if indeg[next]==0:
                q.append(next)

print(['No','Yes'][n!=0])
```

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：

复习了一下 dijkstra 的写法，把 bfs 中的 deque 换成 heap 即可，这道题复杂的地方是输入数据的存储：使用地点名称作为节点，距离作为节点之间边的权值；为了维护 dist 数组的索引，额外构建一个字典，保存地点：对应的索引这个键值对，这可以在前 p 行输入中构建（否则前 p 行就没有用了）；为了得到路径，维护一个 prev 的字典（这个在前两周的作业中也有，于是用起来很方便印象比较深），存的是 `curr_node:(prev_node, weight)`，方便从终点回溯到起点。

代码：

```
from heapq import heappop,heappush
from collections import defaultdict

def dijkstra(g,name_id,s,e):
    q=[]
    heappush(q,(s,0))
    dist=[float('inf')]*len(g)
    dist[name_id[s]]=0
    prev={'s':(None,None)}
    while q:
        for _ in range(len(q)):
            curr_node,curr_dist = heappop(q)
            # if curr_node==e:
            #     return get_path(s,e,prev)
            for next,w in g[curr_node]:
                if curr_dist+w>dist[name_id[next]]:
                    continue
                dist[name_id[next]]=min(curr_dist+w, dist[name_id[next]])
                heappush(q,(next,dist[name_id[next]]))
```

```
                prev[next]=(curr_node,w)
    return get_path(s,e,prev)

def get_path(s,e,prev):
    ans=[]
    while e!=s:
        ans.append(e)
        e,w=prev[e]
        ans.append(f'({w})')
    ans.append(e)
    ans.reverse()
    return ans

p=int(input())
g=defaultdict(list)
name_id={}
for id in range(p):
    name=input()
    name_id[name]=id

q=int(input())
for _ in range(q):
    a,b,w=input().split()
    w=int(w)
    g[a].append((b,w))
    g[b].append((a,w))

n=int(input())
for _ in range(n):
    s,e=input().split()
    path=dijkstra(g,name_id,s,e)
    print('->'.join(path))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49040152提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from heapq import heappop,heappush
from collections import defaultdict

def dijkstra(g,name_id,s,e):
    q=[]
    heappush(q,(s,0))
    dist=[float('inf')]*len(g)
    dist[name_id[s]]=0
    prev={'s':(None,None)}
    while q:
        for _ in range(len(q)):
            curr_node,curr_dist = heappop(q)
            # if curr_node==e:
            #     return get_path(s,e,prev)
            for next,w in g[curr_node]:
                if curr_dist+w>dist[name_id[next]]:
                    continue
                dist[name_id[next]]=min(curr_dist+w, dist[name_id[next]])
                heappush(q,(next,dist[name_id[next]]))
                prev[next]=(curr_node,w)
    return get_path(s,e,prev)
```

基本信息

#:	49040152
题目:	05443
提交人:	24n2400016635
内存:	3672kB
时间:	22ms
语言:	Python3
提交时间:	2025-04-30 14:05:11

T28050: 骑士周游

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

思路：

与正常的 dfs 的区别是：每次不是直接进入 8 个方向，而是对 8 个方向的出度排序，从小的开始遍历到大的。同时保证 dfs 的状态可回溯性，保证能 dfs 完全部路径。

代码：

```
dx,dy=[-1,1,-2,2,-2,2,-1,1],[-2,-2,-1,-1,1,1,2,2]

def can_move(n,s,x,y):
    return 0<=x<n and 0<=y<n and s[x][y]==0

def get_degree(x,y):
    cnt=0
    for i in range(8):
        nx,ny=x+dx[i],y+dy[i]
        if can_move(n,s,nx,ny):
            cnt+=1
    return cnt

def dfs(n,r,c,step,s,sr,sc):
    if step==n**2:
        return True
    degrees=[]
    for i in range(8):
        nx,ny=r+dx[i],c+dy[i]
        if (can_move(n,s,nx,ny) or (step==n**2-1 and (nx,ny)==(sr,sc))):
            deg=get_degree(nx,ny)
            degrees.append((deg,nx,ny))

    degrees.sort()
    for _, nx, ny in degrees:
        s[nx][ny]=1
        if dfs(n,nx,ny,step+1,s,sr,sc):
            return True
        s[nx][ny]=0
    return False

n=int(input())
r,c=map(int,input().split())
s=[[0]*n for _ in range(n)]
s[r][c]=1

print(['fail','success'][dfs(n,r,c,1,s,r,c)])
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

#49035799提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 49035799
题目: 28050
提交人: 24n2400016635
内存: 3852kB
时间: 34ms
语言: Python3
提交时间: 2025-04-29 16:50:53

源代码

```python
dx,dy=[-1,1,-2,2,-2,2,-1,1],[-2,-2,-1,-1,1,1,2,2]

def can_move(n,s,x,y):
    return 0<=x<n and 0<=y<n and s[x][y]==0

def get_degree(x,y):
    cnt=0
    for i in range(8):
        nx,ny=x+dx[i],y+dy[i]
        if can_move(n,s,nx,ny):
            cnt+=1
    return cnt

def dfs(n,r,c,step,s,sr,sc):
    if step==n**2:
        return True
    degrees=[]
    for i in range(8):
        nx,ny=r+dx[i],c+dy[i]
        if (can_move(n,s,nx,ny) or (step==n**2-1 and (nx,ny)==(sr,sc))):
            deg=get_degree(nx,ny)
            degrees.append((deg,nx,ny))
```

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>

力扣的竞赛忘做了，在做 cf 的，最近的一次 div2 居然 ac 了 4 题，里面有一个欧拉筛的题目，又是欧拉筛宣传片级别的题(https://codeforces.com/contest/2104/problem/D)，这样才保证没有掉分，保住了绿名哈哈。。

课件还在学习，每日选做还有一部分落下了在跟进！