

Assignment #D: May月考

Updated 1654 GMT+8 May 8, 2024

2024 spring, Compiled by 同学的姓名、院系

说明:

- 1) 请把每个题目解题思路 (可选), 源码Python, 或者C++ (已经在Codeforces/Openjudge上AC), 截图 (包含Accepted), 填写到下面作业模版中 (推荐使用 typora <https://typoraio.cn>, 或者用 word)。AC 或者没有AC, 都请标上每个题目大致花费时间。
- 2) 提交时候先提交pdf文件, 再把md或者doc文件上传到右侧“作业评论”。Canvas需要有同学清晰头像、提交文件有pdf、“作业评论”区有上传的md或者doc附件。
- 3) 如果不能在截止前提交作业, 请写明原因。

编程环境

(请改为同学的操作系统、编程环境等)

操作系统: macOS Ventura 13.4.1 (c)

Python编程环境: Spyder IDE 5.2.2, PyCharm 2023.1.4 (Professional Edition)

C/C++编程环境: Mac terminal vi (version 9.0.1424), g++/gcc (Apple clang version 14.0.3, clang-1403.0.22.14.1)

1. 题目

02808: 校门外的树

<http://cs101.openjudge.cn/practice/02808/>

思路:

送分

代码

```

L,M=map(int,input().split())
data=[True]*(L+1)
for _ in range(M):
    b,e=map(int,input().split())
    for i in range(b,e+1):
        if data[i]:
            data[i]=False
cnt=0
for x in data:
    if x:
        cnt+=1
print(cnt)

```

代码运行截图 (至少包含有"Accepted")

#44900502提交状态

查看

状态: Accepted

基本信息

源代码

```

L,M=map(int,input().split())
data=[True]*(L+1)
for _ in range(M):
    b,e=map(int,input().split())
    for i in range(b,e+1):
        if data[i]:
            data[i]=False
cnt=0
for x in data:
    if x:
        cnt+=1
print(cnt)

```

#: 449005

题目: 02808

提交人: zxk

内存: 3652kB

时间: 42ms

语言: Python3

提交时间: 2024-01

20449: 是否被5整除

<http://cs101.openjudge.cn/practice/20449/>

思路:

送分

代码

```

def trans(xlst):
    data=xlst
    data.reverse()
    p=0
    ans=0
    for i in data:
        ans+=int(i)*2**p
        p+=1
    if ans%5==0:
        return 1

```

```
else:
    return 0

data=str(input())
stack=[]
for x in data:
    stack.append(x)
anslst=[]
for i in range(1,len(stack)+1):
    origin=stack[:i]
    anslst.append(str(trans(origin)))
print(''.join(anslst))
```

代码运行截图 (至少包含有"Accepted")

OpenJudge

题目ID, 标题, 描述



CS101 / 题库

题目

排名

状态

提问

#44900745提交状态

查看 损

状态: Accepted

基本信息

#: 44900745
题目: 20449
提交人: zxk
内存: 3600kB
时间: 19ms
语言: Python3
提交时间: 2024-05

源代码

```
def trans(xlst):
    data=xlst
    data.reverse()
    p=0
    ans=0
    for i in data:
        ans+=int(i)*2**p
        p+=1
    if ans%5==0:
        return 1
    else:
        return 0

data=str(input())
stack=[]
for x in data:
    stack.append(x)
anslst=[]
for i in range(1,len(stack)+1):
    origin=stack[:i]
    anslst.append(str(trans(origin)))
print(''.join(anslst))
```

01258: Agri-Net

<http://cs101.openjudge.cn/practice/01258/>

思路:

并查集, Kruskal

代码

```
class DisjointsetUnion:#定义并查集类
    def __init__(self,n):
        self.parent=list(range(n))
        self.rank=[0]*n

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

    def union(self,x,y):#按秩合并
        xr=self.find(x)
        yr=self.find(y)
        if xr==yr:
            return False
        elif self.rank[xr]<self.rank[yr]:
            self.parent[xr]=yr
        elif self.rank[xr]>self.rank[yr]:
            self.parent[yr]=xr
        else:
            self.parent[yr]=xr
            self.rank[xr]+=1
        return True

def kruskal(n,edges):
    dsu=DisjointsetUnion(n)
    mst_weight=0
    for weight,u,v in sorted(edges):
        if dsu.union(u,v):
            mst_weight+=weight
    return mst_weight

while True:
    try:
        N=int(input())
        matrix=[]
        for _ in range(N):
            matrix.append([int(x) for x in input().split()])

        edges=[]
        for i in range(N):
            row=matrix[i]
            for j in range(N):
                if row[j]!=0:
```


```

edges.append((row[j],i,j))

print(Kruskal(N,edges))
except EOFError:
    break

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")


CS101 / 题库

[题目](#)
[排名](#)
[状态](#)
[提问](#)

#44961339提交状态

[查看](#)

状态: Accepted

源代码

```

class DisjointsetUnion: #定义并查集类
    def __init__(self,n):
        self.parent=list(range(n))
        self.rank=[0]*n

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

    def union(self,x,y): #按秩合并
        xr=self.find(x)
        yr=self.find(y)
        if xr==yr:
            return False
        elif self.rank[xr]<self.rank[yr]:
            self.parent[xr]=yr
        elif self.rank[xr]>self.rank[yr]:
            self.parent[yr]=xr
        else:
            self.parent[yr]=xr
            self.rank[xr]+=1
        return True

    def Kruskal(n,edges):
        dsu=DisjointsetUnion(n)
        mst_weight=0
        for weight,u,v in sorted(edges):
            if dsu.union(u,v):
                mst_weight+=weight
        return mst_weight

while True:

```

基本信息

#: 44961
 题目: 01258
 提交人: zxx
 内存: 5032k
 时间: 64ms
 语言: Python
 提交时间: 2024-

27635: 判断无向图是否连通有无回路(同23163)

<http://cs101.openjudge.cn/practice/27635/>

思路:

判断连通用的bfs, 有环用的dfs, dfs中的技巧还是有一些的, 记录了路径和上一个访问的节点, 但是学到了如果是连续数字编号的节点可以用visited=[False]*n来记录是否访问, 会快一些

```

class Vertex:
    def __init__(self,v):
        self.value=v
        self.connectedto={}

class Graph:
    def __init__(self):
        self.vertexes={}

    def add_vertex(self,s):
        self.vertexes[s]=Vertex(s)

    def add_edge(self,s1,s2,w):
        if s1 not in self.vertexes:
            self.vertexes[s1]=Vertex(s1)
        if s2 not in self.vertexes:
            self.vertexes[s2]=Vertex(s2)
        v1=self.vertexes[s1]
        v2=self.vertexes[s2]
        v1.connectedto[s2]=w
        v2.connectedto[s1]=w
        self.vertexes[s1]=v1
        self.vertexes[s2]=v2

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

for _ in range(m):
    u,v=map(int,input().split())
    g.add_edge(u,v,1)

def connected(graph):
    if n==1:
        return 'connected:yes'
    visited=[False for i in range(n)]
    queue=[0]
    while queue:
        ver=queue.pop(0)
        neighbours=graph.vertexes[ver].connectedto.keys()
        for neighbour in neighbours:
            if not visited[neighbour]:
                visited[neighbour]=True
                queue.append(neighbour)
    for x in visited:
        if not x:
            return 'connected:no'
    return 'connected:yes'

def dfs(graph,begin,ver,visited,last):
    # print(f'dealing with {ver}, current visited:{visited}')
    if visited[ver] and ver==begin:

```

```

#         print('have loop')
        return True
    elif visited[ver] and ver!=begin:
        return False
    else:
        visited[ver]=True
        for neighbour in graph.vertexes[ver].connectedto.keys():
            if neighbour!=last and dfs(graph,begin,neighbour,visited,ver):
                return True
        return False

def loop(graph):
    for i in range(n):
        visited=[False for i in range(n)]
        if dfs(graph,i,i,visited,i):
            return 'loop:yes'
    return 'loop:no'

print(connected(g))
print(loop(g))

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: Accepted

源代码

```
class Vertex:
    def __init__(self,v):
        self.value=v
        self.connectedto={}

class Graph:
    def __init__(self):
        self.vertexes={}

    def add_vertex(self,s):
        self.vertexes[s]=Vertex(s)

    def add_edge(self,s1,s2,w):
        if s1 not in self.vertexes:
            self.vertexes[s1]=Vertex(s1)
        if s2 not in self.vertexes:
            self.vertexes[s2]=Vertex(s2)
        V1=self.vertexes[s1]
        V2=self.vertexes[s2]
        V1.connectedto[s2]=w
        V2.connectedto[s1]=w
        self.vertexes[s1]=V1
        self.vertexes[s2]=V2

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

for _ in range(m):
    u,v=map(int,input().split())
    g.add_edge(u,v,1)

def connected(graph):
    if n==1:
        return 'connected:yes'
    visited=[False for i in range(n)]
    queue=[0]
    while queue:
        ver=queue.pop(0)
        neighbours=graph.vertexes[ver].connectedto.keys()
```

基本信息

#: 44904

题目: 27635

提交人: zzk

内存: 3940k

时间: 30ms

语言: Python

提交时间: 2024-

27947: 动态中位数

<http://cs101.openjudge.cn/practice/27947/>

思路:

双堆结构，先输入，后维护，数多一个就输出

代码

```
import heapq
class DoubleHeap:
    def __init__(self):
        self.minheap=[]
        self.maxheap=[]
```



```

self.mids=[]

def add_num(self,v):
    if not self.minheap or v>self.minheap[0]:
        heapq.heappush(self.minheap,v)
    else:
        heapq.heappush(self.maxheap,-v)

    if len(self.minheap)>len(self.maxheap)+1:
        heapq.heappush(self.maxheap,-heapq.heappop(self.minheap))
    elif len(self.minheap)<len(self.maxheap)-1:
        heapq.heappush(self.minheap,-heapq.heappop(self.maxheap))
    elif len(self.minheap)==len(self.maxheap)+1:
        self.mids.append(self.minheap[0])
    elif len(self.maxheap)==len(self.minheap)+1:
        self.mids.append(-self.maxheap[0])
    return

def option(self,data):
    for x in data:
        self.add_num(x)
    return self.mids

T=int(input())
for _ in range(T):
    h=DoubleHeap()
    mids=h.option([int(x) for x in input().split()])
    ans=[str(x) for x in mids]
    print(len(ans))
    print(' '.join(ans))

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

题目

排名

状态

提问

#44962043提交状态

查看

状态: Accepted

源代码

```
import heapq
class DoubleHeap:
    def __init__(self):
        self.minheap=[]
        self.maxheap=[]
        self.mids=[]

    def add_num(self,v):
        if not self.minheap or v>self.minheap[0]:
            heapq.heappush(self.minheap,v)
        else:
            heapq.heappush(self.maxheap,-v)

        if len(self.minheap)>len(self.maxheap)+1:
            heapq.heappush(self.maxheap,-heapq.heappop(self.minheap))
        elif len(self.minheap)<len(self.maxheap)-1:
            heapq.heappush(self.minheap,-heapq.heappop(self.maxheap))
        elif len(self.minheap)==len(self.maxheap)+1:
            self.mids.append(self.minheap[0])
        elif len(self.maxheap)==len(self.minheap)+1:
            self.mids.append(-self.maxheap[0])
        return

    def option(self,data):
        for x in data:
            self.add_num(x)
        return self.mids

T=int(input())
for _ in range(T):
    h=DoubleHeap()
    mids=h.option([int(x) for x in input().split()])
    ans=[str(x) for x in mids]
    print(len(ans))
    print(' '.join(ans))
```

基本信息

#: 44962

题目: 27947

提交人: zxk

内存: 11256

时间: 352m

语言: Python

提交时间: 2024-

©2002-2022 POJ 京ICP备20010980号-1

28190: 奶牛排队

<http://cs101.openjudge.cn/practice/28190/>

思路:

代码

```
#
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

2. 学习总结和收获

如果作业题目简单，有否额外练习题目，比如：OJ“2024spring每日选做”、CF、LeetCode、洛谷等网站题目。

本次在限时内只做出1，2，4题，3，5题卡住了不会做，6没看到

感觉难度挺大的，特别是Agri-Net和动态中位数莫名其妙的出错，Agri竟然不用考虑输入数据换行？？？，亏的还想了半天也没弄清楚；学习了Krusakal的最小生成树，感觉对于这个算法并不熟悉，是用于全局最短路径的，应该记下来，关键在代码里“按秩合并”的方法很好，在合并的同时也进行了是否连通的判断，并不是普通的并查集，换句话说，K算法的很大一部分实际是在Union里实现的；动态中位数大概琢磨一下好像AVL能做不过肯定不是我能写出来的（倒是自己想到了双堆的办法，卡在了不知道怎么做一个具体的过程，查了一下看到是大的放一遍小的放一边，保持数目不变，一边最大堆一边最小堆，这样卡在中间的一定是中位数。

单调栈看了很长时间还是没看懂，先交了作业，之后再看（悲