Assignment #D: May月考

Updated 1654 GMT+8 May 8, 2024

2024 spring, Complied by ==祁轩宇、经济学院==

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：Windows 11, version 23H2

Python编程环境：VSCode 1.87.1

C/C++编程环境：

1. 题目

02808: 校门外的树

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

思路：

代码

```
L,M=map(int,input().split())
Tree=list(range(L+1))

Remove=[]
for i in range(M):
    a,b=map(int,input().split())
    Remove+=list(range(a,b+1))

Remove=set(Remove)
n=0
for i in Tree:
    if i in Remove:
        continue
    else:
```

```
            n+=1
print(n)
```

代码运行截图 ==（至少包含有"Accepted"）==

#45037039提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
L,M=map(int,input().split())
Tree=list(range(L+1))

Remove=[]
for i in range(M):
    a,b=map(int,input().split())
    Remove+=list(range(a,b+1))

Remove=set(Remove)
n=0
for i in Tree:
    if i in Remove:
        continue
    else:
        n+=1
print(n)
```

基本信息

#: 45037039
题目: 02808
提交人: 2100015440
内存: 18848kB
时间: 40ms
语言: Python3
提交时间: 2024-05-21 20:41:39

English 帮助 关于

20449: 是否被5整除

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

思路：

代码

```
def Trans(N_i):
    N=0
    for i in range(len(N_i)):
        N+=int(N_i[i])*2**(len(N_i)-1-i)
    return N

A=input()
answer=[]
for i in range(len(A)):
    N_i=A[0:i+1]
    if Trans(N_i)%5==0:
        answer.append('1')
    else:
        answer.append('0')
print(''.join(answer))
```

代码运行截图 ==（至少包含有"Accepted"）==

#45037204提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def Trans(N_i):
    N=0
    for i in range(len(N_i)):
        N+=int(N_i[i])*2**(len(N_i)-1-i)
    return N

A=input()
answer=[]
for i in range(len(A)):
    N_i=A[0:i+1]
    if Trans(N_i)%5==0:
        answer.append('1')
    else:
        answer.append('0')
print(''.join(answer))
```

基本信息

#: 45037204
题目: 20449
提交人: 2100015440
内存: 3612kB
时间: 19ms
语言: Python3
提交时间: 2024-05-21 20:52:39

English 帮助 关于

01258: Agri-Net

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

思路：

代码

```
import sys

# Function to find the minimum spanning tree using Prim's algorithm
def prim_mst(graph, N):
    # Initialize key values, parent array, and MST set
    key = [float('inf')] * N
    parent = [None] * N
    mst_set = [False] * N

    # Start with the first node
    key[0] = 0
    parent[0] = -1

    for _ in range(N):
        # Find the vertex with the minimum key value
        min_key = float('inf')
        min_idx = -1
        for i in range(N):
            if not mst_set[i] and key[i] < min_key:
                min_key = key[i]
                min_idx = i

        # Add the selected vertex to the MST set
        mst_set[min_idx] = True
```

```
        # Update key values and parent array for adjacent vertices
        for j in range(N):
            if graph[min_idx][j] and not mst_set[j] and graph[min_idx][j] <
key[j]:
                parent[j] = min_idx
                key[j] = graph[min_idx][j]

    # Calculate the sum of key values in the MST
    mst_sum = sum(key)
    return mst_sum

def main():
    for line in sys.stdin:
        # Read the number of farms
        N = int(line.strip())

        # Read the connectivity matrix
        graph = []
        for _ in range(N):
            row = list(map(int, input().split()))
            graph.append(row)

        # Find the minimum spanning tree using Prim's algorithm
        mst_length = prim_mst(graph, N)

        # Output the minimum length of fiber required
        print(mst_length)

if __name__ == "__main__":
    main()
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#45038158提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 45038158
题目: 01258
提交人: 2100015440
内存: 3912kB
时间: 29ms
语言: Python3
提交时间: 2024-05-21 22:11:23

源代码

```
import sys

# Function to find the minimum spanning tree using Prim's algorithm
def prim_mst(graph, N):
    # Initialize key values, parent array, and MST set
    key = [float('inf')] * N
    parent = [None] * N
    mst_set = [False] * N

    # Start with the first node
    key[0] = 0
    parent[0] = -1

    for _ in range(N):
        # Find the vertex with the minimum key value
        min_key = float('inf')
        min_idx = -1
        for i in range(N):
            if not mst_set[i] and key[i] < min_key:
                min_key = key[i]
                min_idx = i

        # Add the selected vertex to the MST set
        mst_set[min_idx] = True

        # Update key values and parent array for adjacent vertices
        for j in range(N):
            if graph[min_idx][j] and not mst_set[j] and graph[min_idx][j
                parent[j] = min_idx
                key[j] = graph[min_idx][j]

    # Calculate the sum of key values in the MST
    mst_sum = sum(key)
    return mst_sum

def main():
    for line in sys.stdin:
        # Read the number of farms
        N = int(line.strip())
```

27635: 判断无向图是否连通有无回路(同23163)

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

思路：

代码

```
from collections import deque

def bfs(node, visited, adjacency_list, connected_components):
    queue=deque()
    if not visited[node]:
        queue.append(node)
        visited[node] = True
        before=after=0
        while queue:
            x=queue.popleft()
            before=sum(visited)
            for neighbor in adjacency_list[x]:
```

```
                if not visited[neighbor]:
                    visited[neighbor]=True
                    queue.append(neighbor)
            after=sum(visited)
            connected_components[x]=after-before

n,m = map(int, input().split())
adjacency_list = [[] for _ in range(n)]
for _ in range(m):
    u, v = map(int, input().split())
    adjacency_list[u].append(v)
    adjacency_list[v].append(u)

visited = [False] * n
connected_components = [0]*n
connection=loop='no'

bfs(0, visited, adjacency_list, connected_components)
if sum(visited)==n:
    connection='yes'

for i in range(n):
    if not visited[i]:
        bfs(i, visited, adjacency_list, connected_components)

for i in range(n):
    if len(adjacency_list[i])-connected_components[i]>1:
        loop='yes'

print(f'connected:{connection}')
print(f'loop:{loop}')
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

#45037986提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

def bfs(node, visited, adjacency_list, connected_components):
    queue=deque()
    if not visited[node]:
        queue.append(node)
        visited[node] = True
        before=after=0
        while queue:
            x=queue.popleft()
            before=sum(visited)
            for neighbor in adjacency_list[x]:
                if not visited[neighbor]:
                    visited[neighbor]=True
                    queue.append(neighbor)
            after=sum(visited)
            connected_components[x]=after-before

n,m = map(int, input().split())
adjacency_list = [[] for _ in range(n)]
for _ in range(m):
    u, v = map(int, input().split())
    adjacency_list[u].append(v)
    adjacency_list[v].append(u)

visited = [False] * n
connected_components = [0]*n
connection='no'

bfs(0, visited, adjacency_list, connected_components)
if sum(visited)==n:
    connection='yes'

for i in range(n):
    if not visited[i]:
        bfs(i, visited, adjacency_list, connected_components)

for i in range(n):
    if len(adjacency_list[i])-connected_components[i]>1:
        loop='yes'

print(f'connected:{connection}')
print(f'loop:{loop}')
```

基本信息

#:	45037986
题目:	27635
提交人:	2100015440
内存:	3740kB
时间:	27ms
语言:	Python3
提交时间:	2024-05-21 21:56:41

27947: 动态中位数

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

思路：

代码

```
#
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

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。==

前两题比较简单，属于送分题，第三题考了Prim算法，参考了chatgpt的回答，第四题是bfs，可以使用之前的模板完成。