Задача. Между какими городами нужно проложить линии связи, чтобы все города были

связаны в одну систему и общая длина линий связи была наименьшей?

(минимальное остовное дерево)

###########Ostov\_proga############

N=int(input())

W=[list(map(int, input().split())) for i in range(N)]

col=[i for i in range(N)]

ostov=[]

Sum\_ostov=0

for k in range(N-1):

    minDist=10\*\*10

    for i in range(N-1):

        for j in range(i+1,N):

            if col[i]!=col[j]  and W[i][j]!=0 and W[i][j]<minDist:

                iMin=i

                jMin=j

                minDist=W[i][j]

    ostov.append((iMin,jMin))

    c=col[jMin]

    for i in range(N):

        if col[i]==c:

            col[i]=col[iMin]

for edge in ostov:

    Sum\_ostov+=W[edge[0]][edge[1]]

    print("(",edge[0],",",edge[1],")")

print("Сумма остовного дерева:",Sum\_ostov)

Дан неориентированный невзвешенный граф. Для него вам необходимо найти количество вершин,

лежащих в одной компоненте связности с данной вершиной (считая эту вершину).

sm=set()

def dfs(u):

    global sm

    vis[u]= False

    sm.add(u+1)

    for v in range(n):

        if vis[v] and ma[u][v]==1:

            dfs(v)

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

vis=[True]\*n; ma=[]

for i in range(n):

    ma.append(list(map(int,input().split())))

###### max len компонента свзяности

ans=[]

for u in range(n):

    vis=[True]\*n

    sm=set()

    dfs(u)

    if nach in sm:

        ans.append(len(sm))

print(max(ans))

#####проверка на оринтированный граф

"""dfs(u)

if len(sm)==n:

    print("YES")

else:

    print("NO")"""

В неориентированном графе требуется найти минимальный путь между двумя вершинами.

n = int(input())

a = [list(map(int, input().split())) for i in range(n)]

vis = [-1]\*n

s, f = map(int, input().split())

s -= 1

f -= 1

q = [s]

vis[s]=0

while q:

    ver=q.pop(0)

    for i in range(n):

        if a[ver][i]==1 and vis[i]==-1:

            q.append(i)

            vis[i]=vis[ver]+1

print(vis[f])

Дан ориентированный взвешенный граф. Найти кратчайшее расстояние от вершины S до вершины F.

n,start,f=map(int,input().split())

start -= 1; f -= 1; w = []

for i in range(n):

    w.append(list(map(int,input().split())))

INF = 10 \*\* 10

for i in range(n):

    for j in range(n):

        if w[i][j]==-1:

            w[i][j]=INF

dist = [INF] \* n

dist[start] = 0

used = [False] \* n

min\_dist = 0

min\_vertex = start

while min\_dist < INF:

    i = min\_vertex

    used[i] = True

    for j in range(n):

        if dist[i] + w[i][j] < dist[j]:

            dist[j] = dist[i] + w[i][j]

    min\_dist = INF

    for j in range(n):

        if not used[j] and dist[j] < min\_dist:

            min\_dist = dist[j]

            min\_vertex = j

if dist[f] != 10\*\*10:

    print(dist[f])

else:

    print(-1)

Дан ориентированный взвешенный граф. Найдите кратчайший путь от одной заданной вершины до другой.

def path(f):

    if f==s:

        print(s+1,end=' ')

        return

    path(p[f])

    print(f+1,end=' ')

n, s, f=map(int,input().split())

a=[list(map(int,input().split())) for i in range(n)]

p=[0]\*n; inf=1e10; d=[inf]\*n; vis=[False]\*n

s-=1;f-=1; d[s]=0

for i in range(n):

    mi=inf

    for j in range(n):

        if not vis[j] and d[j]<mi :

            mi=d[j]

            kmi=j

    if mi==inf: break

    vis[kmi]=True

    for j in range(n):

        if a[kmi][j]!=0 and not vis[j] and d[kmi]+a[kmi][j]<d[j]:

            d[j]=d[kmi]+a[kmi][j]

            p[j]=kmi

if d[f]>=inf:

    print(-1)

else:

    path (f)