## 第一题

### 第一问

如果要使各村庄村民到医疗点的距离总和S1最小，请问这3个医疗点分别建立在何处最好？总距离S1是多少？

#### 方法一：暴力递归

# -\*- coding: utf-8 -\*-  
  
import networkx as nx  
import pandas as pd  
import multiprocessing  
import itertools  
  
# 读取位置表单  
locations = pd.read\_excel("data1.xlsx", sheet\_name="位置", header=0,index\_col=0)  
  
# 读取连接道路表单  
roads = pd.read\_excel("data.xlsx", sheet\_name="连接道路")  
  
# 创建空图  
G = nx.Graph()  
  
# 添加节点  
for node in locations.index:  
 G.add\_node(node, pos=(locations.loc[node, "X"], locations.loc[node, "Y"]))  
  
# 添加边  
for i, row in roads.iterrows():  
 G.add\_edge(row["起点"], row["终点"], weight=row["距离"])  
file=open('res1.txt','w')  
  
  
# 定义函数，计算每个村庄到其对应医疗点的距离，并计算距离总和S1  
def calculate\_distance\_sum(G, medical\_centers):  
 distance\_sum = 0  
 for node in G.nodes:  
 min\_distance = float("inf")  
 for center in medical\_centers:  
 distance = nx.shortest\_path\_length(G, source=node, target=center, weight="weight")  
 if distance < min\_distance:  
 min\_distance = distance  
 distance\_sum += min\_distance  
 return distance\_sum  
  
  
  
# 定义函数，找到距离总和S1最小的医疗点组合  
def find\_best\_medical\_centers(G):  
 nodes = list(G.nodes)  
 min\_distance\_sum = float("inf")  
 best\_centers = None  
 # 枚举所有可能的组合情况  
 combinations = [c for c in itertools.combinations(nodes, 3)]  
 a=combinations  
 for centers in a:  
 distance\_sum = calculate\_distance\_sum(G, centers)  
 data=str(centers)+str(distance\_sum)+'\n'  
 # 记录距离总和最小的组合  
 file.write(data)  
 if distance\_sum < min\_distance\_sum:  
 min\_distance\_sum = distance\_sum  
 best\_centers = centers  
 return best\_centers, min\_distance\_sum  
  
best\_centers, min\_distance\_sum = find\_best\_medical\_centers(G)  
print("最优的医疗点组合为：", best\_centers)  
print("距离总和为：", min\_distance\_sum)  
file.close()

注意：在进行暴力递归之前，需要根据X，Y坐标先算出边的权值。在进行暴力递归。暴力递归的数据已给出。最后求出的结果为：

(10, 50, 57)316598.7433864181

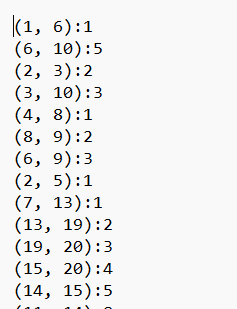
医疗站为10 ，50，57 最短距离为：S1=316598.7433864181

### 第二问：

各村庄村民都选择最近的医疗点看病，请问应该维修哪些道路，维修道路总里程S2是多少？作图用不同颜色标记各村庄到对应医疗点使用的道路。

import numpy as np  
import pandas as pd  
import networkx as nx  
  
# 读取位置表单  
locations = pd.read\_excel("data1.xlsx", sheet\_name="位置", header=0, index\_col=0)  
  
# 读取连接道路表单  
roads = pd.read\_excel("data.xlsx", sheet\_name="连接道路")  
  
# 构建距离矩阵  
dist\_mat = np.zeros((len(locations), len(locations)))  
for i, row in roads.iterrows():  
 start\_loc = row["起点"]  
 end\_loc = row["终点"]  
 distance = row["距离"]  
 start\_idx = locations.index.get\_loc(start\_loc)  
 end\_idx = locations.index.get\_loc(end\_loc)  
 dist\_mat[start\_idx, end\_idx] = distance  
 dist\_mat[end\_idx, start\_idx] = distance  
  
# 创建连通图  
G = nx.Graph()  
  
# 添加节点  
for node in locations.index:  
 G.add\_node(node, pos=(locations.loc[node, "X"], locations.loc[node, "Y"]))  
  
# 添加边  
for i, row in roads.iterrows():  
 G.add\_edge(row["起点"], row["终点"], weight=row["距离"])  
mapping = {node: int(node) for node in G.nodes()}  
G = nx.relabel\_nodes(G, mapping)  
# 找到所有节点到10、50、57节点的最短路径  
shortest\_paths\_10 = nx.single\_source\_dijkstra\_path\_length(G,10)  
shortest\_paths\_50 = nx.single\_source\_dijkstra\_path\_length(G,50)  
shortest\_paths\_57 = nx.single\_source\_dijkstra\_path\_length(G,57)  
shortest\_path = nx.shortest\_path(G, 4, 10, weight='weight')  
print(shortest\_path)  
  
# 计算每个节点到10、50、57节点的实际最短距离和路径  
result = {}  
road\_map={}  
for node in G.nodes():  
  
 shortest\_distance\_10 =shortest\_paths\_10[node]  
 shortest\_path\_10 =nx.shortest\_path(G,node,10,weight='weight')  
   
 shortest\_distance\_50 =shortest\_paths\_50[node]  
 shortest\_path\_50 = nx.shortest\_path(G,node,50,weight='weight')  
  
 shortest\_distance\_57 = shortest\_paths\_57[node]  
 shortest\_path\_57 = nx.shortest\_path(G,node,57,weight='weight')  
  
 shortest\_distance = min(shortest\_distance\_10, shortest\_distance\_50, shortest\_distance\_57)  
 shortest\_target = None  
 shortest\_path = []  
 if shortest\_distance == shortest\_distance\_10:  
 shortest\_target = 10  
 shortest\_path = shortest\_path\_10  
 elif shortest\_distance == shortest\_distance\_50:  
 shortest\_target = 50  
 shortest\_path = shortest\_path\_50  
 elif shortest\_distance == shortest\_distance\_57:  
 shortest\_target = 57  
 shortest\_path = shortest\_path\_57  
 # 计算路径上的权重之和  
 weight\_sum = 0  
 for i in range(len(shortest\_path)-1):  
 u, v = shortest\_path[i], shortest\_path[i+1]  
 weight\_sum += G[u][v]['weight']  
 edges = [(shortest\_path[i],shortest\_path[i+1]) for i in range(len(shortest\_path)-1)]  
 sorted\_edges = [tuple(sorted(edge)) for edge in edges]  
  
 print(node,sorted\_edges)  
 for i in range(len(sorted\_edges)):  
 if sorted\_edges[i] not in road\_map:  
 road\_map[sorted\_edges[i]] = 1  
 else:  
 road\_map[sorted\_edges[i]] += 1  
 result[node] = {'target': shortest\_target, 'distance': shortest\_distance, 'path': shortest\_path, 'weight': weight\_sum}  
  
# 将结果存储到Excel文件中  
df = pd.DataFrame.from\_dict(result, orient='index')  
df.index.name = 'node'  
df.to\_excel('result.xlsx')  
with open('road\_map.txt', 'w') as file:  
 for road in road\_map:  
 file.write(f'{road}:{road\_map[road]}\n')  
total\_distance = 0  
for (a, b) in road\_map:  
 distance =G.get\_edge\_data(a, b)['weight'] # 获取边的权值  
 total\_distance += distance  
print("total\_distance=",total\_distance)

根据dijkstra算法算出最短路径，并将结果存入到excel表中去。将边去重得到一张map，对map进行求和即可



S2=total\_distance= 95365.9386814891

## 第二题

由于每条道路维修都需要成本，因此站在道路维修公司角度出发，希望维修的成本尽量低。假定问题1中得到的医疗点不变，应该维修哪些道路，使得维修成本最低。给出维修道路的总长度S2，并作出图形。同时根据维修的道路，计算各村庄到医疗点的总距离S1。

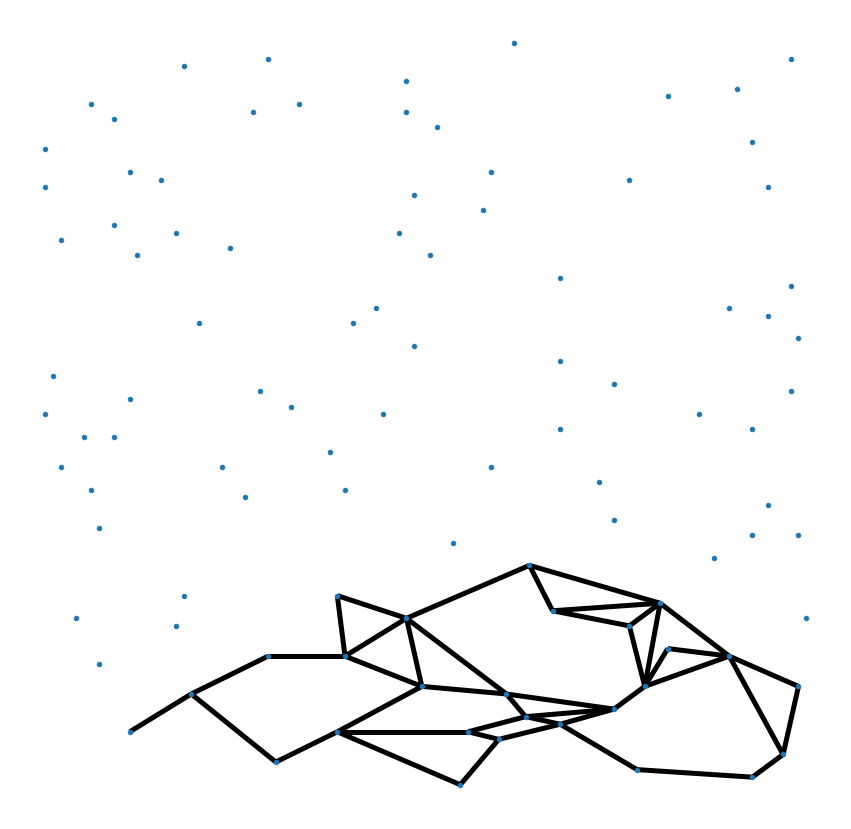
思路：将整个大图划分为三个子图，划分的规则就是第一问所求的村庄和医疗站的关系，分别对三个子图应用最小生成树的规则即可:

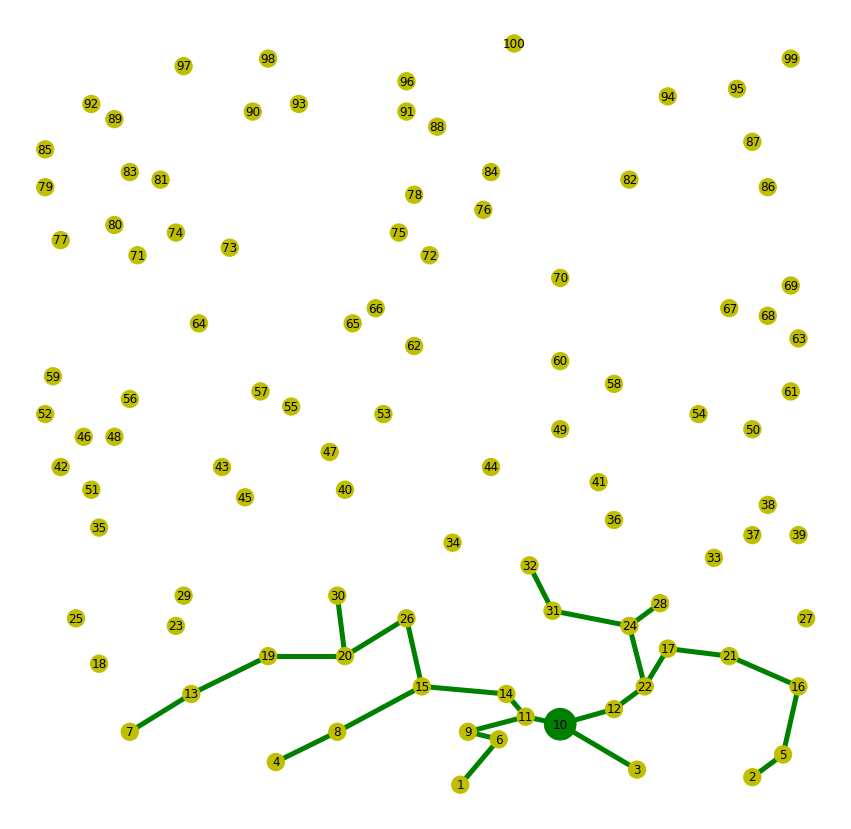
根据result.xlsx按照target为10 50 57 分别生成三个excel文件,便于之后的读取:

# -\*- coding: utf-8 -\*-
  
import pandas as pd
  
  
# 读取 excel 文件
  
df = pd.read\_excel('result.xlsx', sheet\_name='Sheet1')
  
  
# 按照 target 分组
  
grouped = df.groupby('target')
  
  
# 遍历每个分组，将分组数据保存为单独的 Excel 文件
  
for target, group in grouped:
  
 # 构造文件名
  
 filename = f'target\_{target}.xlsx'
  
 # 保存数据
  
 group.to\_excel(filename, index=False)

### 第一组数据:

# -\*- coding: utf-8 -\*-
  
"""
  
Created on Tue May 2 10:03:10 2023
  
  
@author: wangyufan
  
"""
  
  
  
import networkx as nx
  
import pandas as pd
  
import numpy as np
  
  
res1 = pd.read\_excel("target\_10.xlsx", sheet\_name="Sheet1", header=0,index\_col=0).index
  
locations1 = pd.read\_excel("data1.xlsx", sheet\_name="位置", header=0,index\_col=0)
  
# 读取连接道路表单
  
roads1 = pd.read\_excel("data.xlsx", sheet\_name="连接道路")
  
  
# 创建空图
  
G = nx.Graph()
  
  
# 添加节点
  
  
for node in locations1.index:
  
 #if node in res:
  
 G.add\_node(node, pos=(locations1.loc[node, "X"], locations1.loc[node, "Y"]))
  
  
# 添加边
  
nodes1=res1
  
for i, row in roads1.iterrows():
  
  
 if int(row["起点"]) in nodes1 and int(row["终点"]) in nodes1:
  
 G.add\_edge(row["起点"], row["终点"], weight=row["距离"])
  
# 对节点进行整数转换
  
mapping = {node: int(node) for node in G.nodes()}
  
G = nx.relabel\_nodes(G, mapping)
  
  
import matplotlib.pyplot as plt
  
  
# 获取节点位置
  
pos = nx.get\_node\_attributes(G, "pos")
  
plt.figure(figsize=(15, 15))
  
# 绘制节点和边
  
nx.draw\_networkx\_nodes(G, pos, node\_size=20)
  
nx.draw\_networkx\_edges(G, pos, width=5)
  
  
# 显示图
  
plt.axis("off")
  
plt.show()
  
  
T = nx.minimum\_spanning\_tree(G)
  
# 获取最小生成树的边
  
plt.figure(figsize=(15, 15))
  
  
tree\_edges = list(T.edges())
  
node\_colors = ['y' if node != 10 else 'g' for node in G.nodes()]
  
node\_sizes = [1000 if node == 10 else 300 for node in G.nodes()]
  
# 绘制节点和边，并将最小生成树的边标记为红色
  
nx.draw\_networkx\_nodes(G, pos,node\_color=node\_colors, node\_size=node\_sizes)
  
nx.draw\_networkx\_labels(G, pos, {n: str(n) for n in G.nodes()})
  
nx.draw\_networkx\_edges(G, pos, edgelist=tree\_edges, edge\_color='g', width=5)
  
  
# 显示图
  
plt.axis("off")
  
plt.show()
  
  
s\_10=0
  
for node in nodes1:
  
 distance = nx.shortest\_path\_length(T, source=node, target=10, weight='weight')
  
 s\_10+=distance
  
  
print("最短距离",s\_10)
  
  
length=0
  
for (a,b) in tree\_edges:
  
 length+=G.get\_edge\_data(a,b)['weight']
  
  
  
print("最小生成树长度：",length)

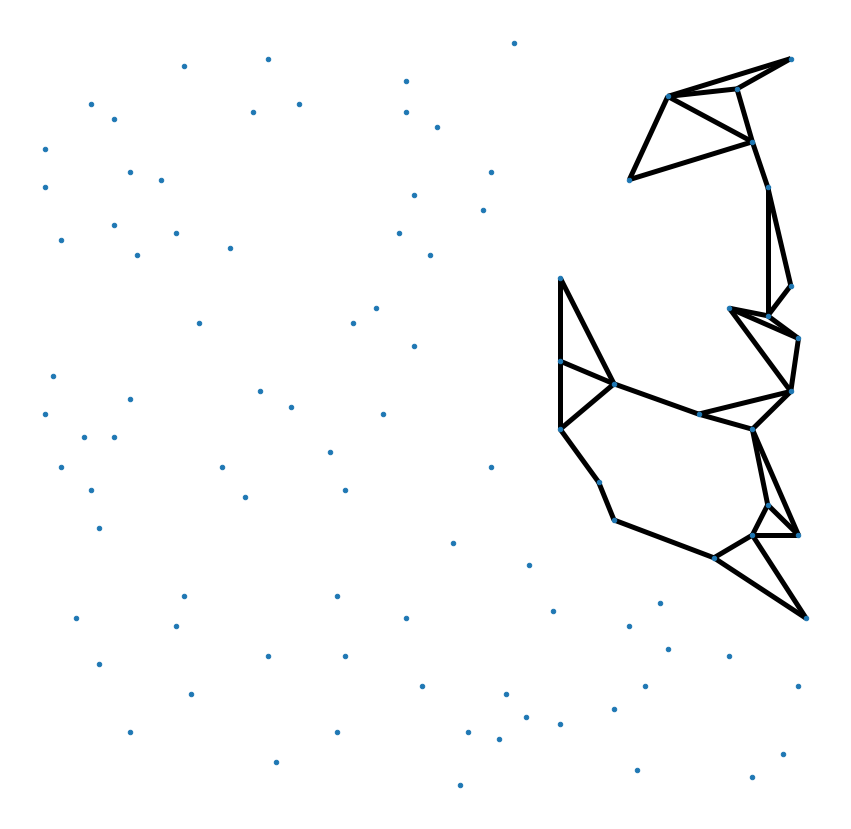


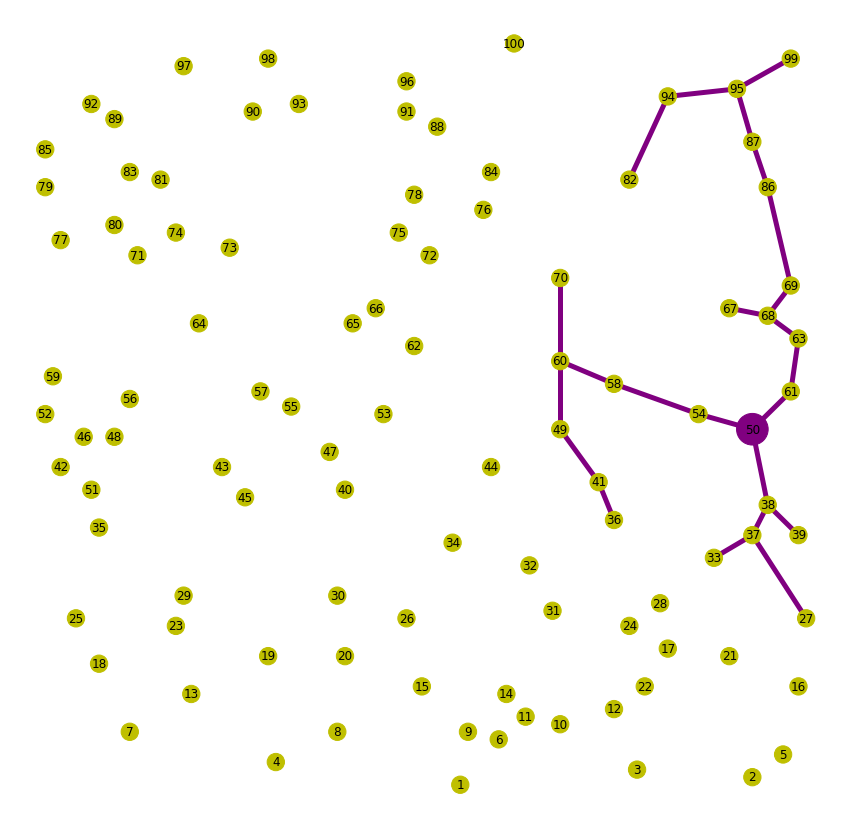


最短距离 76990.90151994597
  
最小生成树长度： 21011.612087891008

### 第二组

import networkx as nx
  
import pandas as pd
  
import numpy as np
  
  
res1 = pd.read\_excel("target\_50.xlsx", sheet\_name="Sheet1", header=0,index\_col=0).index
  
locations1 = pd.read\_excel("data1.xlsx", sheet\_name="位置", header=0,index\_col=0)
  
# 读取连接道路表单
  
roads1 = pd.read\_excel("data.xlsx", sheet\_name="连接道路")
  
  
# 创建空图
  
G = nx.Graph()
  
  
# 添加节点
  
  
for node in locations1.index:
  
 #if node in res:
  
 G.add\_node(node, pos=(locations1.loc[node, "X"], locations1.loc[node, "Y"]))
  
  
# 添加边
  
nodes1=res1
  
for i, row in roads1.iterrows():
  
  
 if int(row["起点"]) in nodes1 and int(row["终点"]) in nodes1:
  
 G.add\_edge(row["起点"], row["终点"], weight=row["距离"])
  
# 对节点进行整数转换
  
mapping = {node: int(node) for node in G.nodes()}
  
G = nx.relabel\_nodes(G, mapping)
  
  
import matplotlib.pyplot as plt
  
  
# 获取节点位置
  
plt.figure(figsize=(15, 15))
  
pos = nx.get\_node\_attributes(G, "pos")
  
  
# 绘制节点和边
  
nx.draw\_networkx\_nodes(G, pos, node\_size=20)
  
nx.draw\_networkx\_edges(G, pos, width=5)
  
  
# 显示图
  
plt.axis("off")
  
plt.show()
  
  
T = nx.minimum\_spanning\_tree(G)
  
# 获取最小生成树的边
  
tree\_edges = list(T.edges())
  
  
plt.figure(figsize=(15, 15))
  
  
tree\_edges = list(T.edges())
  
node\_colors = ['y' if node != 50 else 'purple' for node in G.nodes()]
  
node\_sizes = [1000 if node == 50 else 300 for node in G.nodes()]
  
# 绘制节点和边，并将最小生成树的边标记为红色
  
nx.draw\_networkx\_nodes(G, pos,node\_color=node\_colors, node\_size=node\_sizes)
  
nx.draw\_networkx\_labels(G, pos, {n: str(n) for n in G.nodes()})
  
nx.draw\_networkx\_edges(G, pos, edgelist=tree\_edges, edge\_color='purple', width=5)
  
# 显示图
  
plt.axis("off")
  
plt.show()
  
  
s\_50=0
  
for node in nodes1:
  
 distance = nx.shortest\_path\_length(T, source=node, target=50, weight='weight')
  
 s\_50+=distance
  
  
print(s\_50)
  
  
length=0
  
for (a,b) in tree\_edges:
  
 length+=G.get\_edge\_data(a,b)['weight']
  
  
  
print(length)

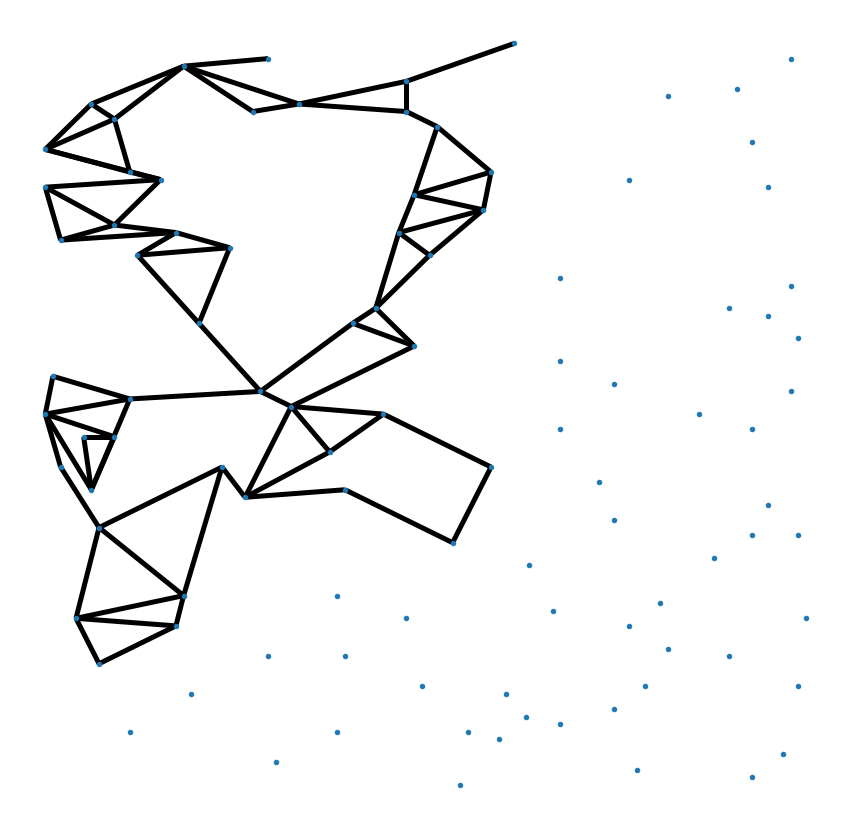




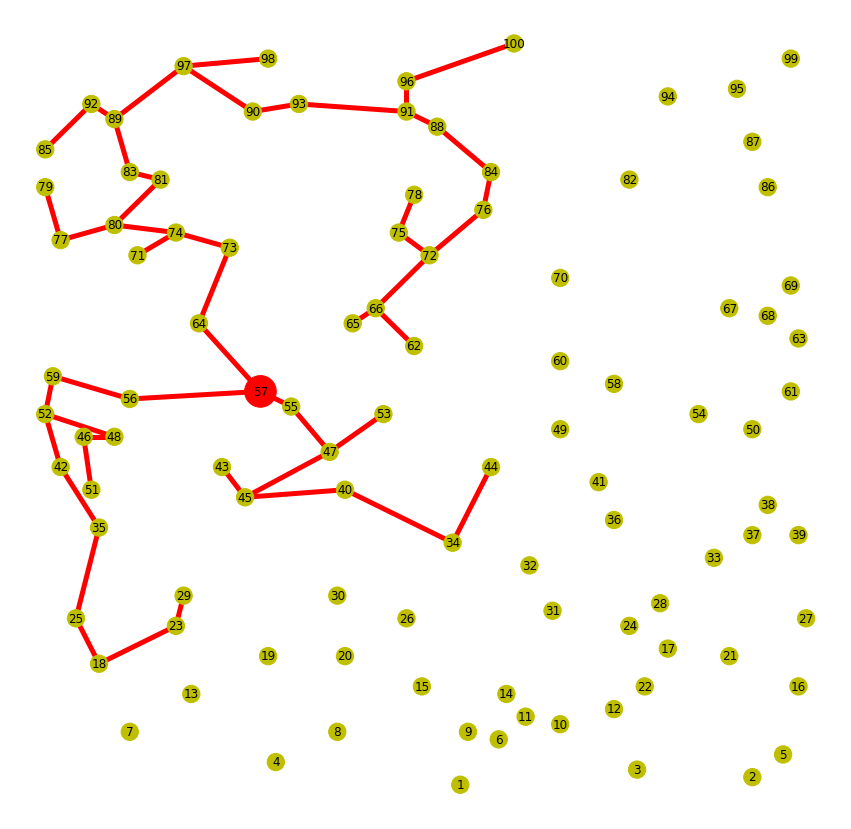
72143.11816199898
  
18517.26321880672

### 第三组数据

# -\*- coding: utf-8 -\*-
  
"""
  
Created on Tue May 2 10:03:10 2023
  
  
@author: wangyufan
  
"""
  
  
  
import networkx as nx
  
import pandas as pd
  
import numpy as np
  
  
res1 = pd.read\_excel("target\_57.xlsx", sheet\_name="Sheet1", header=0,index\_col=0).index
  
locations1 = pd.read\_excel("data1.xlsx", sheet\_name="位置", header=0,index\_col=0)
  
# 读取连接道路表单
  
roads1 = pd.read\_excel("data.xlsx", sheet\_name="连接道路")
  
  
# 创建空图
  
G = nx.Graph()
  
  
# 添加节点
  
  
for node in locations1.index:
  
 #if node in res:
  
 G.add\_node(node, pos=(locations1.loc[node, "X"], locations1.loc[node, "Y"]))
  
  
# 添加边
  
nodes1=res1
  
for i, row in roads1.iterrows():
  
  
 if int(row["起点"]) in nodes1 and int(row["终点"]) in nodes1:
  
 G.add\_edge(row["起点"], row["终点"], weight=row["距离"])
  
# 对节点进行整数转换
  
mapping = {node: int(node) for node in G.nodes()}
  
G = nx.relabel\_nodes(G, mapping)
  
  
import matplotlib.pyplot as plt
  
plt.figure(figsize=(15, 15))
  
# 获取节点位置
  
pos = nx.get\_node\_attributes(G, "pos")
  
  
# 绘制节点和边
  
nx.draw\_networkx\_nodes(G, pos, node\_size=20)
  
nx.draw\_networkx\_edges(G, pos, width=5)
  
  
# 显示图
  
plt.axis("off")
  
plt.show()
  
  
T = nx.minimum\_spanning\_tree(G)
  
# 获取最小生成树的边
  
plt.figure(figsize=(15, 15))
  
  
tree\_edges = list(T.edges())
  
node\_colors = ['y' if node != 57 else 'red' for node in G.nodes()]
  
node\_sizes = [1000 if node == 57 else 300 for node in G.nodes()]
  
# 绘制节点和边，并将最小生成树的边标记为红色
  
nx.draw\_networkx\_nodes(G, pos,node\_color=node\_colors, node\_size=node\_sizes)
  
nx.draw\_networkx\_labels(G, pos, {n: str(n) for n in G.nodes()})
  
nx.draw\_networkx\_edges(G, pos, edgelist=tree\_edges, edge\_color='r', width=5)
  
  
# 显示图
  
plt.axis("off")
  
plt.show()
  
  
print(len(nodes1))
  
print(nodes1)
  
s\_57=0
  
for node in nodes1:
  
 distance = nx.shortest\_path\_length(T, source=node, target=57, weight='weight')
  
 s\_57+=distance
  
  
print(s\_57)
  
  
length=0
  
for (a,b) in tree\_edges:
  
 length+=G.get\_edge\_data(a,b)['weight']
  
  
  
print(length)

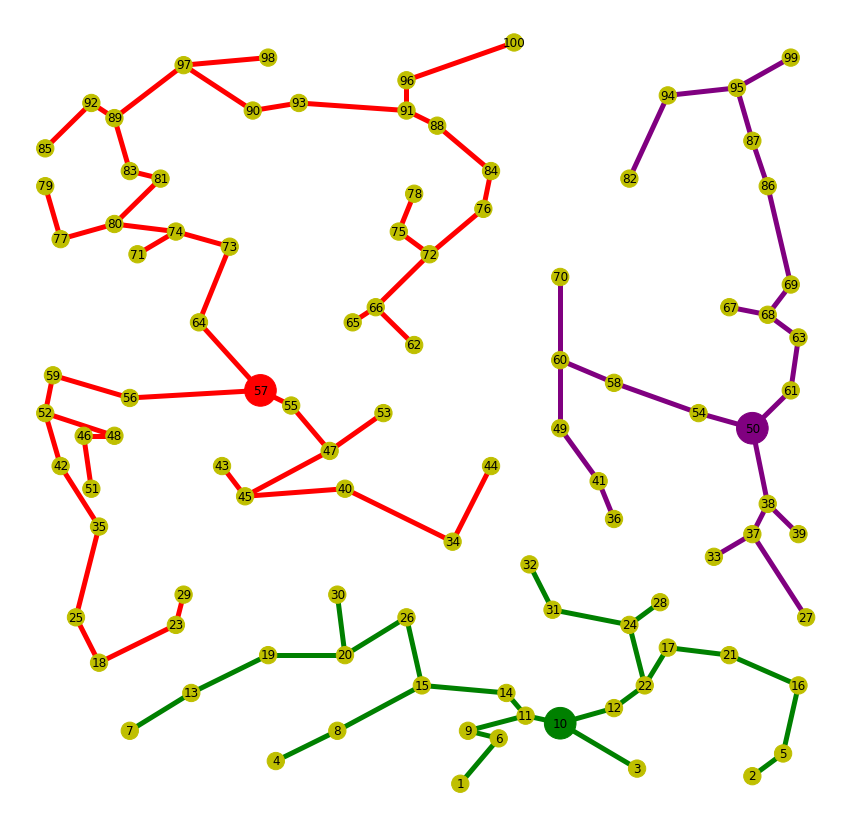


317809.1693991888
  
40425.215152604505



317809.1693991888
  
40425.215152604505

## 总和~哈哈哈画的有点丑



总距离:

s1=21011.612087891008+18517.26321880672+40425.215152604505=79954.09045930223

s2=76990.90151994597+72143.11816199898+317809.1693991888= 467943.18908113375