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

```python
# -*- 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

```
(1, 6):1
(6, 10):5
(2, 3):2
(3, 10):3
(4, 8):1
(8, 9):2
(6, 9):3
(2, 5):1
(7, 13):1
(13, 19):2
(19, 20):3
(15, 20):4
(14, 15):5
```

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：

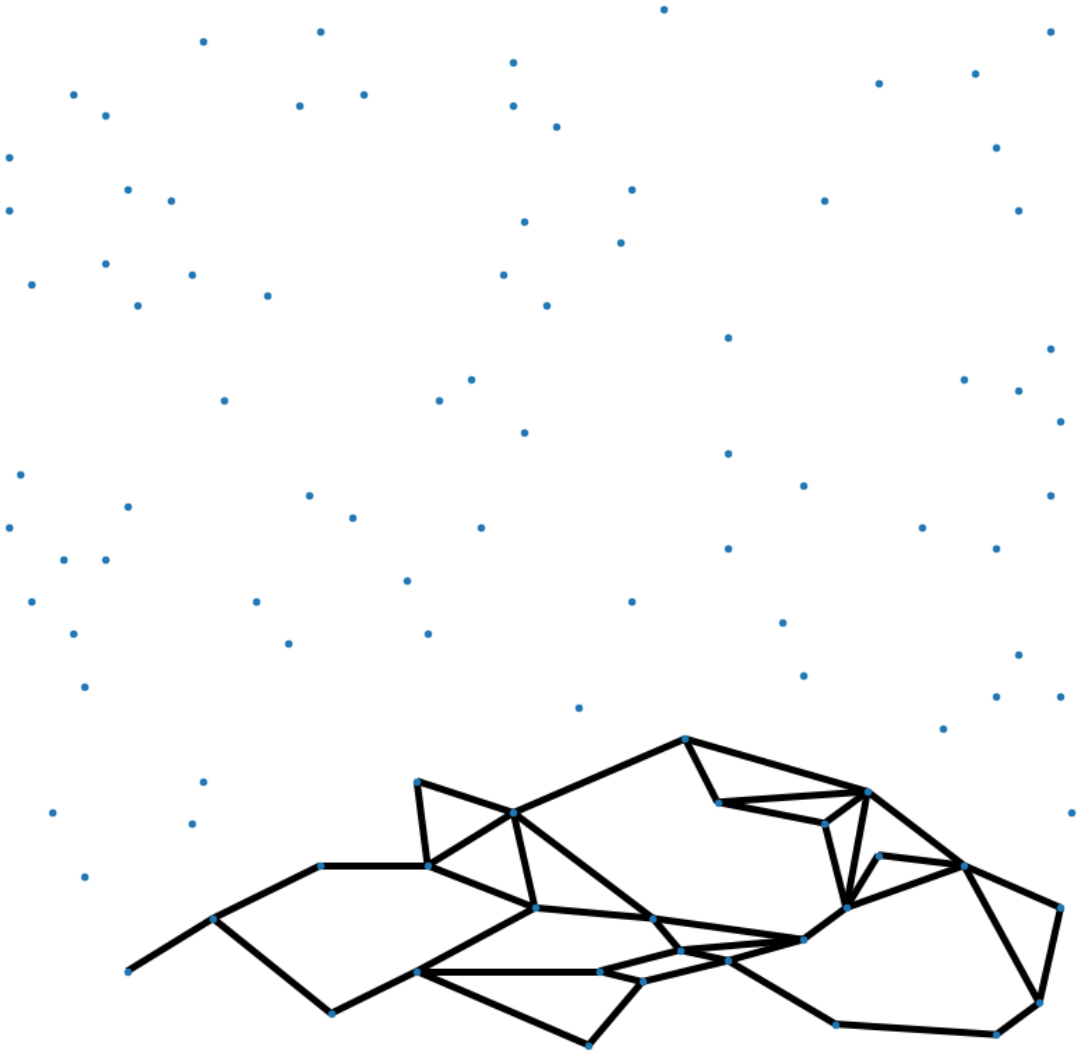
```python
# -*- coding: utf-8 -*-
import pandas as pd

# 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)
```

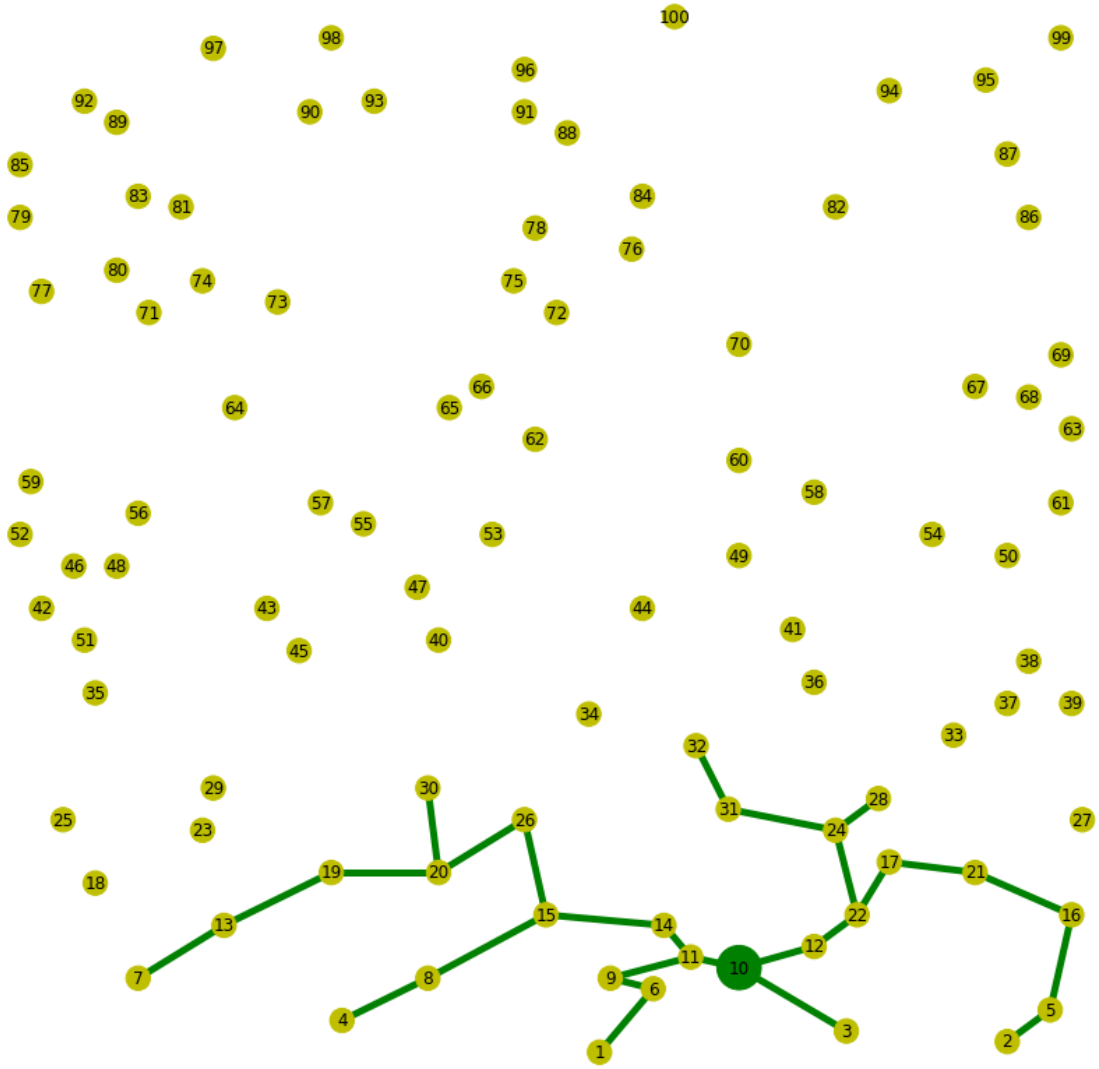
第一组数据：

```python
# -*- 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)



```





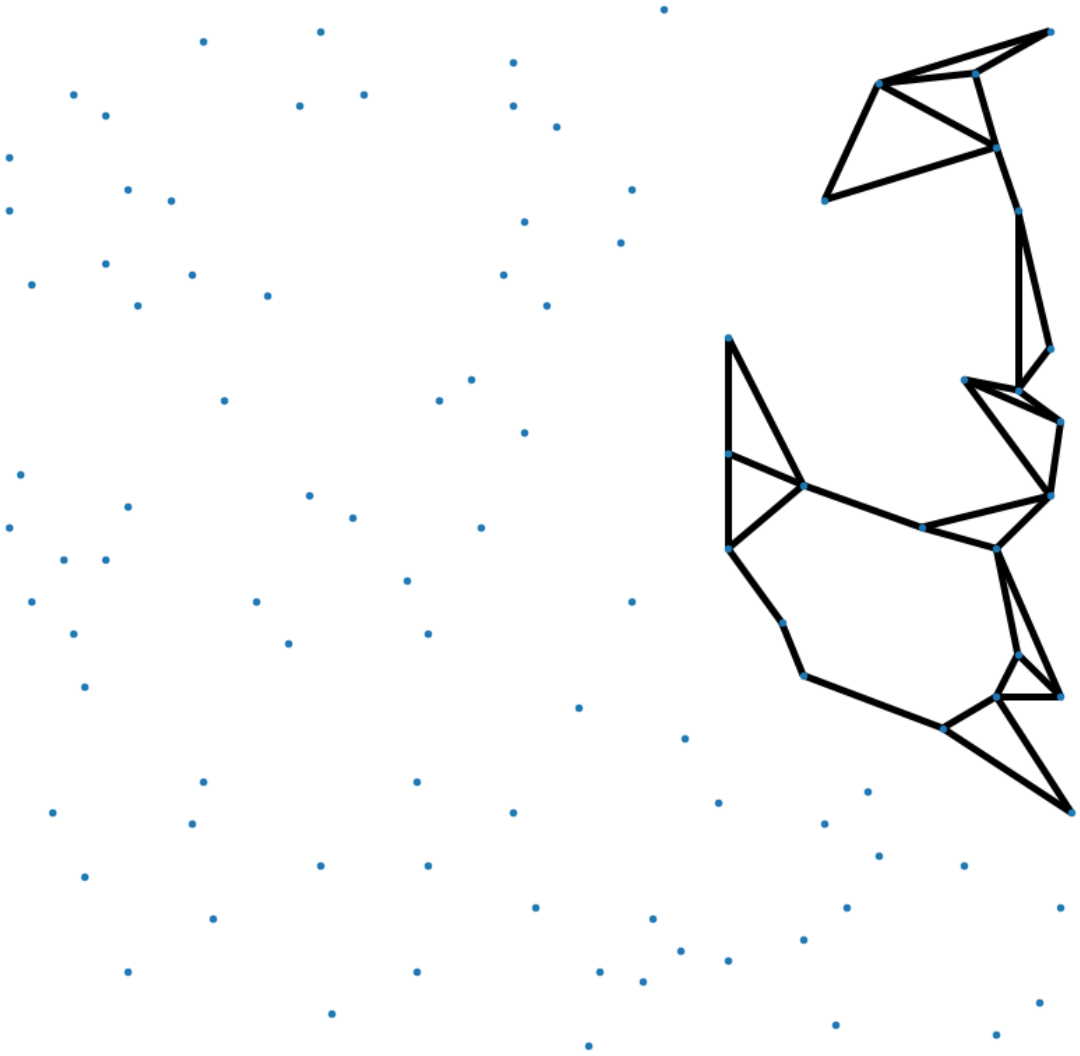

```
最短距离 76990.90151994597
最小生成树长度： 21011.612087891008
```

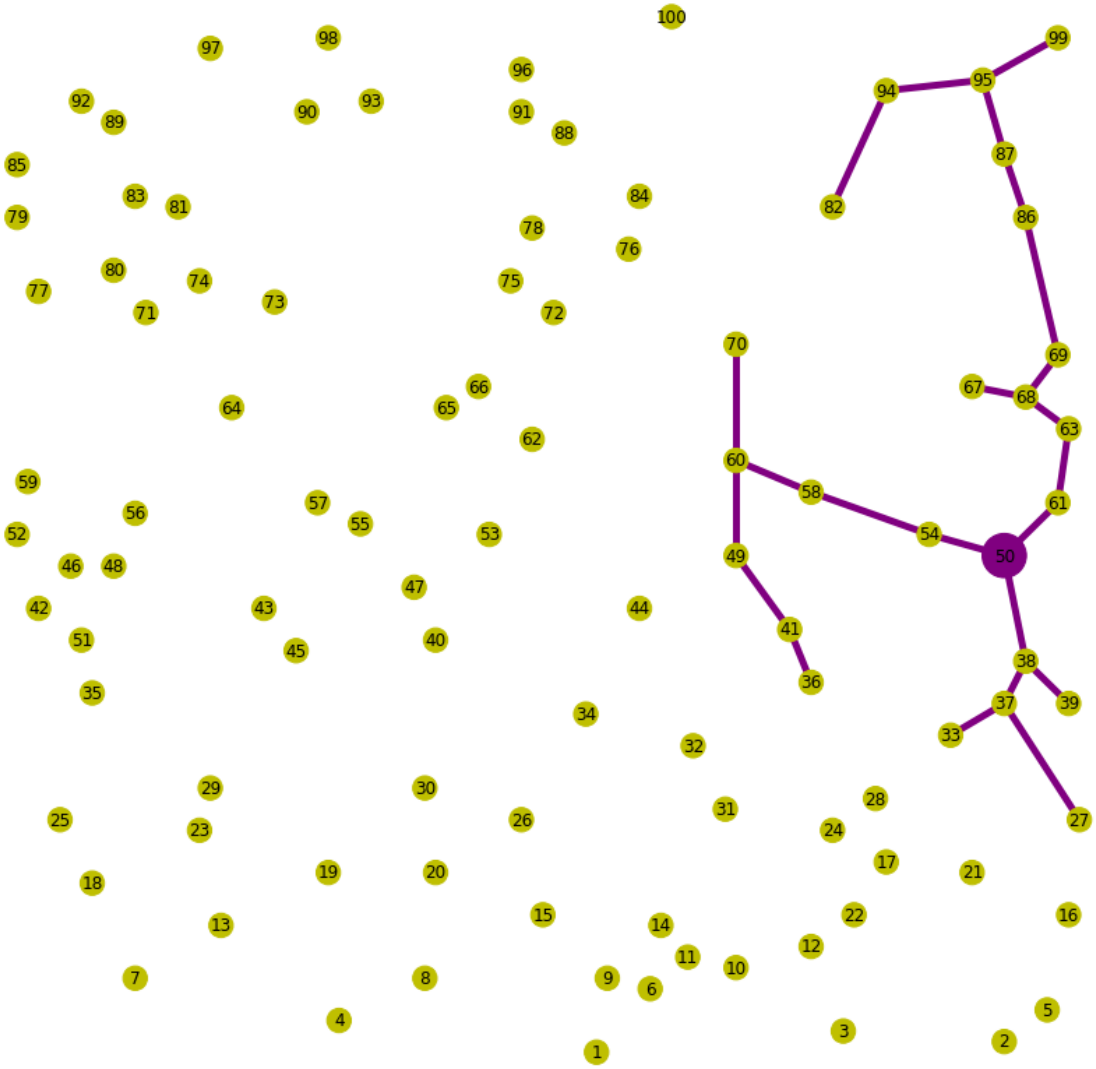
第二组

```

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)


```





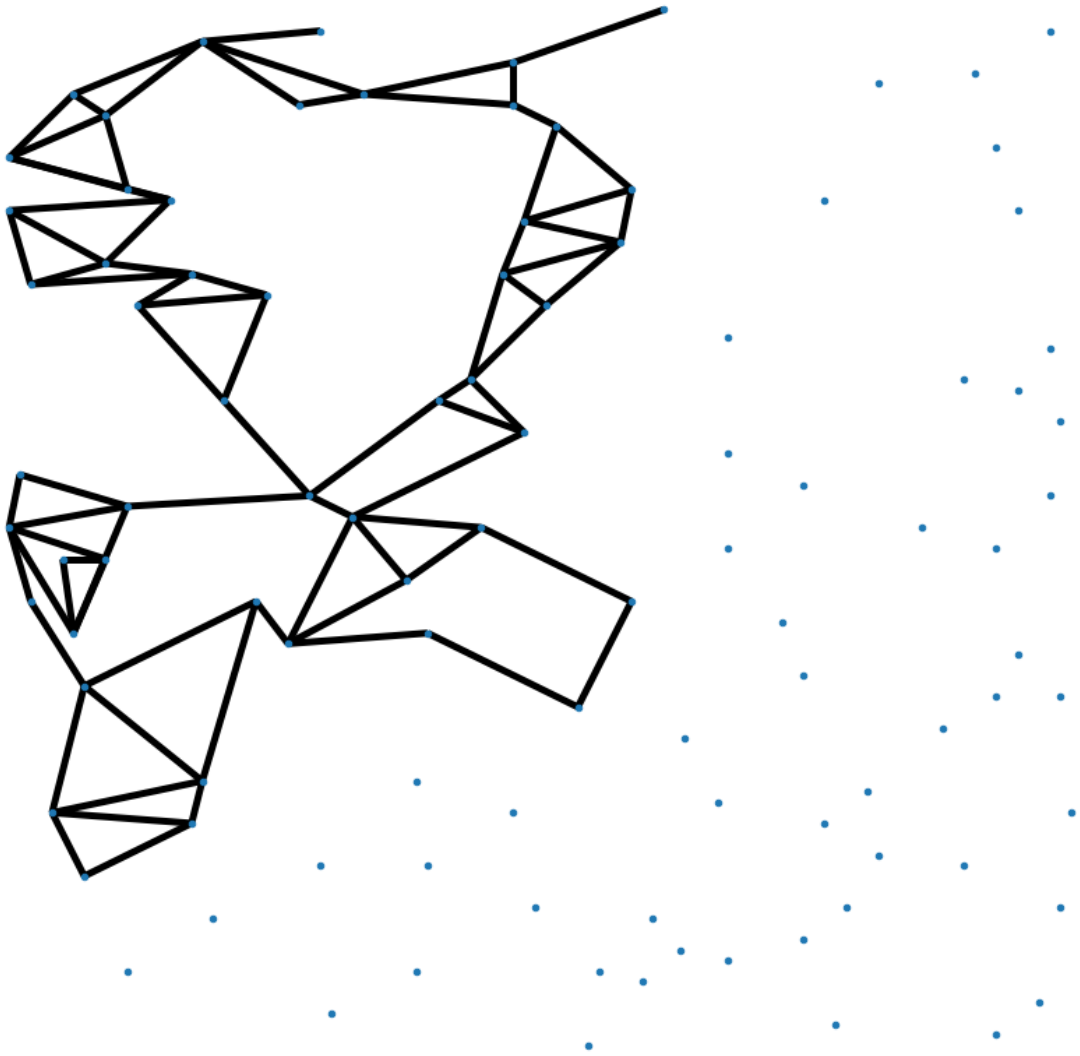

```
72143.11816199898
18517.26321880672
```

第三组数据

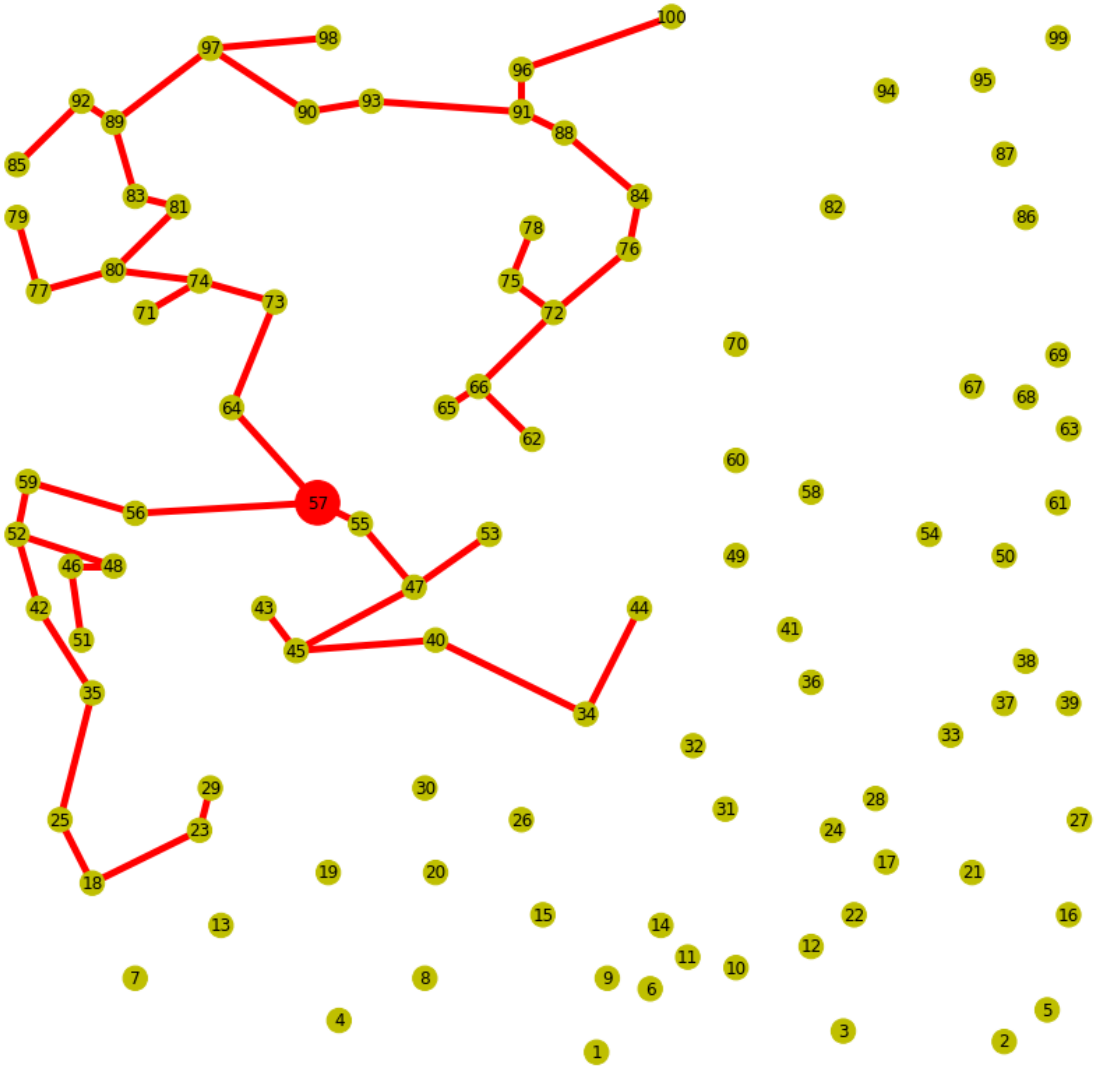
```
# -*- coding: utf-8 -*-
"""
Created on Tue May  2 10:03:10 2023

@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)

```

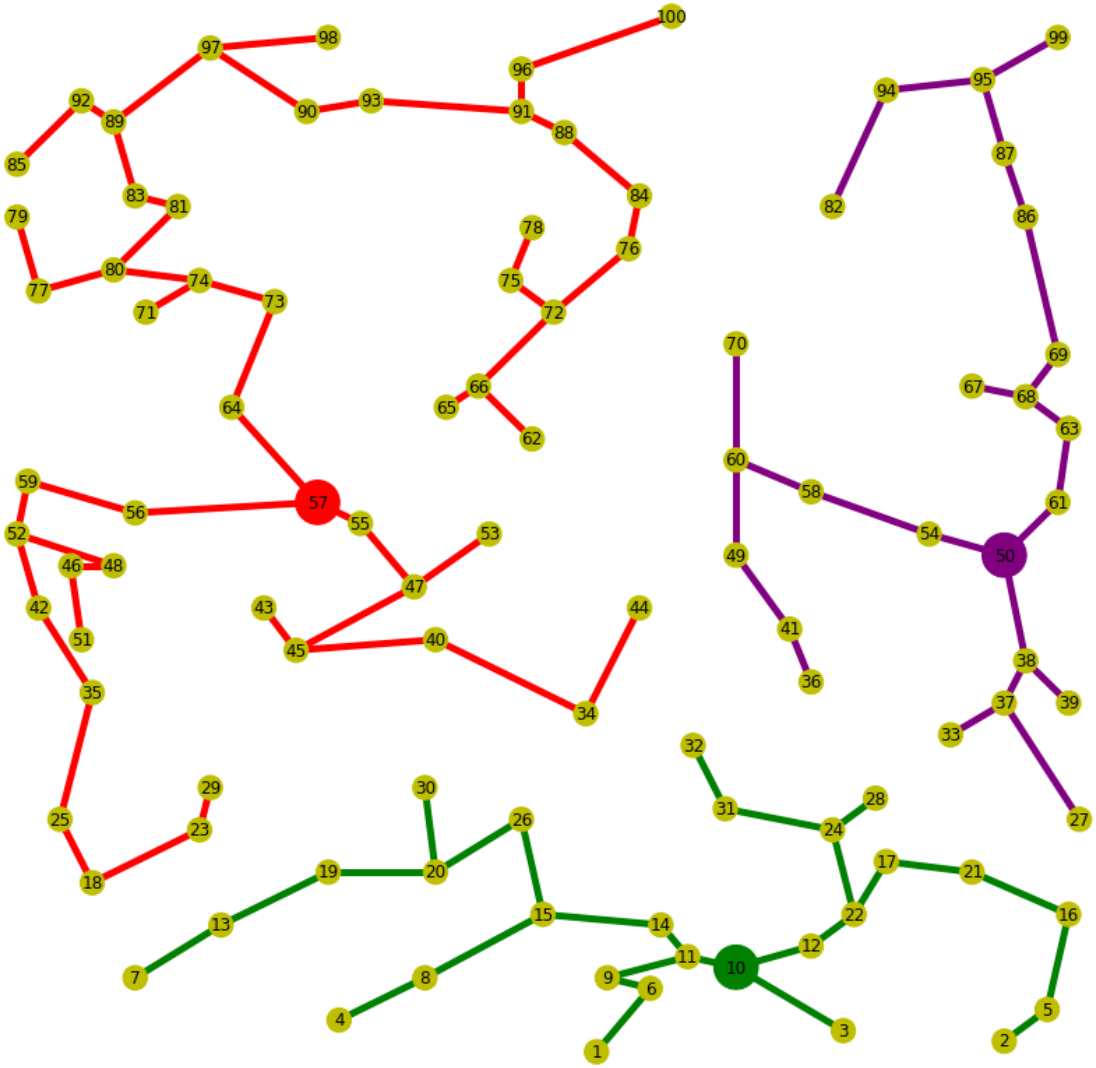


```
317809.1693991888
40425.215152604505
```




```
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