

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

```
1  # -*- coding: utf-8 -*-
2
3  import networkx as nx
4  import pandas as pd
5  import multiprocessing
6  import itertools
7
8  # 读取位置表单
9  locations = pd.read_excel("data1.xlsx", sheet_name="位置",
10                           header=0, index_col=0)
11
12 # 读取连接道路表单
13 roads = pd.read_excel("data.xlsx", sheet_name="连接道路")
14
15 # 创建空图
16 G = nx.Graph()
17
18 # 添加节点
19 for node in locations.index:
20     G.add_node(node, pos=(locations.loc[node, "X"],
21                          locations.loc[node, "Y"]))
22
23 # 添加边
24 for i, row in roads.iterrows():
25     G.add_edge(row["起点"], row["终点"], weight=row["距离"])
26
27 # 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
28 def calculate_distance_sum(G, medical_centers):
29     distance_sum = 0
30     for node in G.nodes:
31         min_distance = float("inf")
```

```

32         for center in medical_centers:
33             distance = nx.shortest_path_length(G, source=node,
target=center, weight="weight")
34             if distance < min_distance:
35                 min_distance = distance
36             distance_sum += min_distance
37         return distance_sum
38
39
40
41 # 定义函数，找到距离总和S1最小的医疗点组合
42 def find_best_medical_centers(G):
43     nodes = list(G.nodes)
44     min_distance_sum = float("inf")
45     best_centers = None
46     # 枚举所有可能的组合情况
47     combinations = [c for c in itertools.combinations(nodes, 3)]
48     a=combinations
49     for centers in a:
50         distance_sum = calculate_distance_sum(G, centers)
51         data=str(centers)+str(distance_sum)+'\n'
52         # 记录距离总和最小的组合
53         file.write(data)
54         if distance_sum < min_distance_sum:
55             min_distance_sum = distance_sum
56             best_centers = centers
57     return best_centers, min_distance_sum
58
59 best_centers, min_distance_sum = find_best_medical_centers(G)
60 print("最优的医疗点组合为: ", best_centers)
61 print("距离总和为: ", min_distance_sum)
62 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：

```
1 | (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。

```
1 import numpy as np
2 import pandas as pd
3 import networkx as nx
4
5 # 读取位置表单
6 locations = pd.read_excel("data1.xlsx", sheet_name="位置",
7                             header=0, index_col=0)
8
9 # 读取连接道路表单
10 roads = pd.read_excel("data.xlsx", sheet_name="连接道路")
11
12 # 构建距离矩阵
13 dist_mat = np.zeros((len(locations), len(locations)))
14 for i, row in roads.iterrows():
15     start_loc = row["起点"]
16     end_loc = row["终点"]
17     distance = row["距离"]
18     start_idx = locations.index.get_loc(start_loc)
19     end_idx = locations.index.get_loc(end_loc)
20     dist_mat[start_idx, end_idx] = distance
21     dist_mat[end_idx, start_idx] = distance
22
23 # 创建连通图
24 G = nx.Graph()
25
26 # 添加节点
27 for node in locations.index:
28     G.add_node(node, pos=(locations.loc[node, "X"],
29                             locations.loc[node, "Y"]))
30
31 # 添加边
32 for i, row in roads.iterrows():
33     G.add_edge(row["起点"], row["终点"], weight=row["距离"])
34 mapping = {node: int(node) for node in G.nodes()}
35 G = nx.relabel_nodes(G, mapping)
36
37 # 找到所有节点到10、50、57节点的最短路径
38 shortest_paths_10 =
39     nx.single_source_dijkstra_path_length(G, 10)
40 shortest_paths_50 =
41     nx.single_source_dijkstra_path_length(G, 50)
```

```

37 shortest_paths_57 =
    nx.single_source_dijkstra_path_length(G,57)
38 shortest_path = nx.shortest_path(G, 4, 10, weight='weight')
39 print(shortest_path)
40
41 # 计算每个节点到10、50、57节点的实际最短距离和路径
42 result = {}
43 road_map={}
44 for node in G.nodes():
45
46     shortest_distance_10 =shortest_paths_10[node]
47     shortest_path_10
    =nx.shortest_path(G,node,10,weight='weight')
48
49     shortest_distance_50 =shortest_paths_50[node]
50     shortest_path_50 =
    nx.shortest_path(G,node,50,weight='weight')
51
52     shortest_distance_57 = shortest_paths_57[node]
53     shortest_path_57 =
    nx.shortest_path(G,node,57,weight='weight')
54
55     shortest_distance = min(shortest_distance_10,
    shortest_distance_50, shortest_distance_57)
56     shortest_target = None
57     shortest_path = []
58     if shortest_distance == shortest_distance_10:
59         shortest_target = 10
60         shortest_path = shortest_path_10
61     elif shortest_distance == shortest_distance_50:
62         shortest_target = 50
63         shortest_path = shortest_path_50
64     elif shortest_distance == shortest_distance_57:
65         shortest_target = 57
66         shortest_path = shortest_path_57
67     # 计算路径上的权重之和
68     weight_sum = 0
69     for i in range(len(shortest_path)-1):
70         u, v = shortest_path[i], shortest_path[i+1]
71         weight_sum += G[u][v]['weight']
72     edges = [(shortest_path[i],shortest_path[i+1]) for i in
    range(len(shortest_path)-1)]
73     sorted_edges = [tuple(sorted(edge)) for edge in edges]
74
75     print(node,sorted_edges)

```

```

76     for i in range(len(sorted_edges)):
77         if sorted_edges[i] not in road_map:
78             road_map[sorted_edges[i]] = 1
79         else:
80             road_map[sorted_edges[i]] += 1
81     result[node] = {'target': shortest_target, 'distance':
shortest_distance, 'path': shortest_path, 'weight':
weight_sum}
82
83     # 将结果存储到Excel文件中
84     df = pd.DataFrame.from_dict(result, orient='index')
85     df.index.name = 'node'
86     df.to_excel('result.xlsx')
87     with open('road_map.txt', 'w') as file:
88         for road in road_map:
89             file.write(f'{road}:{road_map[road]}\n')
90     total_distance = 0
91     for (a, b) in road_map:
92         distance = G.get_edge_data(a, b)['weight'] # 获取边的权值
93         total_distance += distance
94     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 S_2 ，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 S_1 。

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:

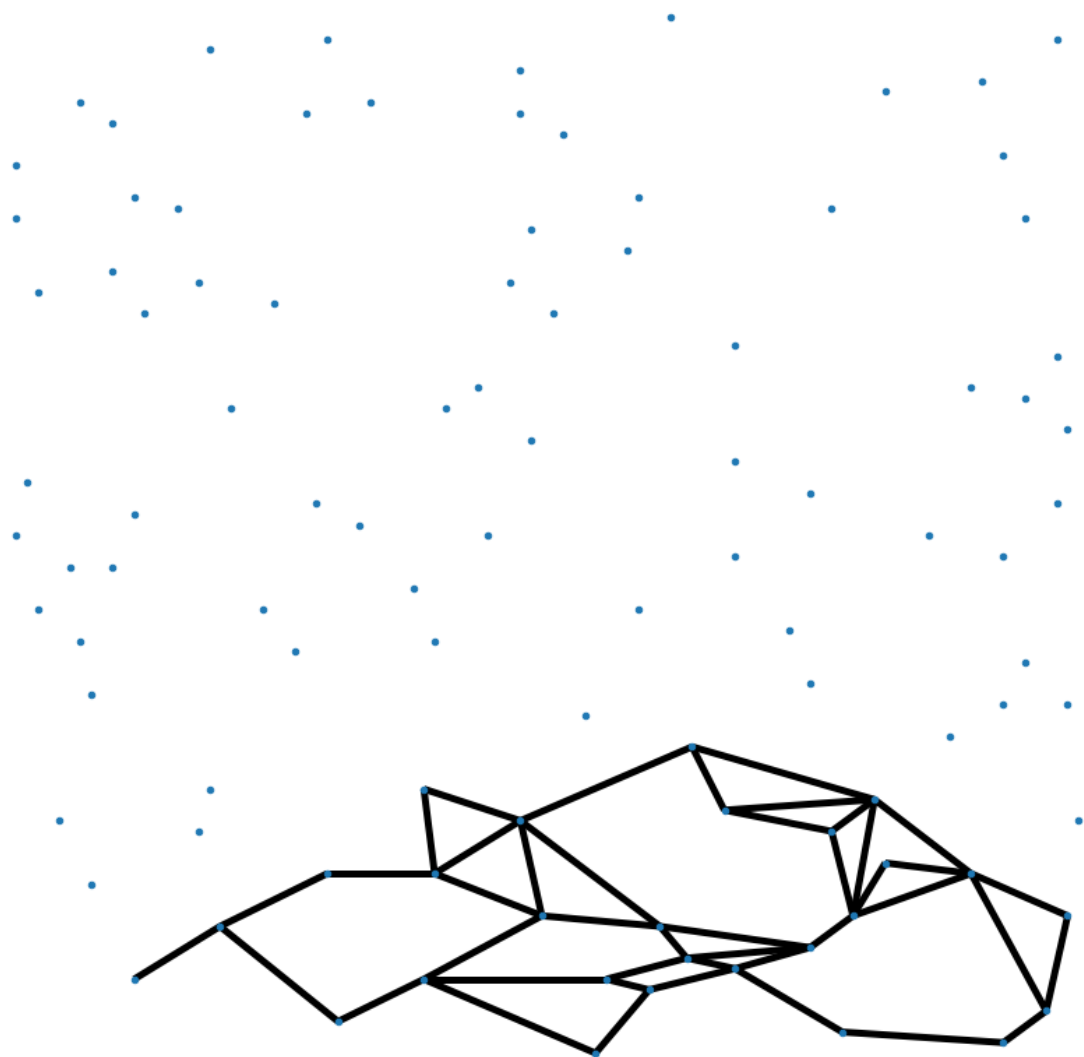
```
1 # -*- coding: utf-8 -*-
2 import pandas as pd
3
4 # 读取 excel 文件
5 df = pd.read_excel('result.xlsx', sheet_name='Sheet1')
6
7 # 按照 target 分组
8 grouped = df.groupby('target')
9
10 # 遍历每个分组，将分组数据保存为单独的 Excel 文件
11 for target, group in grouped:
12     # 构造文件名
13     filename = f'target_{target}.xlsx'
14     # 保存数据
15     group.to_excel(filename, index=False)
```

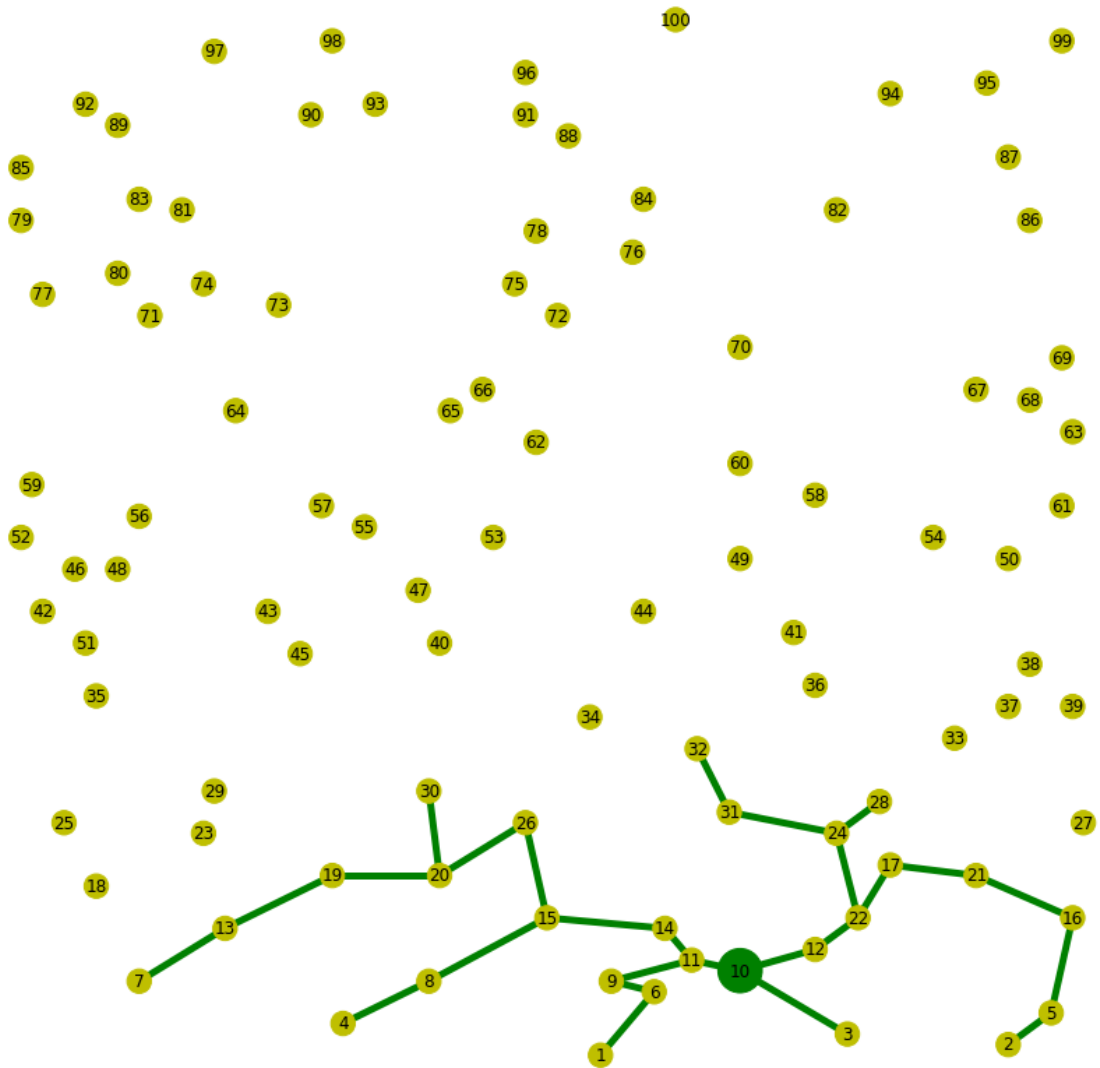
第一组数据:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue May  2 10:03:10 2023
4
5 @author: wangyufan
6 """
7
8
9 import networkx as nx
10 import pandas as pd
11 import numpy as np
12
13 res1 = pd.read_excel("target_10.xlsx", sheet_name="Sheet1",
14 header=0,index_col=0).index
15 locations1 = pd.read_excel("data1.xlsx", sheet_name="位置",
16 header=0,index_col=0)
17
18 # 读取连接道路表单
```

```
16 roads1 = pd.read_excel("data.xlsx", sheet_name="连接道路")
17
18 # 创建空图
19 G = nx.Graph()
20
21 # 添加节点
22
23 for node in locations1.index:
24     #if node in res:
25         G.add_node(node, pos=(locations1.loc[node, "X"],
26 locations1.loc[node, "Y"]))
27
28 # 添加边
29 nodes1=res1
30
31 for i, row in roads1.iterrows():
32
33     if int(row["起点"]) in nodes1 and int(row["终点"]) in
nodes1:
34         G.add_edge(row["起点"], row["终点"], weight=row["距离"])
35
36 # 对节点进行整数转换
37 mapping = {node: int(node) for node in G.nodes()}
38 G = nx.relabel_nodes(G, mapping)
39
40 import matplotlib.pyplot as plt
41
42 # 获取节点位置
43 pos = nx.get_node_attributes(G, "pos")
44 plt.figure(figsize=(15, 15))
45
46 # 绘制节点和边
47 nx.draw_networkx_nodes(G, pos, node_size=20)
48 nx.draw_networkx_edges(G, pos, width=5)
49
50 # 显示图
51 plt.axis("off")
52 plt.show()
53
54 T = nx.minimum_spanning_tree(G)
55
56 # 获取最小生成树的边
57 plt.figure(figsize=(15, 15))
58
59 tree_edges = list(T.edges())
60 node_colors = ['y' if node != 10 else 'g' for node in
G.nodes()]
61 node_sizes = [1000 if node == 10 else 300 for node in
G.nodes()]
```

```
57 # 绘制节点和边，并将最小生成树的边标记为红色
58 nx.draw_networkx_nodes(G, pos, node_color=node_colors,
59 node_size=node_sizes)
60 nx.draw_networkx_labels(G, pos, {n: str(n) for n in
61 G.nodes()})
62 nx.draw_networkx_edges(G, pos, edgelist=tree_edges,
63 edge_color='g', width=5)
64
65 # 显示图
66 plt.axis("off")
67 plt.show()
68
69 s_10=0
70 for node in nodes1:
71     distance = nx.shortest_path_length(T, source=node,
72 target=10, weight='weight')
73     s_10+=distance
74
75 print("最短距离", s_10)
76
77 length=0
78 for (a,b) in tree_edges:
79     length+=G.get_edge_data(a,b)['weight']
80
81 print("最小生成树长度: ", length)
```



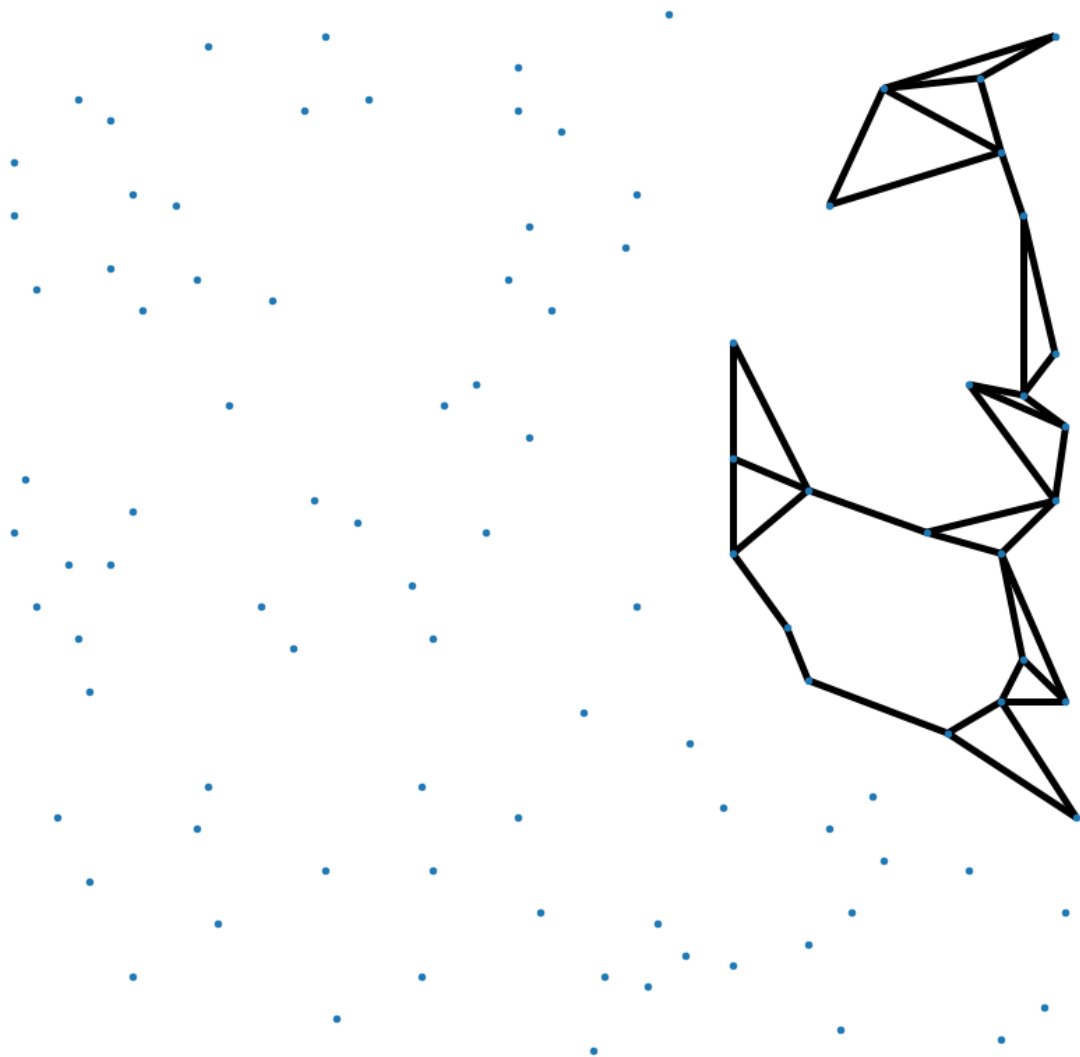
- 1 最短距离 76990.90151994597
- 2 最小生成树长度: 21011.612087891008

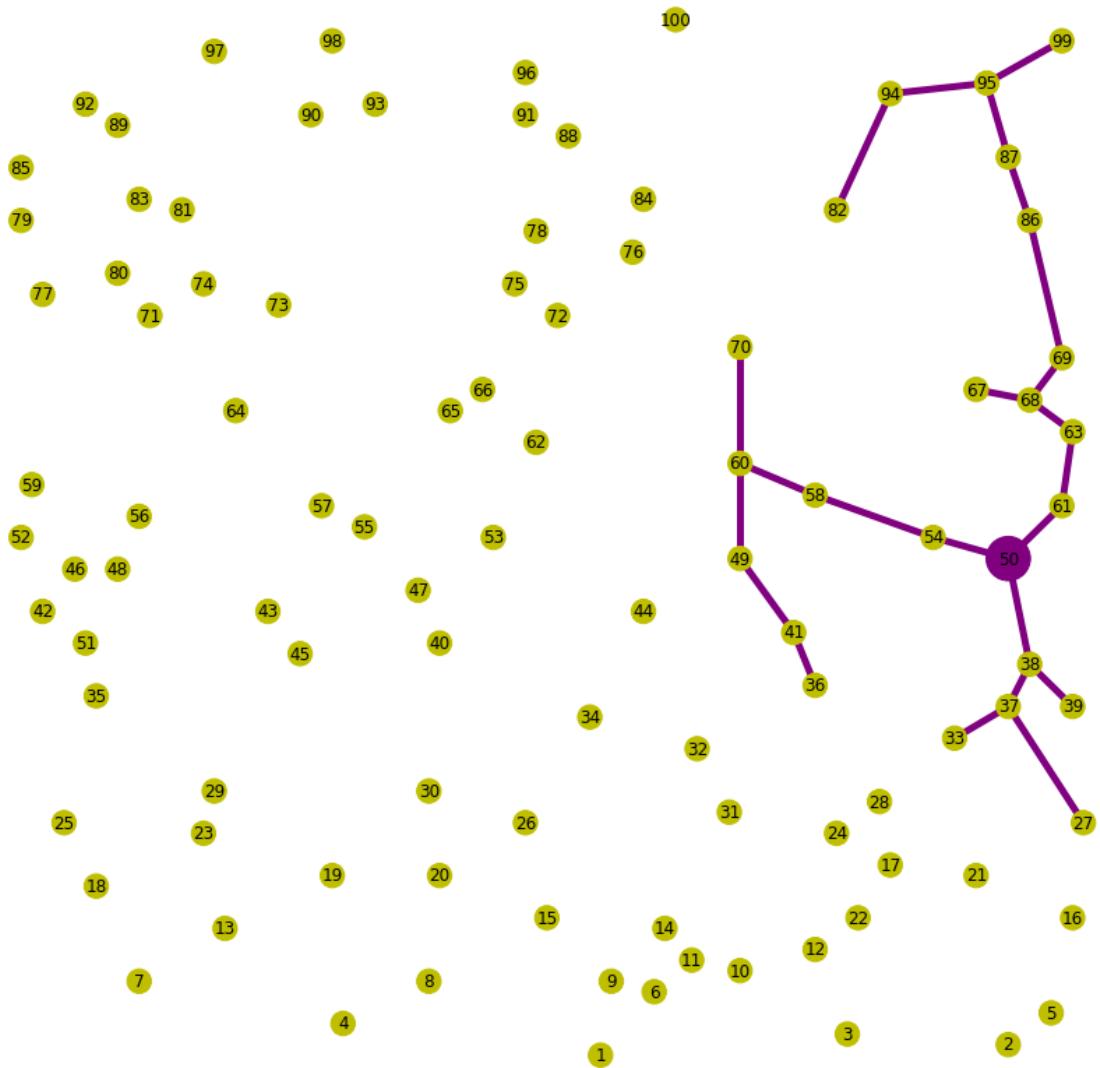
第二组

```
1
2 import networkx as nx
3 import pandas as pd
4 import numpy as np
5
6 res1 = pd.read_excel("target_50.xlsx", sheet_name="Sheet1",
7 header=0,index_col=0).index
8 locations1 = pd.read_excel("data1.xlsx", sheet_name="位置",
9 header=0,index_col=0)
10 # 读取连接道路表单
11 roads1 = pd.read_excel("data.xlsx", sheet_name="连接道路")
12 # 创建空图
```

```
12 G = nx.Graph()
13
14 # 添加节点
15
16 for node in locations1.index:
17     #if node in res:
18         G.add_node(node, pos=(locations1.loc[node, "X"],
19 locations1.loc[node, "Y"]))
19
20 # 添加边
21 nodes1=res1
22 for i, row in roads1.iterrows():
23
24     if int(row["起点"]) in nodes1 and int(row["终点"]) in
nodes1:
25         G.add_edge(row["起点"], row["终点"], weight=row["距离"])
26 # 对节点进行整数转换
27 mapping = {node: int(node) for node in G.nodes()}
28 G = nx.relabel_nodes(G, mapping)
29
30 import matplotlib.pyplot as plt
31
32 # 获取节点位置
33 plt.figure(figsize=(15, 15))
34 pos = nx.get_node_attributes(G, "pos")
35
36 # 绘制节点和边
37 nx.draw_networkx_nodes(G, pos, node_size=20)
38 nx.draw_networkx_edges(G, pos, width=5)
39
40 # 显示图
41 plt.axis("off")
42 plt.show()
43
44 T = nx.minimum_spanning_tree(G)
45 # 获取最小生成树的边
46 tree_edges = list(T.edges())
47
48 plt.figure(figsize=(15, 15))
49
50 tree_edges = list(T.edges())
51 node_colors = ['y' if node != 50 else 'purple' for node in
G.nodes()]
52 node_sizes = [1000 if node == 50 else 300 for node in
G.nodes()]
```

```
53 # 绘制节点和边，并将最小生成树的边标记为红色
54 nx.draw_networkx_nodes(G, pos, node_color=node_colors,
55 node_size=node_sizes)
56 nx.draw_networkx_labels(G, pos, {n: str(n) for n in
57 G.nodes()})
58 nx.draw_networkx_edges(G, pos, edgelist=tree_edges,
59 edge_color='purple', width=5)
60 # 显示图
61 plt.axis("off")
62 plt.show()
63
64 s_50=0
65 for node in nodes1:
66     distance = nx.shortest_path_length(T, source=node,
67 target=50, weight='weight')
68     s_50+=distance
69
70 print(s_50)
71
72 length=0
73 for (a,b) in tree_edges:
74     length+=G.get_edge_data(a,b)['weight']
75
76 print(length)
```





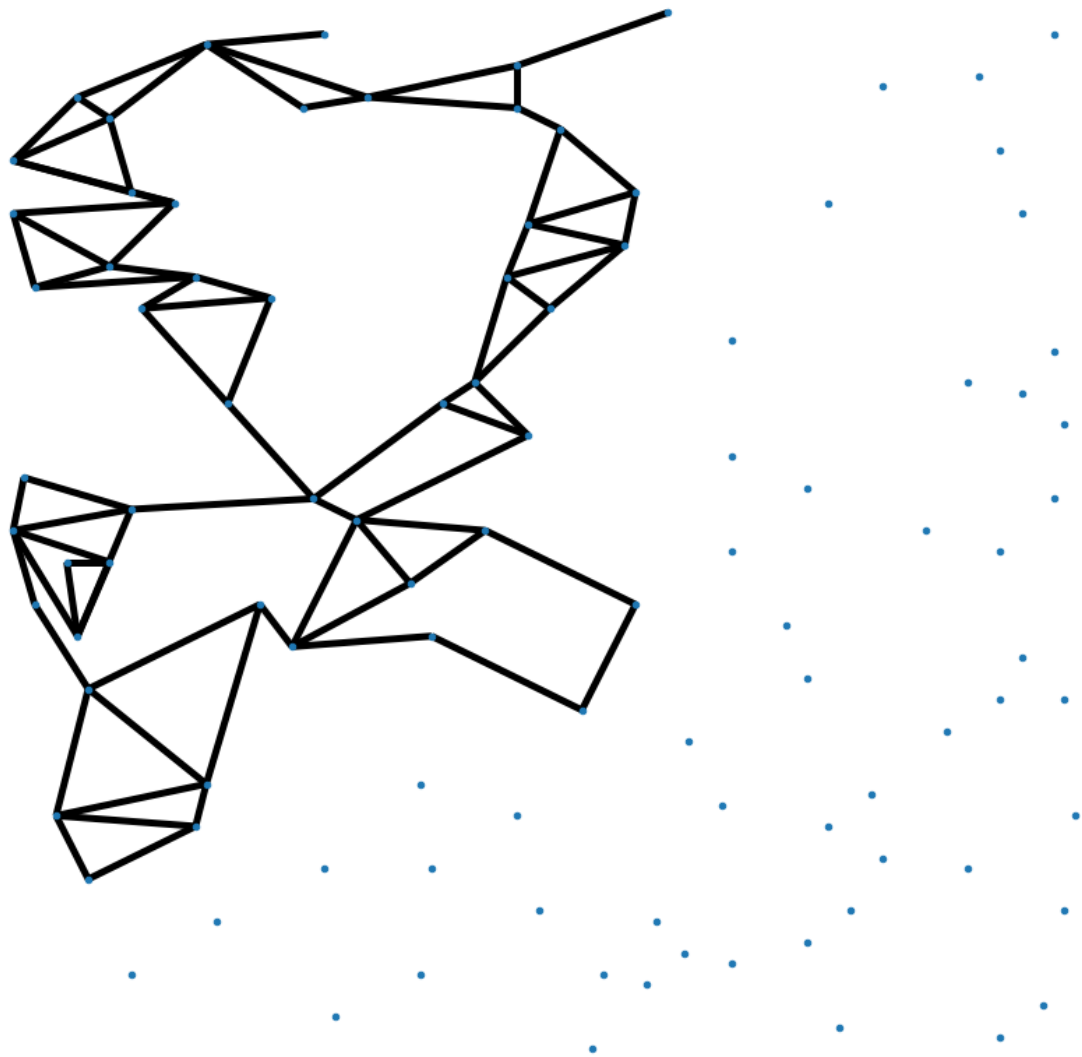
```
1 72143.11816199898
2 18517.26321880672
```

第三组数据

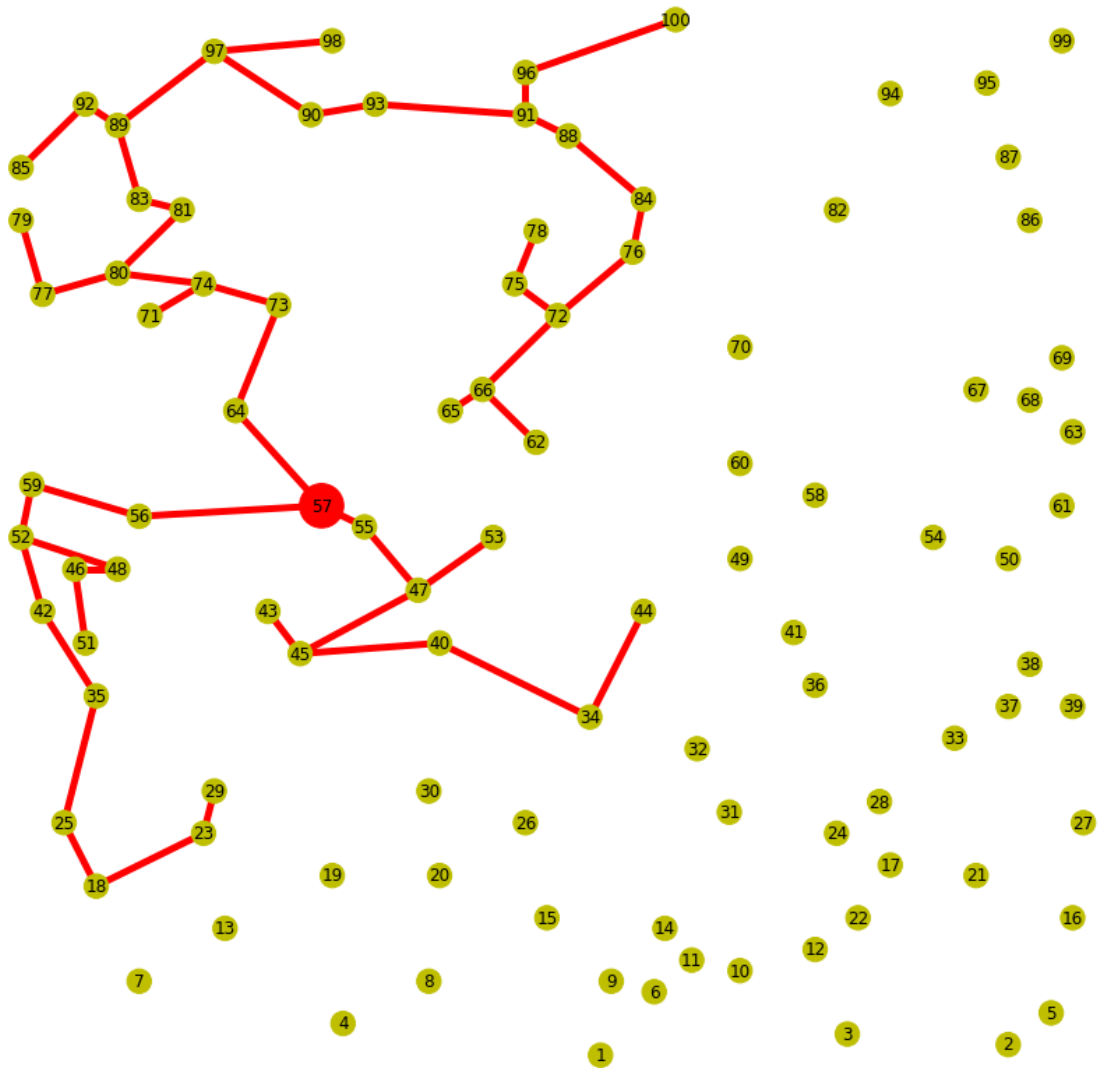
```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue May 2 10:03:10 2023
4
5 @author: wanguyufan
6 """
7
8
9 import networkx as nx
10 import pandas as pd
11 import numpy as np
12
```

```
13 res1 = pd.read_excel("target_57.xlsx", sheet_name="Sheet1",
14 header=0,index_col=0).index
15 locations1 = pd.read_excel("data1.xlsx", sheet_name="位置",
16 header=0,index_col=0)
17
18 # 读取连接道路表单
19 roads1 = pd.read_excel("data.xlsx", sheet_name="连接道路")
20
21 # 创建空图
22 G = nx.Graph()
23
24 # 添加节点
25 for node in locations1.index:
26     #if node in res:
27         G.add_node(node, pos=(locations1.loc[node, "X"],
28 locations1.loc[node, "Y"]))
29
30 # 添加边
31 nodes1=res1
32 for i, row in roads1.iterrows():
33
34     if int(row["起点"]) in nodes1 and int(row["终点"]) in
35 nodes1:
36         G.add_edge(row["起点"], row["终点"], weight=row["距离"])
37
38 # 对节点进行整数转换
39 mapping = {node: int(node) for node in G.nodes()}
40 G = nx.relabel_nodes(G, mapping)
41
42
43 import matplotlib.pyplot as plt
44 plt.figure(figsize=(15, 15))
45
46 # 获取节点位置
47 pos = nx.get_node_attributes(G, "pos")
48
49
50 # 绘制节点和边
51 nx.draw_networkx_nodes(G, pos, node_size=20)
52 nx.draw_networkx_edges(G, pos, width=5)
53
54
55 # 显示图
56 plt.axis("off")
57 plt.show()
58
59 T = nx.minimum_spanning_tree(G)
60
61 # 获取最小生成树的边
62 plt.figure(figsize=(15, 15))
63
```

```
54 tree_edges = list(T.edges())
55 node_colors = ['y' if node != 57 else 'red' for node in
G.nodes()]
56 node_sizes = [1000 if node == 57 else 300 for node in
G.nodes()]
57 # 绘制节点和边，并将最小生成树的边标记为红色
58 nx.draw_networkx_nodes(G, pos, node_color=node_colors,
node_size=node_sizes)
59 nx.draw_networkx_labels(G, pos, {n: str(n) for n in
G.nodes()})
60 nx.draw_networkx_edges(G, pos, edgelist=tree_edges,
edge_color='r', width=5)
61
62 # 显示图
63 plt.axis("off")
64 plt.show()
65
66 print(len(nodes1))
67 print(nodes1)
68 s_57=0
69 for node in nodes1:
70     distance = nx.shortest_path_length(T, source=node,
target=57, weight='weight')
71     s_57+=distance
72
73 print(s_57)
74
75 length=0
76 for (a,b) in tree_edges:
77     length+=G.get_edge_data(a,b)['weight']
78
79
80 print(length)
81
```

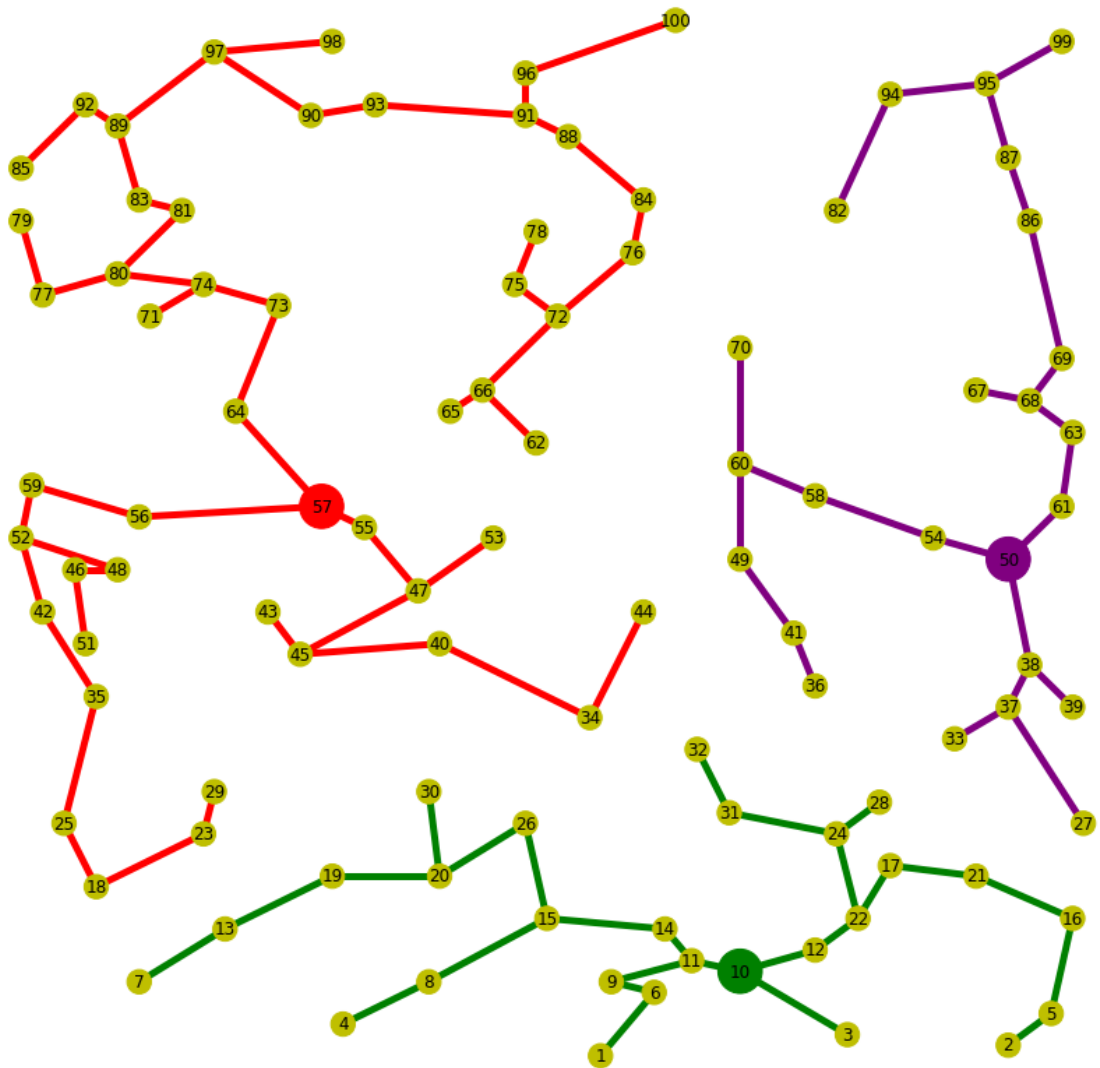



1	317809.1693991888
2	40425.215152604505



1	317809.1693991888
2	40425.215152604505

总和~哈哈画的有点丑



总距离:

$$s1=21011.612087891008+18517.26321880672+40425.215152604505=79954.09045930223$$

$$s2=76990.90151994597+72143.11816199898+317809.1693991888=467943.18908113375$$