华东师范大学数据科学与工程学院上机实践报告

课程名称：算法设计与分析　　**年级：**22 级　　**上机实践成绩：**

指导教师：金澈清　　**姓名：**郭夏辉

上机实践名称：国家电网超高压网络规划　　**学号：**10211900416　　**上机实践日期：**

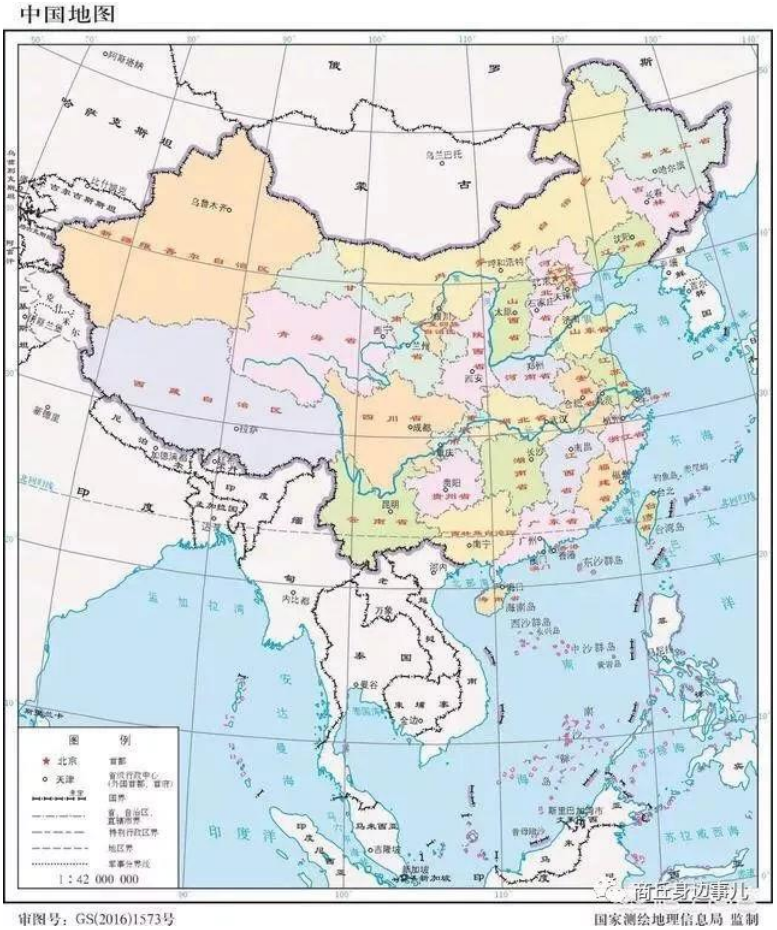
上机实践编号：No.12　　**组号：**1-416　　2023 年 5 月 25 日

一、目的

1．熟悉算法设计的基本思想

2．掌握最小生成树算法的思路

二、内容与设计思想



国家电网公司想在全国布局超高压输电网络，联通所有省会城市。为了降低成本，并且达到某些硬性要求，国家电网按照以下五种策略进行规划布局。

（1）要求整个电网的长度最短。

（2）要求在西宁与郑州拉一根直达专线的情况下，使得整个电网长度最短

（3）要求不仅在西宁与郑州之间拉直达专线，还在杭州与长沙之间拉直达专线的情况下，使得整个电网长度最短。

（4）在香港与澳门、澳门与广州不拉直达线路的前提之下，使得整个电网的长度最短。

（5）山东、河南、山西、甘肃、青海、新疆以及比他们更北的省份称为北方省份，其余省份称为南方省份。如果在南方省份和北方省份之间仅规划一条直通专线，如何使得整个电网的

长度最短。

请分别根据这五种情况计算最优情况。

提示：

1. 如无特殊约定，各个城市之间均可拉专线，其长度是直线长度。

2. 地球上任意两点之间的距离计算方法可以参照以下文件：

https://www.cnblogs.com/ycsfwhh/archive/2010/12/20/1911232.html

摘录如下：

地球是一个近乎标准的椭球体，它的赤道半径为 6378.140 千米，极半径为 6356.755 千米，平均半径 6371.004 千米。如果我们假设地球是一个完美的球体，那么它的半径就是地球的平均半径，记为 R。如果以 0 度经线为基 准，那么根据地球表面任意两点的经纬度就可以计算出这两点间的地表距离（这里忽略地球表面地形对计算带来的误差，仅仅是理论上的估算值）。设第一点 A 的经 纬度为(LonA, LatA)，第二点 B 的经纬度为(LonB, LatB)，按照 0 度经线的基准，东经取经度的正值(Longitude)，西经取经度负值(-Longitude)，北纬取 90-纬度值(90- Latitude)，南纬取 90+纬度值(90+Latitude)，则经过上述处理过后的两点被计为(MLonA, MLatA)和(MLonB, MLatB)。那么根据三角推导，可以得到计算两点距离的如下公式：

C = sin(MLatA)*sin(MLatB)*cos(MLonA-MLonB) + cos(MLatA)*cos(MLatB)

Distance = R*Arccos(C)*Pi/180

这里，R 和 Distance 单位是相同，如果是采用 6371.004 千米作为半径，那么 Distance 就是千米为单位，如果要使用其他单位，比如 mile，还需要做单位换算，1 千米=0.621371192mile

如果仅对经度作正负的处理，而不对纬度作 90-Latitude(假设都是北半球，南半球只有澳洲具有应用意义)的处理，那么公式将是：

C = sin(LatA)*sin(LatB) + cos(LatA)*cos(LatB)*cos(MLonA-MLonB)

Distance = R*Arccos(C)*Pi/180

以上通过简单的三角变换就可以推出。

3. 全国省会城市的经纬度如下所示。

城市,经度,纬度
沈阳市,123.429092,41.796768
长春市,125.324501,43.886841
哈尔滨市,126.642464,45.756966
北京市,116.405289,39.904987
天津市,117.190186,39.125595
呼和浩特市,111.751990,40.841490
银川市,106.232480,38.486440
太原市,112.549248,37.857014
石家庄市,114.502464,38.045475
济南市,117.000923,36.675808
郑州市,113.665413,34.757977
西安市,108.948021,34.263161
武汉市,114.298569,30.584354
南京市,118.76741,32.041546
合肥市,117.283043,31.861191
上海市,121.472641,31.231707
长沙市,112.982277,28.19409
南昌市,115.892151,28.676493
杭州市,120.15358,30.287458
福州市,119.306236,26.075302
广州市,113.28064,23.125177

台北市,121.5200760,25.0307240
海口市,110.199890,20.044220
南宁市,108.320007,22.82402
重庆市,106.504959,29.533155
昆明市,102.71225,25.040609
贵阳市,106.713478,26.578342
成都市,104.065735,30.659462
兰州市,103.834170,36.061380
西宁市,101.777820,36.617290
拉萨市,91.11450,29.644150
乌鲁木齐市,87.616880,43.826630
香港,114.165460,22.275340
澳门,113.549130,22.198750

三、使用环境

推荐使用 C/C++集成编译环境。

四、实验过程

1. 编写相关实验代码
2. 写出算法的思路。
3. 输出各组实验的电网数据表，以及电网总长度，并且通过可视化方式进行呈现。

本次实验是关于最小生成树的，结合所学知识，我先来年回顾一下相关的知识。

Kruskal 算法

```
MST-KRUSKAL(G,w)
1  A=∅
2  for each vertex v∈G.V
3      MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge(u,v)∈G.E,taken in nondecreasing order by weight
6      if FIND-SET(v)≠FIND-SET(v)
7          A=A∪{(u,v)}
8          UNION(u,v)
9  return A
```

整体来说，Kruskal 算法使用的是贪心算法，每次选择的是权值最小且选择的边的两端点不能在同一棵生成树中。至于具体的证明，这里不详细展开了，课本上有详细的论述。

我们使用一个不相交集合数据结构来维护几个互不相交的元素集合，每个集合代表当前森林中的一棵树。FIND-SET(u)查询的是结点 u 所属的生成树，UNION 操作是对两棵生成树进行合并操作。

Prim 算法

```
MST-PRIM(G,w,r)
1  for each u∈G.V
2      u.key=∞
3      u.π=NIL
4  r.key=0
5  Q=G.V
6  while Q≠∅
7      u=EXTRACT-MIN(Q)
8      for each v∈G.Adj[u]
9      if v∈ Q and w(u,v)<v.key
10         v.π=u
11         v.key=w(u,v)
```

Prim 算法的思想更简单了，它的工作原理其实特别像 Dijkstra 算法。从任意一个根节点开始，Prim 算法维护了一个集合 A，这个 A 便是构建的一棵生成树的集合。它每次迭代是怎么拓展呢？其实就是在连接 A 和 A 之外的所有边中，选择一条权重最小的边加入 A,利用课本中介绍的相关性质，最终我们便可以得到需要的最短生成树了。

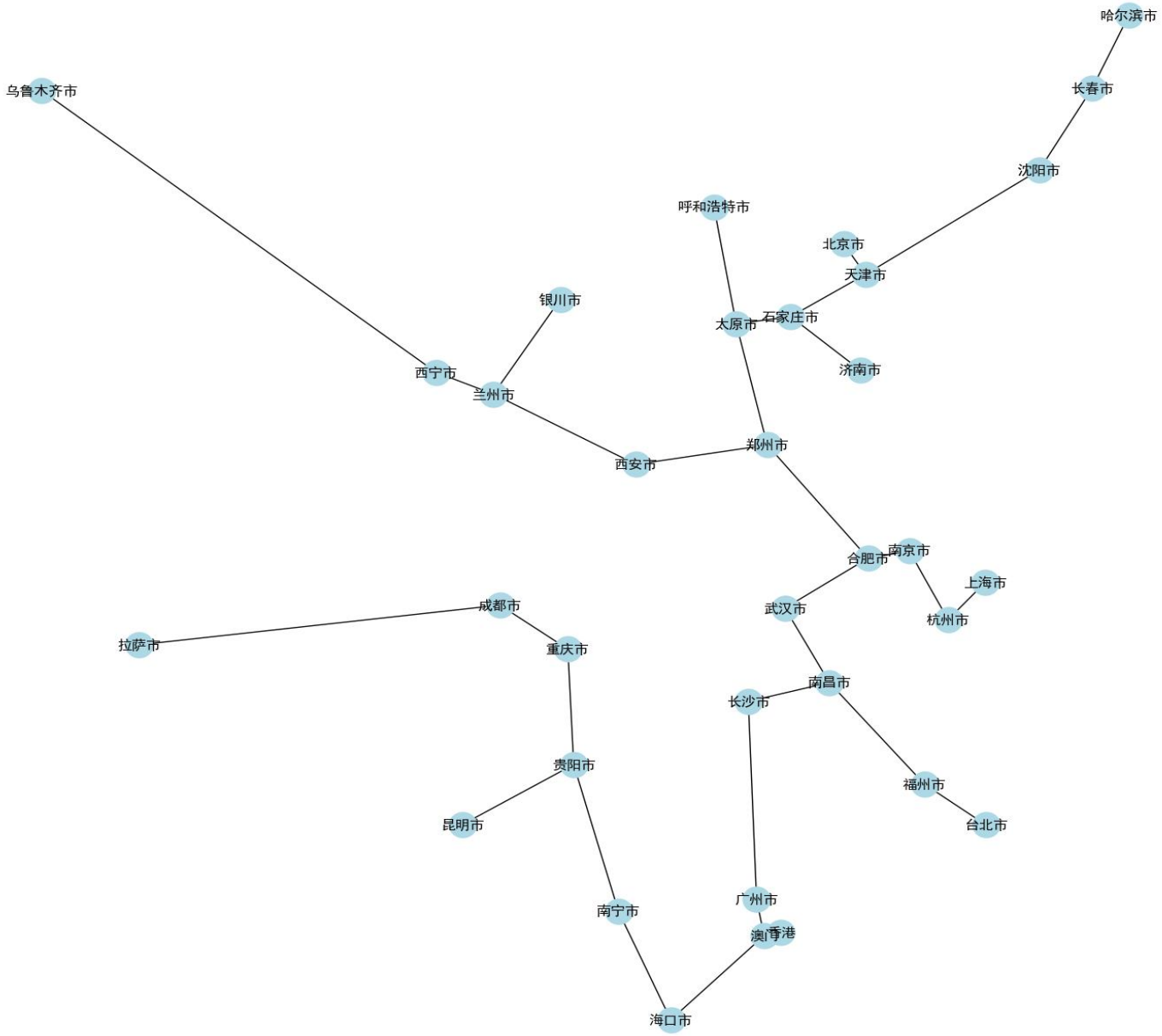
可以发现，就是 Prim 算法每次都要在 Q 中查询最短的边，如果采用朴素的方法一个一个地枚举，显然时间复杂度是不可接受的，我们可以使用优先队列来优化问题的求解，这样 Prim 算法地时间复杂度为 O(ElgV),渐进意义上与 Kruskal 算法一致。从工程的角度上来说，Kruskal 算法更适合稀疏图，而 Prim 算法更适合稠密图。

接下来我就一个问题接着一个问题的完成本次实验吧。有个问题就是我在最初算两个城市之间的距离时，算出来的竟然是负数。经过仔细的核查，这个问题出现在 cmath 库中的 sin,cos 等三角函数，输入应该是弧度而并不是角度。

(1) 要求整个电网的长度最短。

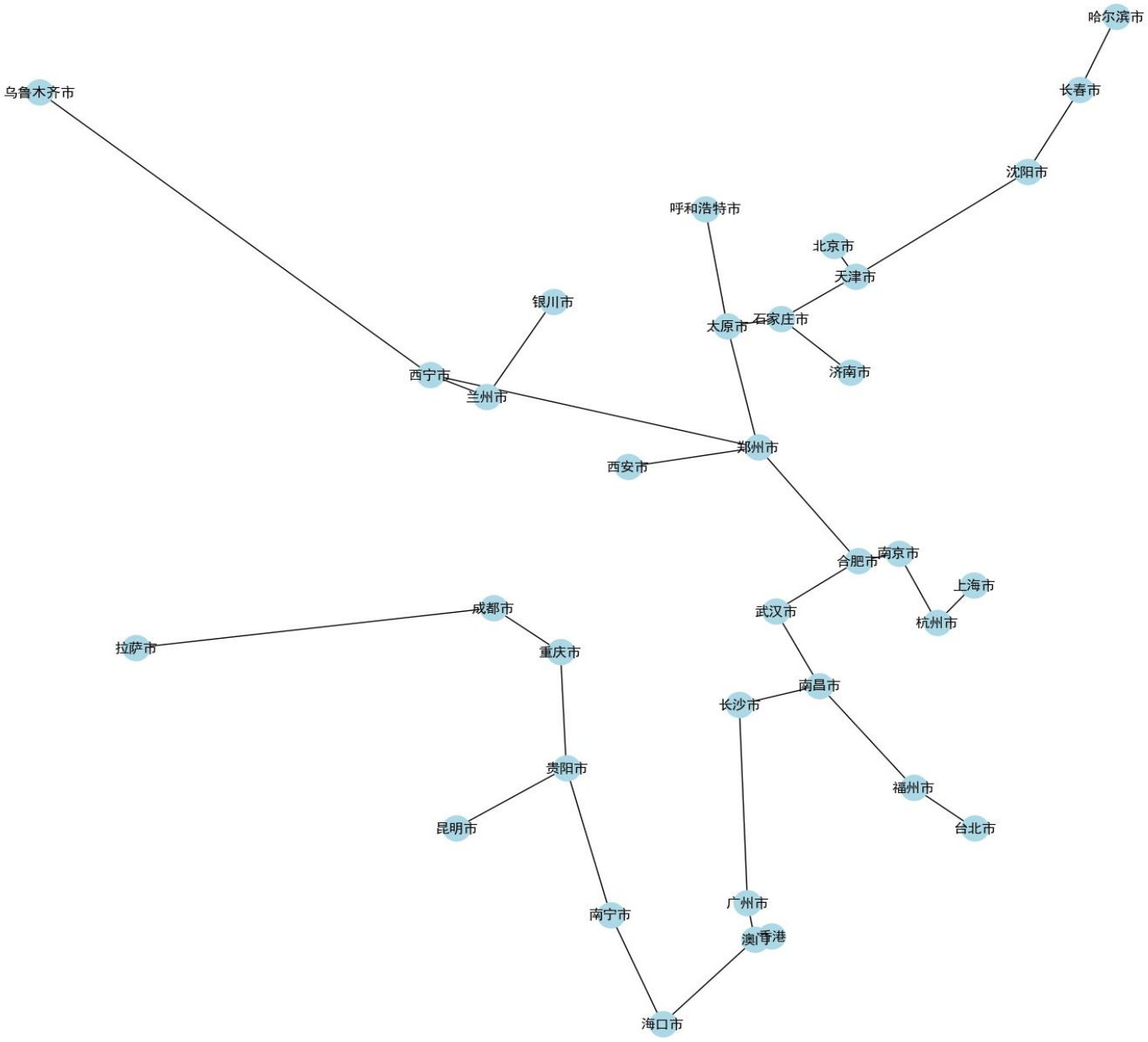
这个其实就是实现一个最短生成树就好了，我采用的是 Prim 算法，重点是输入、测算两城市之间的距离。运行结果如下所示:……SUM LEN: 12369.5km

由于篇幅有限，具体的数据我放在了一个专门的文档中供参考，接下来我以可视化的方式展示结果：



(2)要求在西宁与郑州拉一根直达专线的情况下，使得整个电网长度最短。

这个问题只需要在(1)的算法运行之前将西宁与郑州加入到树中，并且更新对应的相邻结点即可。运行结果如下所示:……SUM LEN: 12956.1km，接下来我以可视化的方式展示结果：

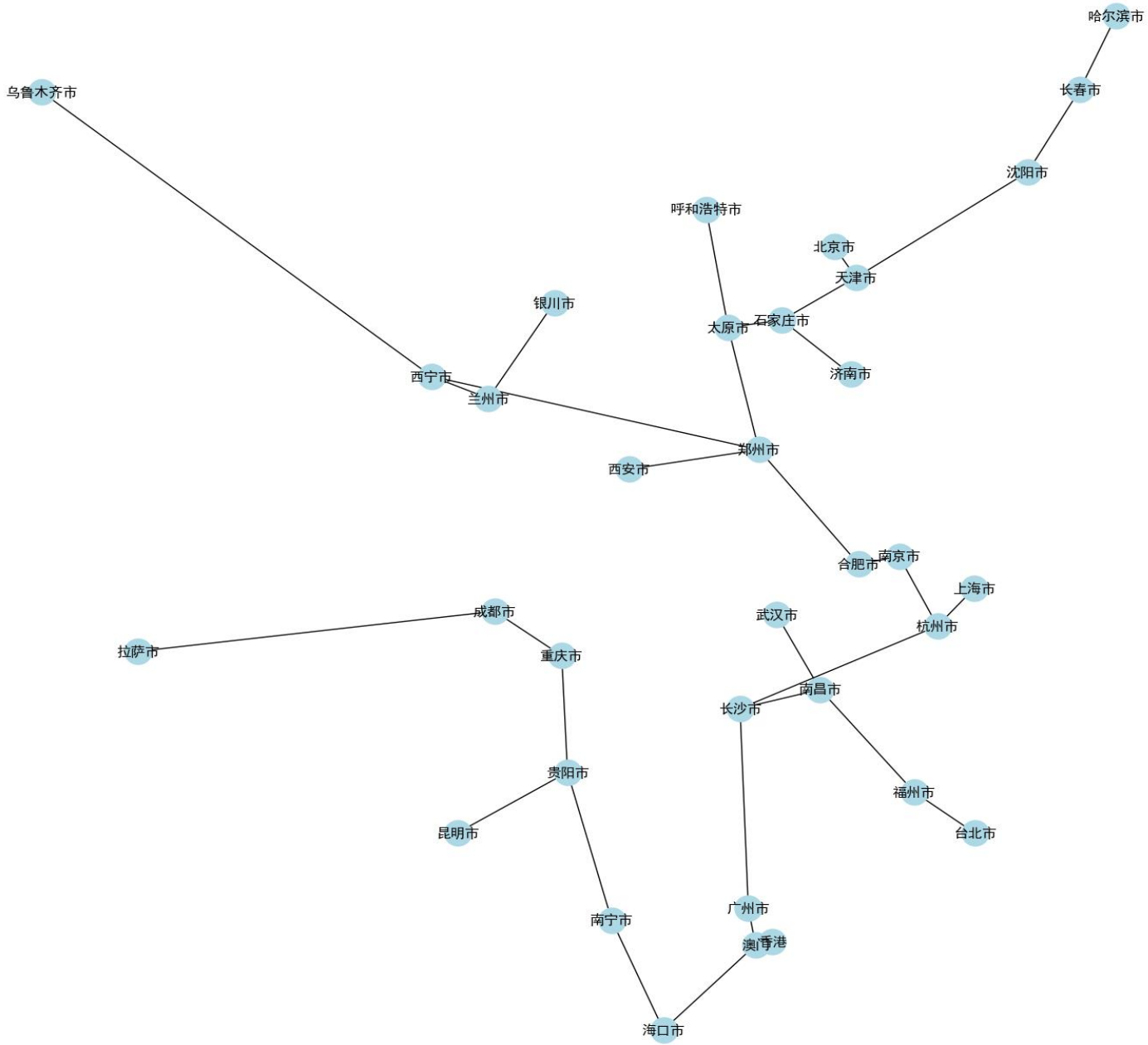


(3)要求不仅在西宁与郑州之间拉直达专线，还在杭州与长沙之间拉直达专线的情况下，使得整个电网长度最短。

这个问题就有点小小的陷阱了，本来我的想法是模仿(2),事先将西宁与郑州，杭州与长沙全部加入到树种，再更新相应的相邻结点。但是这样实际是不对的，如果只是连通这两对，它们四个并没有联通（不在一棵生成树中），这样强行地在(2)基础之上加入杭州或长沙并不一定保证下一个距离连通部分最短的点加入进来，无法保证加入的是安全边，最小生成树的性质被破坏了。

然后我的想法是这样的，在(2)的基础上，不断拓展延申，直到碰到了杭州/长沙，然后再将杭州与长沙之间那条边强制性地加入进来，之后再进行类似的更新即可。

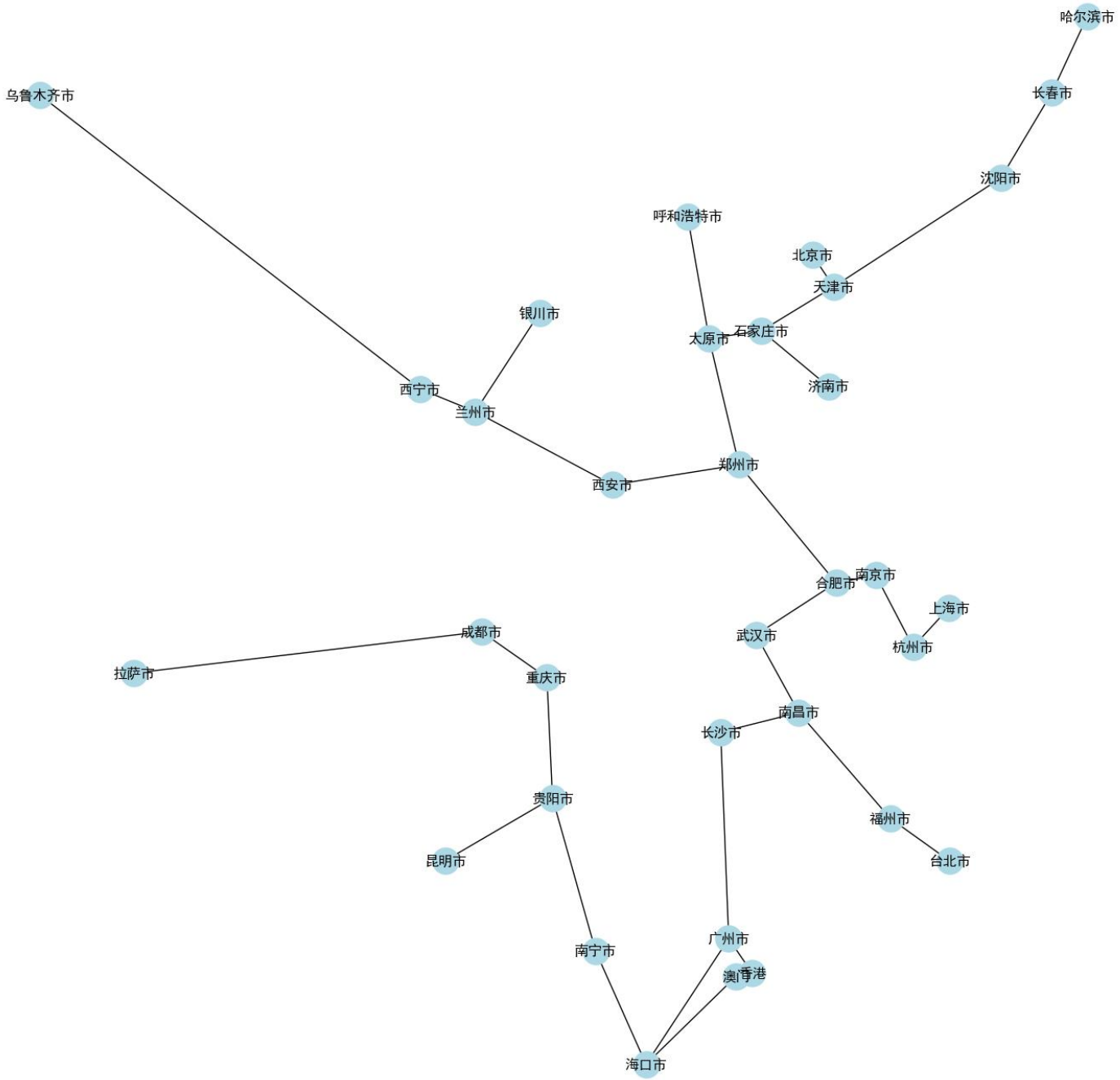
运行结果如下所示:……SUM LEN: 13372.3km，接下来我以可视化的方式展示结果：



(4)在香港与澳门、澳门与广州不拉直达线路的前提之下，使得整个电网的长度最短。

结合(1),我的思路是这样的:在每一次更新操作时，特判香港/澳门/广州，对于可能出现的不能拉直达专线的情况，停止更新最小边。这样一来，由于这两条路线一直不能被更新，会停留在很大很大的初始情况，自然就不会被选择了。

运行结果如下所示:……SUM LEN: 12797.6km，接下来我以可视化的方式展示结果：



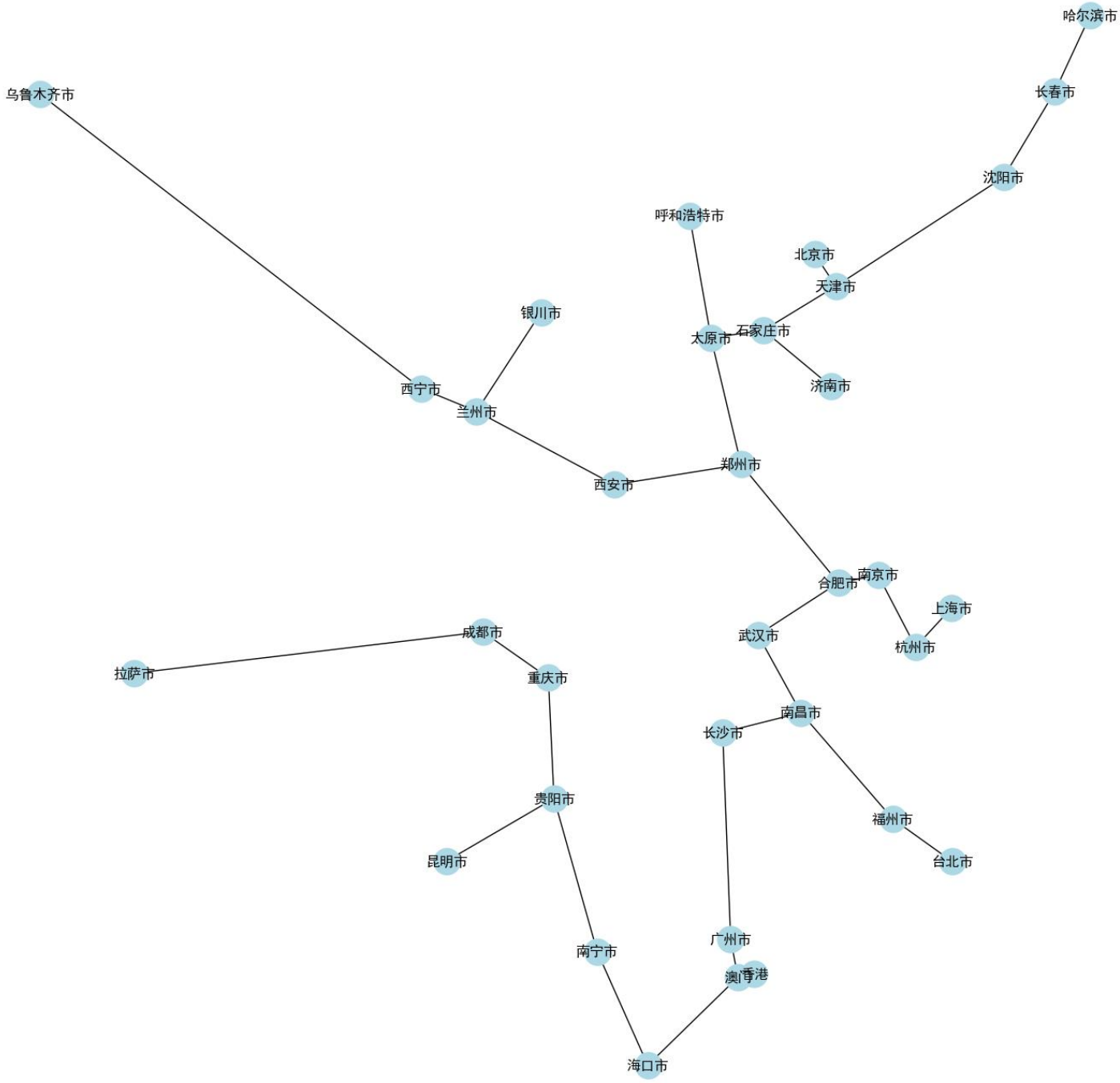
(5)山东、河南、山西、甘肃、青海、新疆以及比他们更北的省份称为北方省份，其余省份称为南方省份。如果在南方省份和北方省份之间仅规划一条直通专线，如何使得整个电网的长度最短。

这个在作业快截止时候加了一个很离谱的要求，就是以黄河为界划分南方和北方，然后这样的话西安都是南方城市了，有一些城市既有属于黄河以南的部分，又有属于黄河以北的部分。为了明确化问题，我划分的北方城市是这些:

```
string north_province[15] = {"济南市", "石家庄市", "天津市", "北京市", "沈阳市",
                             "长春市", "哈尔滨市", "郑州市", "太原市", "西安市",
                             "呼和浩特市", "兰州市", "银川市", "西宁市", "乌鲁木齐市"};
```

我的思路是将南方和北方的城市看作两张图分别进行最小生成树求解，然后最后再找一条南方到北方最短的边联通两张图即可。

运行结果如下所示:……SUM LEN: 12369.5km，接下来我以可视化的方式展示结果：



五、总结

对上机实践结果进行分析，问题回答，上机的心得体会及改进意见。

总而言之，本次实验花费了我大量的时间精力但并没有真正地多学什么新的、有价值的知识，反而被无穷的 debug 给拖延了。如果真的要说收获，我对两种最短生成树算法的掌握程度更上一层楼，本来自己还想利用优先队列来优化 prim 算法，但是在碰到这次实验中的各种特判、回溯的问题时，面对一个不那么大的数据集，我还是选择了朴素的方法一步一步去更新，选择。

面对不同的情境，最短生成树算法的使用有着不同的策略。为了在应用中最大化算法的价值，我们需要不断地分析，结合实际实现相应的功能。比如从(2)到(3)时，如果想当然，算法最基本的性质都不满足了，自然得不到正确的解。