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：21级	上机实践成绩：
指导教师：金澈清	**姓名**：包亦晟	
上机实践名称：国家电网超高压网络规划	**学号**：10215501451	**上机实践日期**：2023.5.25
上机实践编号：No.12	**组号**：1-451	

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

task1

求最小生成树既可以使用Prim，也可以使用Kruskal。这里因为我一开始省会城市的数量数错了，导致一开始采用的Prim算法得不出正确答案，之后又写了Kruskal，结果得出的答案是一样的。最后纠正了省会城市的数量才得到了正确的结果。

Prim

Prim算法的基本思想很简单。它每次将离连通部分的最近的点和点对应的边加入连通部分，连通部分就会渐渐扩大，最后整个图都会连通起来，并且边长之和是最小的。

为了完成上述的过程，我们需要定义如下的变量：

```
double g[N][N];
double dist[N];
bool st[N];
int pre[N];
```

二维数组g用来存放图，dist[i]表示第 i 节点到连通部分的最短距离，st[i]为true表示第i节点已经连通，为false表示第i节点还没连通。根据任务要求，我们是需要记录下路径的，所以需要一个pre数组。pre[i]表示第i节点的前驱节点。

这里还要考虑初始化的问题。我们首先要将dist数组都初始化为无穷大。这里我习惯性地将使用memset函数将dist数组初始化为了0x3f3f3f3f。但我忽略了这里的dist是double数组，0x3f3f3f3f并非是一个大数。之后经过查证，我用最朴素的循环方式将dist的每个元素都赋值为了：

```
numeric_limits<double>::infinity()
```

另外不要忘记pre数组也要初始化为全-1。由于在代码实现中pre是全局数组，所以默认为全0.但这就带来了一个问题，你不知道pre数组中的0是好没有前驱节点，还是说前驱节点是0号节点。所以需要初始化为-1.当然，我们还有一种解决方法是让顶点编号从1开始，不过我在本次实验中采用了前一种方式。

Kruskal

Kruskal算法的基本思想如下：

- 将所有边按照权值升序排序
- 遍历所有边，如果该边与之前所选择的边**没有组成回路**，那么就选择该边；如果会组成回路，就舍去
- 重复以上的过程，直到筛选出n-1条边

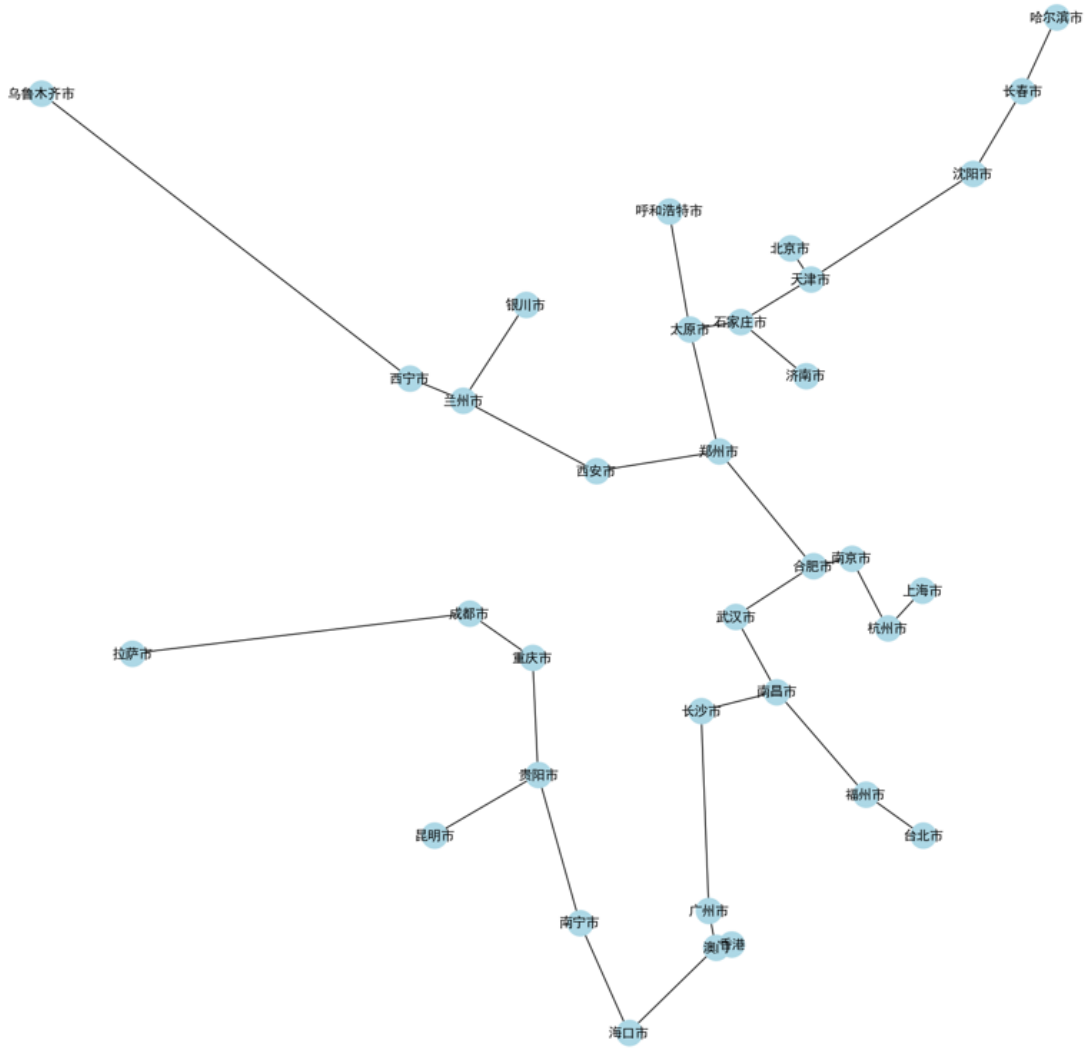
这里存在一个问题，如何判断是否会产生回路？这里要用到一个数据结构：并查集。基本步骤如下：

- 最初每个个顶点在不同的集合中
- 遍历过程的每条边，判断对应的两个顶点是否在一个集合中
- 如果边上的这两个顶点在一个集合中，说明两个顶点已经连通，不需要该边；如果不在一个集合中，则需要这条边

基本的代码实现限于篇幅也就不赘述了。

```
总长度为：12369.5
```

澳门 广州市 106.634
香港 澳门 64.0048
乌鲁木齐市 西宁市 1441.9
拉萨市 成都市 1249.67
西宁市 兰州市 194.278
兰州市 西安市 505.96
成都市 重庆市 265.982
贵阳市 南宁市 447.881
昆明市 贵阳市 435.472
重庆市 贵阳市 329.197
南宁市 海口市 365.228
海口市 澳门 421.967
台北市 福州市 250.622
广州市 长沙市 564.43
福州市 南昌市 444.143
杭州市 南京市 235.447
南昌市 武汉市 262.154
长沙市 南昌市 289.531
上海市 杭州市 164.04
合肥市 郑州市 465.513
南京市 合肥市 141.475
武汉市 合肥市 317.307
西安市 郑州市 435.687
郑州市 太原市 358.809
济南市 石家庄市 268.228
石家庄市 天津市 262.662
太原市 石家庄市 172.534
银川市 兰州市 343.112
呼和浩特市 太原市 338.861
天津市 沈阳市 605.429
北京市 天津市 109.744
哈尔滨市 长春市 232.473
长春市 沈阳市 279.076

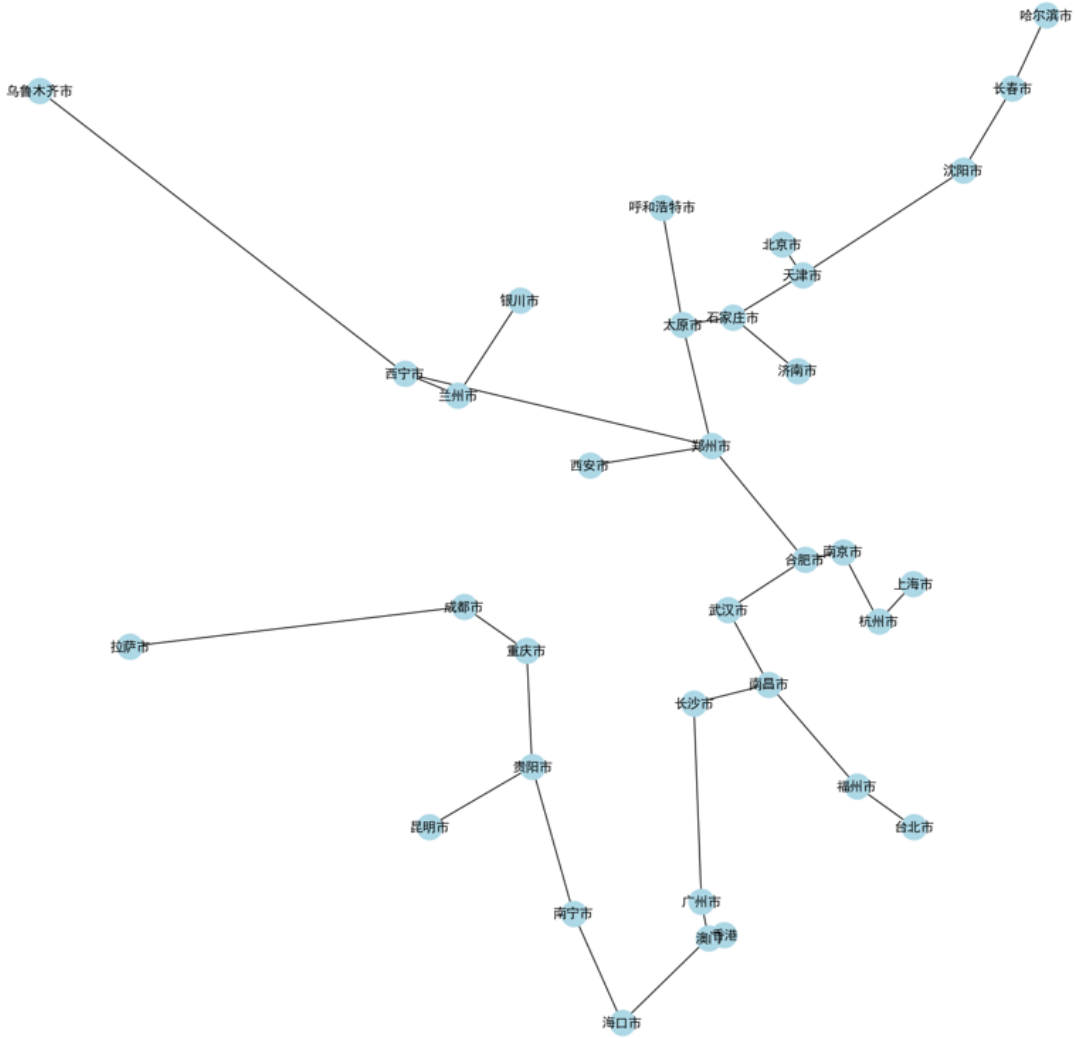


task2

我们只要在task1的基础上先将郑州和西宁加入树中，并且更新它们的相邻节点，最后从西宁/郑州出发即可。

```
总长度为：12956.1
澳门 广州市 106.634
香港 澳门 64.0048
乌鲁木齐市 西宁市 1441.9
拉萨市 成都市 1249.67
兰州市 西宁市 194.278
成都市 重庆市 265.982
贵阳市 南宁市 447.881
昆明市 贵阳市 435.472
重庆市 贵阳市 329.197
南宁市 海口市 365.228
海口市 澳门 421.967
台北市 福州市 250.622
广州市 长沙市 564.43
福州市 南昌市 444.143
杭州市 南京市 235.447
南昌市 武汉市 262.154
长沙市 南昌市 289.531
上海市 杭州市 164.04
合肥市 郑州市 465.513
南京市 合肥市 141.475
```

```
武汉市 合肥市 317.307
西安市 郑州市 435.687
郑州市 西宁市 1092.57
济南市 石家庄市 268.228
石家庄市 太原市 172.534
太原市 郑州市 358.809
银川市 兰州市 343.112
呼和浩特市 太原市 338.861
天津市 石家庄市 262.662
北京市 天津市 109.744
哈尔滨市 长春市 232.473
长春市 沈阳市 279.076
沈阳市 天津市 605.429
```

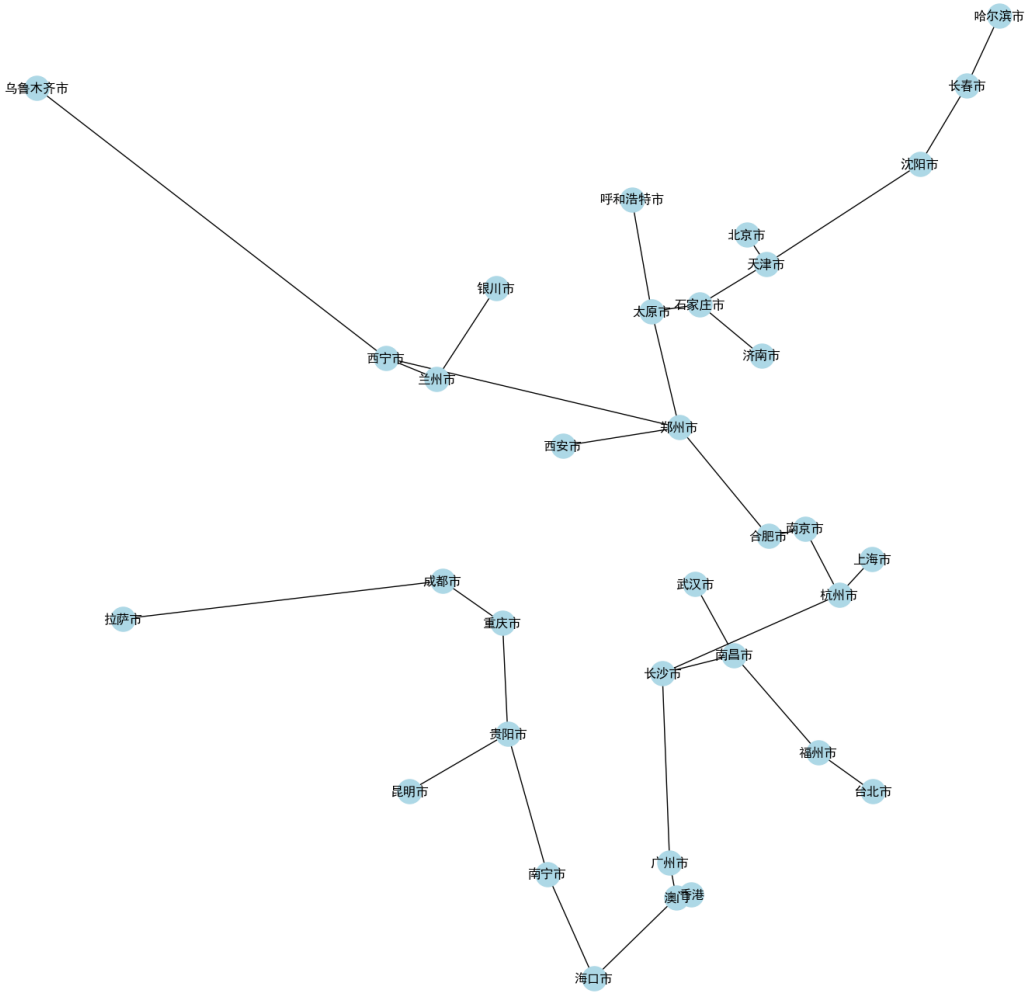


task3

这里我最开始是想要遵循着task2的思路，事先将郑州和西宁，杭州和长沙全部加入树中，并且更新相邻节点，再从这四个城市随机选择一个作为出发点。但是这种想法是**错误**的。因为当你处理完郑州和西宁的时候，并不代表着下一个距离连通部分最短的就是杭州或者是长沙了。所以，正确的做法是，顺着task2，当被选中的点是杭州/长沙的时候，再将边加入进来，并进行更新。

```
总长度为：13372.3
澳门 广州市 106.634
香港 澳门 64.0048
乌鲁木齐市 西宁市 1441.9
```

拉萨市 成都市 1249.67
兰州市 西宁市 194.278
成都市 重庆市 265.982
贵阳市 南宁市 447.881
昆明市 贵阳市 435.472
重庆市 贵阳市 329.197
南宁市 海口市 365.228
海口市 澳门 421.967
台北市 福州市 250.622
广州市 长沙市 564.43
福州市 南昌市 444.143
杭州市 南京市 235.447
南昌市 长沙市 289.531
长沙市 杭州市 733.531
上海市 杭州市 164.04
合肥市 郑州市 465.513
南京市 合肥市 141.475
武汉市 南昌市 262.154
西安市 郑州市 435.687
郑州市 西宁市 1092.57
济南市 石家庄市 268.228
石家庄市 太原市 172.534
太原市 郑州市 358.809
银川市 兰州市 343.112
呼和浩特市 太原市 338.861
天津市 石家庄市 262.662
北京市 天津市 109.744
哈尔滨市 长春市 232.473
长春市 沈阳市 279.076
沈阳市 天津市 605.429

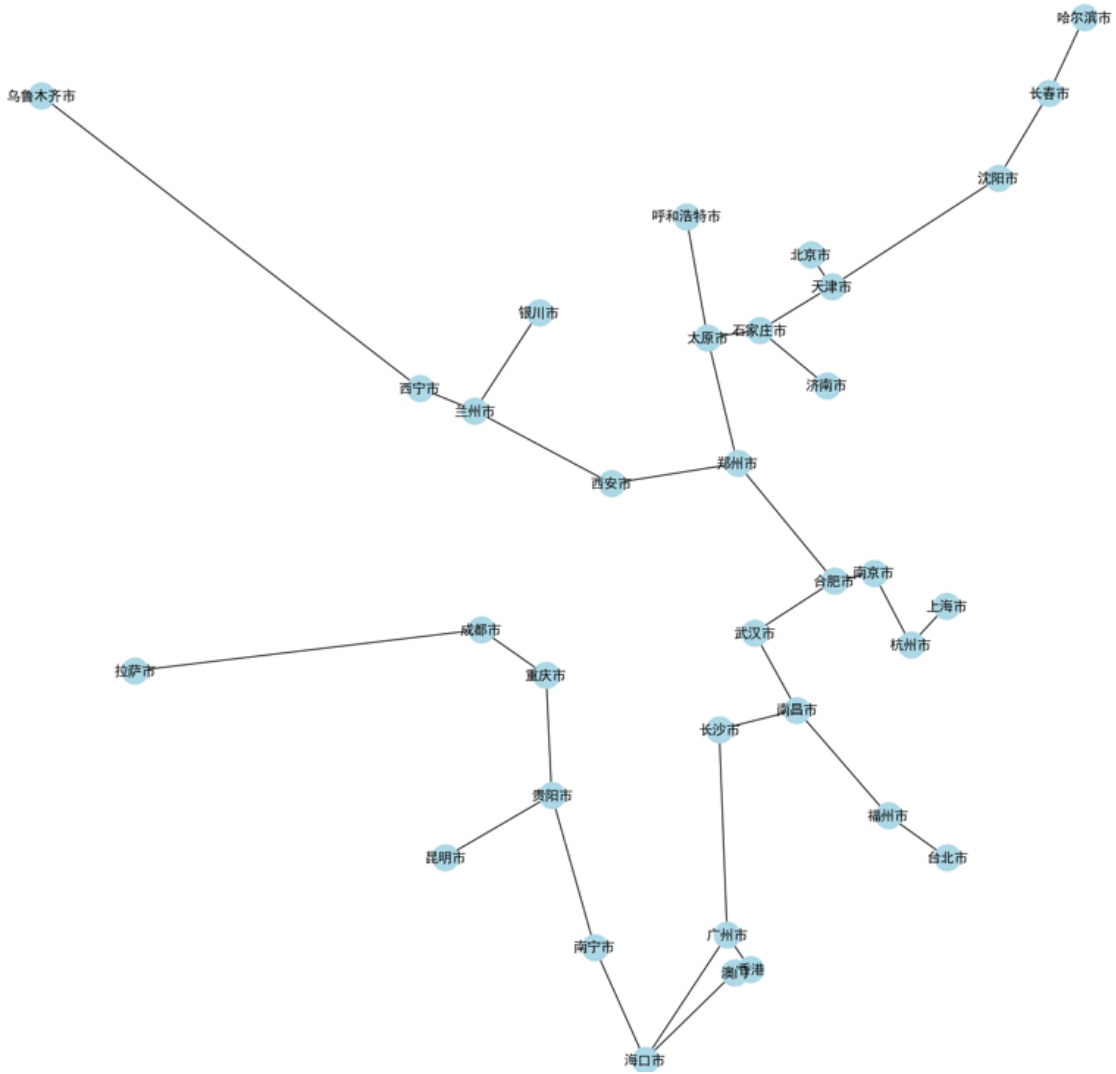


task4

这里只需要在task1的基础上，当扩展到香港/澳门/广州的时候，不要去更新其禁止连通节点的最小边即可，让其永远停留在无穷大，那么就不会被选择了。

```
总长度为：12797.6
澳门 海口市 421.967
香港 广州市 131.027
乌鲁木齐市 西宁市 1441.9
拉萨市 成都市 1249.67
西宁市 兰州市 194.278
兰州市 西安市 505.96
成都市 重庆市 265.982
贵阳市 南宁市 447.881
昆明市 贵阳市 435.472
重庆市 贵阳市 329.197
南宁市 海口市 365.228
海口市 广州市 467.756
台北市 福州市 250.622
广州市 长沙市 564.43
福州市 南昌市 444.143
杭州市 南京市 235.447
南昌市 武汉市 262.154
```

```
长沙市 南昌市 289.531
上海市 杭州市 164.04
合肥市 郑州市 465.513
南京市 合肥市 141.475
武汉市 合肥市 317.307
西安市 郑州市 435.687
郑州市 太原市 358.809
济南市 石家庄市 268.228
石家庄市 天津市 262.662
太原市 石家庄市 172.534
银川市 兰州市 343.112
呼和浩特市 太原市 338.861
天津市 沈阳市 605.429
北京市 天津市 109.744
哈尔滨市 长春市 232.473
长春市 沈阳市 279.076
```



task5

分别将南方的省会和北方的省会看作两张图。运用task1的方法生成两个电网图，最后找到南方省会到北方省会中权重最小的一条边加入即可。

由于结果与task1相同，为了节省篇幅，就不附上结果了。

五、总结

在本次实验中，因为很多细节方面没搞清楚耗费了我大量的时间，就比如说task1中的节点数量数错。希望以后自己能够在神智清醒并且事先完全学透了相应的算法之后再进行代码的撰写。

本次实验中的Prim算法仍然有优化的空间。如果我们可以加入优先队列的话，就不需要遍历所有的节点寻找距离集合最近的了。

在图像方面，我是根据实验所给经纬度绘制图像的，但是在最终效果的呈现上部分地区出现了重合，我并不知晓该如何解决。但是地区与地区之间的连接关系还是较为清晰的。

我发现Prim算法其实和Dijkstra算法是有着些许的相似性的。前者是更新到集合的距离，而后者是更新到起始点的距离。我们可以结合这一点对算法流程进行记忆。

本次实验中，我们是知道最小生成树是一定存在的。但其实并不是每一张图都会有最小生成树，这时候就需要我们加以判断。对于Prim算法来说，我们只需要看看最后的总长度是否是无穷大即可，如果是无穷大那么就不存在；对于Kruskal算法来说，我们需要设立一个cnt变量来记录边的数量，如果最后边的数量比顶点数量-1要小，那就说明不存在最小生成树。