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：**19 级	**上机实践成绩：**
指导教师：金澈清	**姓名：**龚敬洋	
上机实践名称：国家电网超高压网络规划	**学号：**10195501436	**上机实践日期：**2020/12/18
上机实践编号：No. 11	**组号：**1-436	

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

中国地图

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

(1) 整个电网的长度最短

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <cstring>
#define R 6371.004
#define SUP 100000005
using namespace std;
struct city{
    string name;
    double longitude;
    double latitude;
};
struct edge{
    int from;
    int to;
    double weight;
};
double rad(double angle){
    return angle * M_PI / 180;
}
//Only correct when the positions are in the north-eastern hemisphere
double earth_dist(city *pos_a, city *pos_b){
    double long_a = rad(pos_a->longitude);
    double lat_a = rad(pos_a->latitude);
    double long_b = rad(pos_b->longitude);
```

```
    double lat_b = rad(pos_b->latitude);
    double c = sin(lat_a) * sin(lat_b) + cos(lat_a) * cos(lat_b) * cos(long_b - long_a);
    return R * acos(c);
}
int main() {
    city c[35];
    edge chosen[35], low_cost[35];
    double graph[35][35] = {SUP};
    int visit[35] = {0}, flag = 0, n = 0, last_visit, cnt = 0;
    ifstream fin("position.txt");
    while (!fin.eof()){
        fin>>c[n].name>>c[n].longitude>>c[n].latitude;
        n++;
    }
    //Build the graph
    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
            if(i == j) graph[i][j] = 0;
            else {
                double dist = earth_dist(&c[i], &c[j]);
                graph[i][j] = dist;
                graph[j][i] = dist;
            }
    for(int i = 0; i < n; i++){
        low_cost[i] = {-1, -1, SUP};
    }
    visit[0] = 1;
    last_visit = 0;
    //Procedure of searching the MST
    while (!flag){
        flag = 1;
        //Update the lowest-cost edge set
        for(int i = 0; i < n; i++){
            if(visit[i] == 0 && graph[last_visit][i] < low_cost[i].weight){
                low_cost[i].from = last_visit;
                low_cost[i].to = i;
                low_cost[i].weight = graph[last_visit][i];
            }
        }
        int min_edge_idx = -1;
        double min_edge_dist = SUP;
        //Search for the smallest edge
        for (int i = 0; i < n; i++) {
            if(low_cost[i].weight != -1.0 && low_cost[i].weight < min_edge_dist){
                min_edge_dist = low_cost[i].weight;
                min_edge_idx = i;
            }
        }
        chosen[cnt] = low_cost[min_edge_idx];
        visit[min_edge_idx] = 1;
        low_cost[min_edge_idx] = {-1, -1, -1.0};
        last_visit = min_edge_idx;
        cnt++;
        for(int i = 0; i < n; i++){
            if(visit[i] == 0){
                flag = 0;
                break;
            }
        }
    }
    double sum = 0.0;
    for(int i = 0; i < cnt; i++){
        cout<<c[chosen[i].from].name<<"->"<<c[chosen[i].to].name<<" "<<chosen[i].weight<<"
km"<<endl;
        sum += chosen[i].weight;
    }
```

```
    cout<<"总长度: "<<sum<<"km"<<endl;
    return 0;
}
```

(2) 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（省略号代表与**(1)**对应位置完全相同）

```
...
int main() {
    ...
    int xi_ning_idx, zheng_zhou_idx;
    ...
    while (!fin.eof()){
        ...
        if (c[n].name == "西宁市") xi_ning_idx = n;
        else if (c[n].name == "郑州市") zheng_zhou_idx = n;
        ...
    }
    ...
    chosen[0].from = xi_ning_idx;
    chosen[0].to = zheng_zhou_idx;
    chosen[0].weight = graph[xi_ning_idx][zheng_zhou_idx];
    cnt++;
    visit[xi_ning_idx] = 1;
    visit[zheng_zhou_idx] = 1;
    last_visit = xi_ning_idx;
    for(int i = 0; i < n; i++){
        if(visit[i] == 0 && graph[last_visit][i] < low_cost[i].weight){
            low_cost[i].from = last_visit;
            low_cost[i].to = i;
            low_cost[i].weight = graph[last_visit][i];
        }
    }
    last_visit = zheng_zhou_idx;
    while (!flag){
        ...
    }
    ...
}
```

(3) 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（省略号代表与**(1)**对应位置完全相同）

```
...
int main() {
    ...
    int xi_ning_idx, zheng_zhou_idx, hang_zhou_idx, chang_sha_idx;
    ...
    while (!fin.eof()){
        ...
        if (c[n].name == "西宁市") xi_ning_idx = n;
        else if (c[n].name == "郑州市") zheng_zhou_idx = n;
        else if (c[n].name == "杭州市") hang_zhou_idx = n;
        else if (c[n].name == "长沙市") chang_sha_idx = n;
        ...
    }
    ...
```

```
    chosen[0].from = xi_ning_idx;
    chosen[0].to = zheng_zhou_idx;
    chosen[0].weight = graph[xi_ning_idx][zheng_zhou_idx];
    cnt ++;
    visit[xi_ning_idx] = 1;
    visit[zheng_zhou_idx] = 1;
    last_visit = xi_ning_idx;
    for(int i = 0; i < n; i++){
        if(visit[i] == 0 && graph[last_visit][i] < low_cost[i].weight){
            low_cost[i].from = last_visit;
            low_cost[i].to = i;
            low_cost[i].weight = graph[last_visit][i];
        }
    }
    last_visit = zheng_zhou_idx;
    while (!flag){
        ...
        if(last_visit == hang_zhou_idx && visit[chang_sha_idx] == 0){
            chosen[cnt].from = hang_zhou_idx;
            chosen[cnt].to = chang_sha_idx;
            chosen[cnt].weight = graph[hang_zhou_idx][chang_sha_idx];
            visit[chang_sha_idx] = 1;
            low_cost[chang_sha_idx] = {-1, -1, -1.0};
            for(int i = 0; i < n; i++){
                if(visit[i] == 0 && graph[last_visit][i] < low_cost[i].weight){
                    low_cost[i].from = last_visit;
                    low_cost[i].to = i;
                    low_cost[i].weight = graph[last_visit][i];
                }
            }
            cnt++;
            last_visit = chang_sha_idx;
        }
        else if(last_visit == chang_sha_idx && visit[hang_zhou_idx] == 0){
            chosen[cnt].from = chang_sha_idx;
            chosen[cnt].to = hang_zhou_idx;
            chosen[cnt].weight = graph[chang_sha_idx][hang_zhou_idx];
            visit[hang_zhou_idx] = 1;
            low_cost[hang_zhou_idx] = {-1, -1, -1.0};
            for(int i = 0; i < n; i++){
                if(visit[i] == 0 && graph[last_visit][i] < low_cost[i].weight){
                    low_cost[i].from = last_visit;
                    low_cost[i].to = i;
                    low_cost[i].weight = graph[last_visit][i];
                }
            }
            cnt++;
            last_visit = hang_zhou_idx;
        }
        ...
    }
    ...
}
```

(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（省略号代表与**(1)**对应位置完全相同）

```
...
int main() {
    ...
    int hong_kong_idx, macao_idx, guang_zhou_idx;
    ...
```

```
    while (!fin.eof()){
        ...
        if (c[n].name == "香港") hong_kong_idx = n;
        else if (c[n].name == "澳门") macao_idx = n;
        else if (c[n].name == "广州市") guang_zhou_idx = n;
        ...
    }
    ...
    while (!flag){
        ...
        for(int i = 0; i < n; i++){
            if(visit[i] == 0 && graph[last_visit][i] < low_cost[i].weight){
                bool spec_1, spec_2;
                spec_1 = (last_visit == hong_kong_idx && i == macao_idx) || (last_visit == macao_idx && i == hong_kong_idx);
                spec_2 = (last_visit == macao_idx && i == guang_zhou_idx) || (last_visit == guang_zhou_idx && i == macao_idx);
                if (!spec_1 && !spec_2) {
                    low_cost[i].from = last_visit;
                    low_cost[i].to = i;
                    low_cost[i].weight = graph[last_visit][i];
                }
            }
        }
        ...
    }
    ...
}
```

(5) 在南方省份和北方省份之间仅规划一条直通专线，使得整个电网的长度最短（省略号代表与**(1)**对应位置完全相同）

```
...
int main() {
    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", "呼和浩特市",
                                 "西安市", "兰州市", "银川市", "西宁市", "乌鲁木齐市"};
    map<int, int> region_map;
    ...
    int visit[35] = {0}, flag = 0, n = 0, last_visit, cnt = 0, north_tag;
    ...
    while (!fin.eof()){
        ...
        north_tag = 0;
        for (int i = 0; i < 15; i++){
            if (north_province[i] == c[n].name){
                region_map[n] = 0;
                north_tag = 1;
                break;
            }
        }
        if (north_tag == 0) region_map[n] = 1;
        ...
    }
    ...
    //Connect northern city
    while (!flag){
        ...
        for(int i = 0; i < n; i++){
            if(region_map[i] == 0 && visit[i] == 0 && graph[last_visit][i] < low_cost[i].weight){
```

```
                low_cost[i].from = last_visit;
                low_cost[i].to = i;
                low_cost[i].weight = graph[last_visit][i];
            }
        }
        ...
    }
    flag = 0;
    for(int i = 0; i < n; i++){
        low_cost[i] = {-1, -1, SUP};
    }
    visit[12] = 1;
    last_visit = 12;
    //Connect southern city
    while (!flag){
        ...
        for(int i = 0; i < n; i++){
            if(region_map[i] == 1 && visit[i] == 0 && graph[last_visit][i] < low_cost[i].weight){
                low_cost[i].from = last_visit;
                low_cost[i].to = i;
                low_cost[i].weight = graph[last_visit][i];
            }
        }
        ...
    }
    int min_from, min_to;
    double min_dist = SUP;
    for (int i = 0; i < n; i++)
        for(int j = 0; j < i; j++){
            int spec = (region_map[i] == 0 && region_map[j] == 1) || (region_map[i] == 1 && region_map[j] == 0);
            if (spec && graph[i][j] < min_dist){
                min_from = i;
                min_to = j;
                min_dist = graph[i][j];
            }
        }
    chosen[cnt].from = min_from;
    chosen[cnt].to = min_to;
    chosen[cnt].weight = min_dist;
    cnt++;
    ...
}
```

2. 写出算法的思路。

(1) 整个电网的长度最短

利用 Prim 算法，每次选择已连通节点集到未连通相邻节点集中最小的边连通，并更新相邻节点集及对应的最小边，最终即可得到长度最短的电网图。

(2) 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

先将西宁到郑州的边加入电网图，并更新西宁和郑州的相邻节点，再以西宁或郑州为起点执行(1)的步骤。

(3) 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

在(2)的基础上，当电网图扩展到杭州或长沙时，将杭州到长沙的边加入电网图，并更新杭州和长沙的相邻节点，再以杭州或长沙为起点继续扩展电网图。

(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

在(1)的基础上，当电网图扩展到香港、澳门或广州时，不更新其禁连通节点的最小边，即可避免算法选择非法边。

(5) 在南方省份和北方省份之间仅规划一条直通专线，使得整个电网的长度最短

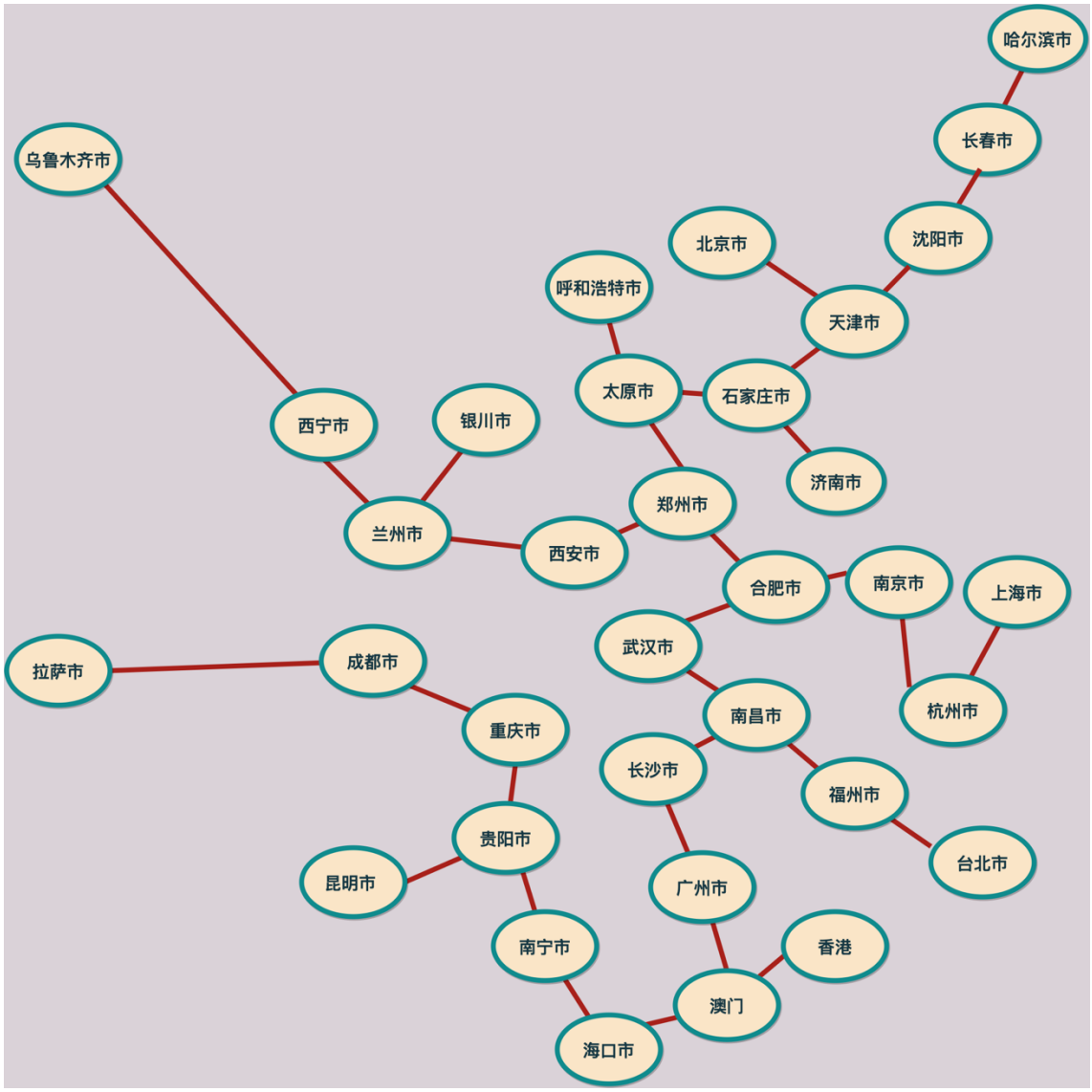
将南方省份和北方省份分别看作两张图，并使用(1)的方法分别生成两张电网图，最后找出南方省份中到北方省份最近的一条边加入电网图，即可得到一张全连通的电网图。由于初始位置对 Prim 算法没有影响，故可以保证得到的电网长度最短。

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。

(1) 整个电网的长度最短

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

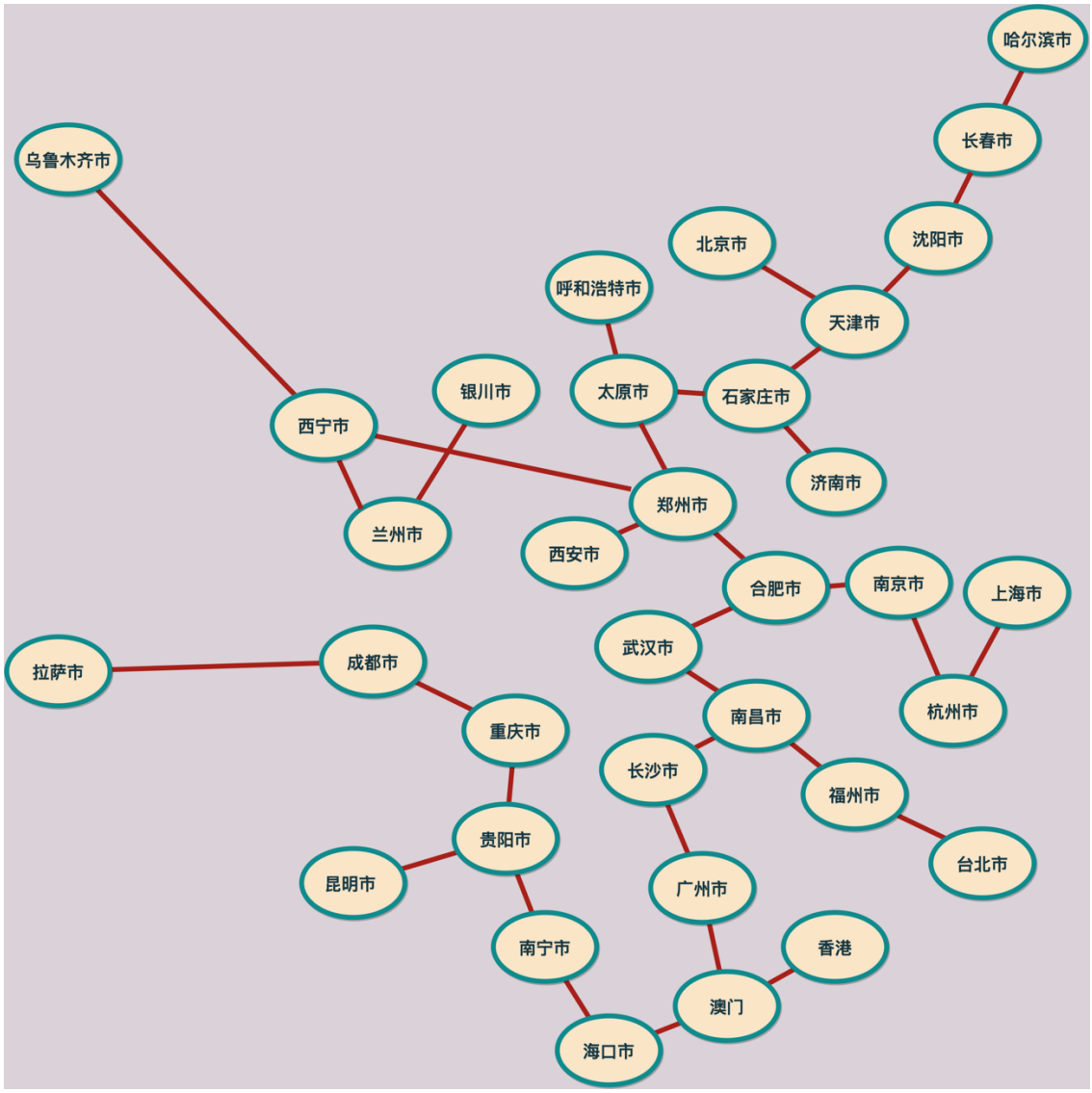
总长度：12369.5km



(2) 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

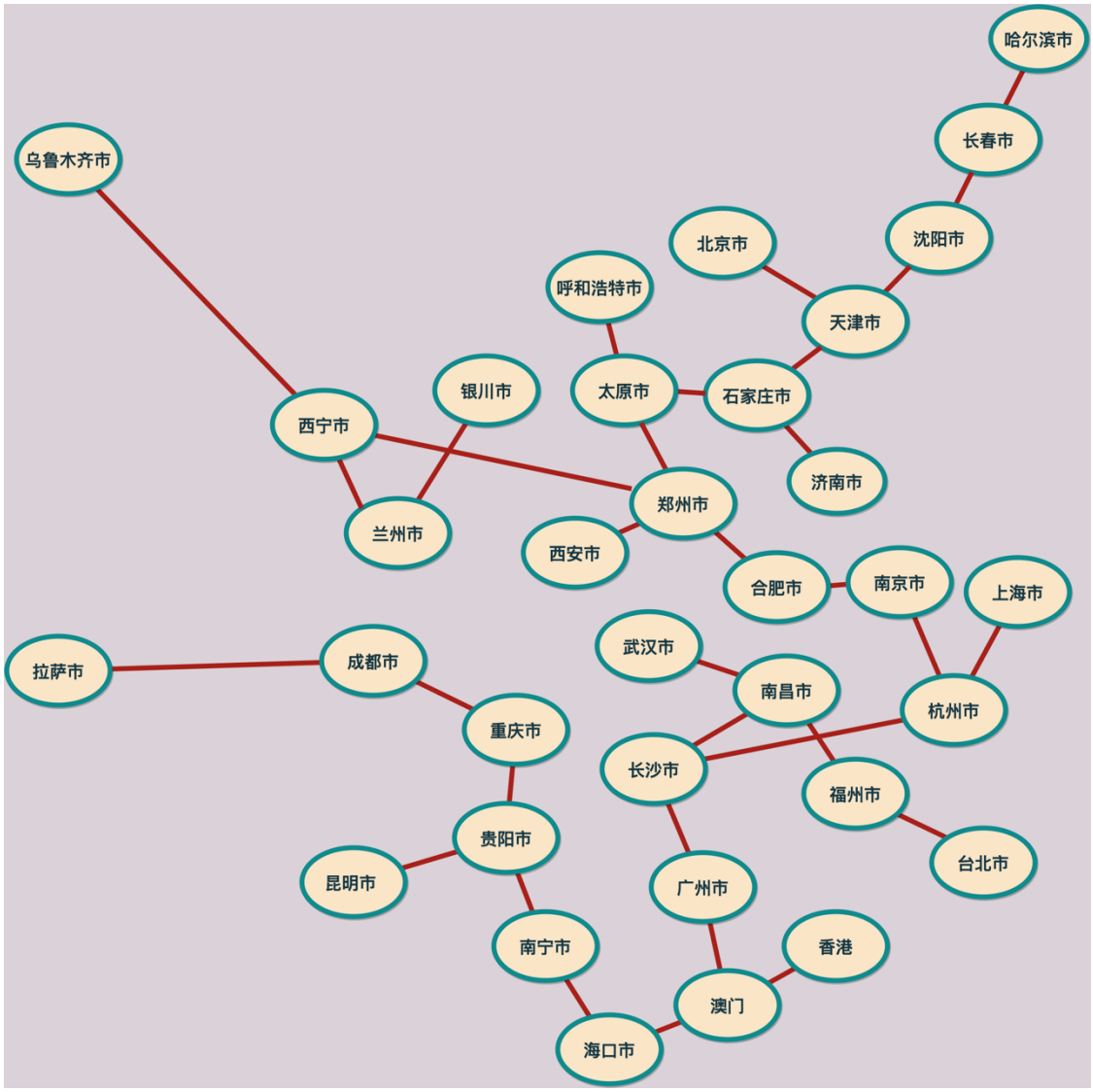
西宁市->郑州市 1092.57km
西宁市->兰州市 194.278km
兰州市->银川市 343.112km
郑州市->太原市 358.809km
太原市->石家庄市 172.534km
石家庄市->天津市 262.662km
天津市->北京市 109.744km
石家庄市->济南市 268.228km
太原市->呼和浩特市 338.861km
郑州市->西安市 435.687km
郑州市->合肥市 465.513km
合肥市->南京市 141.475km
南京市->杭州市 235.447km
杭州市->上海市 164.04km
合肥市->武汉市 317.307km
武汉市->南昌市 262.154km
南昌市->长沙市 289.531km

南昌市->福州市 444.143km
福州市->台北市 250.622km
长沙市->广州市 564.43km
广州市->澳门 106.634km
澳门->香港 64.0048km
澳门->海口市 421.967km
海口市->南宁市 365.228km
南宁市->贵阳市 447.881km
贵阳市->重庆市 329.197km
重庆市->成都市 265.982km
贵阳市->昆明市 435.472km
天津市->沈阳市 605.429km
沈阳市->长春市 279.076km
长春市->哈尔滨市 232.473km
成都市->拉萨市 1249.67km
西宁市->乌鲁木齐市 1441.9km
总长度：12956.1km



(3) 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

西宁市->郑州市 1092.57km
西宁市->兰州市 194.278km
兰州市->银川市 343.112km
郑州市->太原市 358.809km
太原市->石家庄市 172.534km
石家庄市->天津市 262.662km
天津市->北京市 109.744km
石家庄市->济南市 268.228km
太原市->呼和浩特市 338.861km
郑州市->西安市 435.687km
郑州市->合肥市 465.513km
合肥市->南京市 141.475km
南京市->杭州市 235.447km
杭州市->长沙市 733.531km
杭州市->上海市 164.04km
长沙市->南昌市 289.531km
南昌市->武汉市 262.154km
南昌市->福州市 444.143km
福州市->台北市 250.622km
长沙市->广州市 564.43km
广州市->澳门 106.634km
澳门->香港 64.0048km
澳门->海口市 421.967km
海口市->南宁市 365.228km
南宁市->贵阳市 447.881km
贵阳市->重庆市 329.197km
重庆市->成都市 265.982km
贵阳市->昆明市 435.472km
天津市->沈阳市 605.429km
沈阳市->长春市 279.076km
长春市->哈尔滨市 232.473km
成都市->拉萨市 1249.67km
西宁市->乌鲁木齐市 1441.9km
总长度：13372.3km

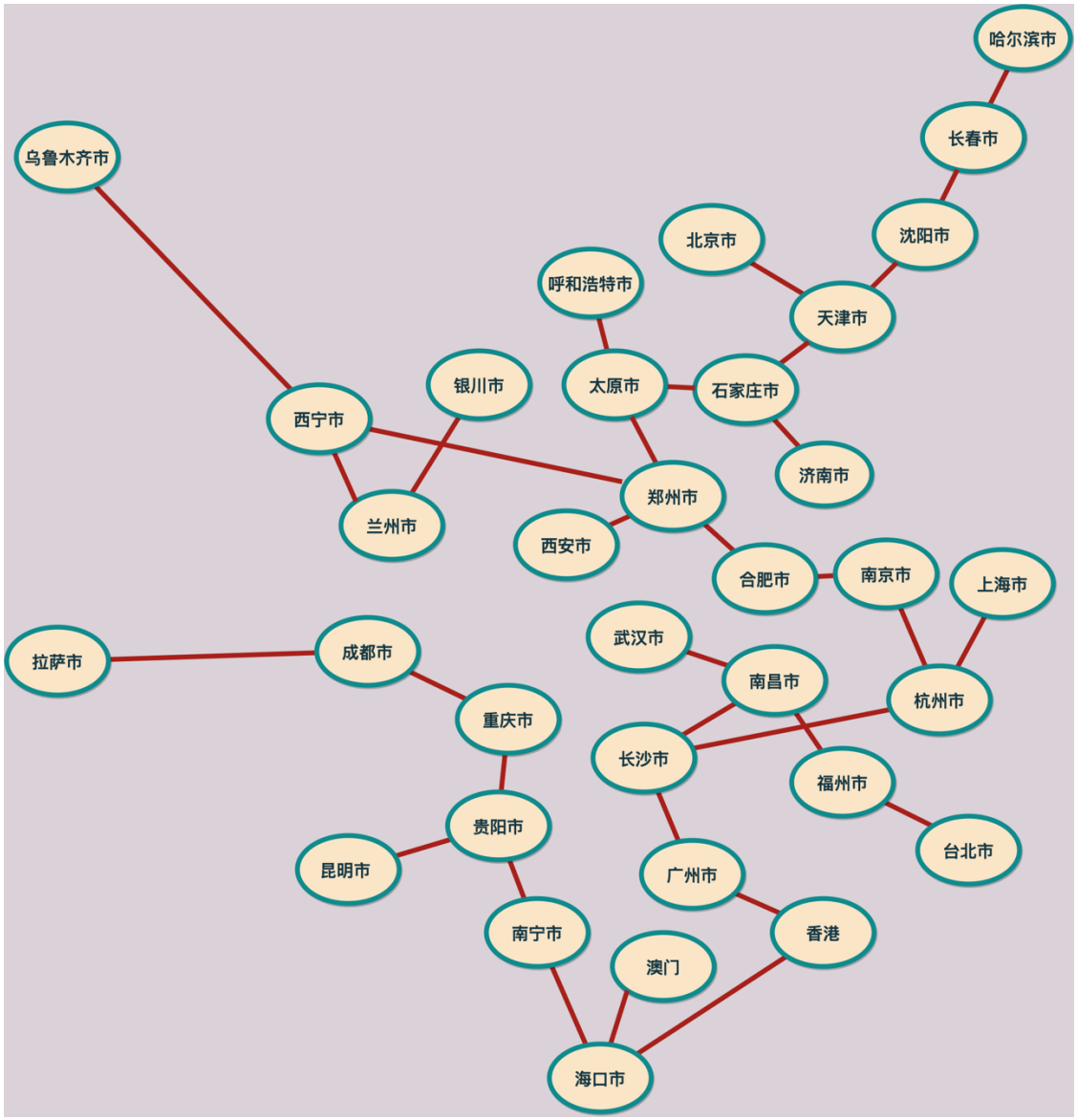


(4) 在香港与澳门、澳门与广州不拉直达线路的前提之下，使得整个电网的长度最短

沈阳市->长春市 279.076km
长春市->哈尔滨市 232.473km
沈阳市->天津市 605.429km
天津市->北京市 109.744km
天津市->石家庄市 262.662km
石家庄市->太原市 172.534km
石家庄市->济南市 268.228km
太原市->呼和浩特市 338.861km
太原市->郑州市 358.809km
郑州市->西安市 435.687km
郑州市->合肥市 465.513km
合肥市->南京市 141.475km
南京市->杭州市 235.447km
杭州市->上海市 164.04km
合肥市->武汉市 317.307km
武汉市->南昌市 262.154km
南昌市->长沙市 289.531km

南昌市->福州市 444.143km
福州市->台北市 250.622km
西安市->兰州市 505.96km
兰州市->西宁市 194.278km
兰州市->银川市 343.112km
长沙市->广州市 564.43km
广州市->香港 131.027km
广州市->海口市 467.756km
海口市->南宁市 365.228km
海口市->澳门 421.967km
南宁市->贵阳市 447.881km
贵阳市->重庆市 329.197km
重庆市->成都市 265.982km
贵阳市->昆明市 435.472km
成都市->拉萨市 1249.67km
西宁市->乌鲁木齐市 1441.9km

总长度：12797.6km



(5) 在南方省份和北方省份之间仅规划一条直通专线，使得整个电网的长度最短

与(1)结果相同

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。

电网规划问题本质上就是求最小生成树。由于两两城市均可连通，故为一张稠密图，因此使用 Prim 算法更加合适。Prim 算法可在$O(n^2)$的时间内求出一张对应的最小生成树，而实验中对特殊情况处理的时间复杂度均未超过$O(n^2)$，因此总时间代价仍为$O(n^2)$。