谭徽宇的日志

序言——（本次实验的目的）

我们通过对省的每个点进行连接形成的 `全连接图` 和由铁路经停站生成的 `铁路网图` 来生成不同的最小生成树，通过图像对比它们的结构研究各省城市间的铁路布局与城市地理位置的关系。

总共要完成的内容

1. 数据收集
2. 经纬度到距离的换算
3. 经纬度与特定分辨率区域的换算
4. kruskal 算法实现
5. 以距离为标准抽象各省的最小生成树
6. 对比当前铁路网抽象出的最小生成树与理论最小生成树
7. 分析对比结果,研究差异原因
8. 其他分析

##

2021年12月14日——（导入城市信息与坐标）

我今天的任务是将 JSON 文件中的内容用 C 语言以链表形式按结构存储

> **记录**
>
> 我国有34个省级行政区

国家标准 GB/T 2260 下的地区 `JSON` 文件结构

省结构

```
[
    {
    "code": "420000",
    "name": "湖北省",
    "province": "42",
    "children": []/*直辖单位*/
    },
]
```

市结构

```
"children": [
      {
        "code": "420100",
        "name": "武汉市",
        "province": "42",
        "city": "01",
        "children": []/*市级区*/
      },
]
```

市级区结构

```
"children": [
          {
            "code": "420102",
            "name": "江岸区",
            "province": "42",
            "city": "01",
            "area": "02"
          },
]
```

分析

从数据的本身结构由干到支来分析，首先需要一个集合 `Country` 包含所有区域，这个集合将作为传入数据和传出数据的通道出入口的总支。

然后到 `Provinces` 层，该层由所有**省结点**组成，总大小是 **34** ，因为它的数据总量较少且需要在运行过程中起到分流的作用，所以我认为采用**数组**存储该层更合适。

下一个是 `province` 结点，一个这样的省结点存储了**它自己的信息**以及**地级市**。

它自己的信息包括：

1. 编码
2. 名字
3. 地级市

```
#include "global.h"

#define Provinces_NUMBER 34;
typedef char* String;

Province ProvincesTier[35];// one sentry with thirty_five "Province"

// replace by "ProvincesTier"
//typedef struct Countrys {
//
//}Country;

typedef struct Provinces {
```

```
    String code;
    String name;
    String province;
    City* childrens;    //citys
    int cityNum;
}Province;

typedef struct Citys {
    float x, y;         // longitude and latitude
    String code;
    String name;
    Province* province; // link with its own province
    Block* childrens;   //blocks
    int blockNum;
}City;

typedef struct Blocks {
    String code;
    String name;
    Province* province; // link with its own province
    City* chileres;     // link with its own city
}Block;
```

2021年12月15日--(简化铁路信息数据集)

当前铁路信息数据集中不需要的数据过多，今天的目的是将其简化为以下内容：

```
[{
    "stationA":"name",
    "stationB":"name",
    "train_code":"code"
}]
```

逻辑如下：

```
{"station_train_code":"D27(天津西-哈尔滨西)","train_no":"2500000D2700"}
```

原有信息中，我需要的信息结构是：`"code(name-name)"`，而原数据中大量存在`""`用于包裹属性、值。
并用`{}`来分离每一组同类数据。那么，在我当前读取的`"str"`中如果不包含`-`这个连接符，那么我就不需要这个值，则清空记录来缓存下一组。

```
void store_train_code_name(train* collection,char*record,FILE*save){
    if (record == NULL) {
        return;
    }
    char* delim01 = "e";
    char* delim02 = "\"";
```

```
    char* delim03 = "(";
    char* delim04 = "-";
    char* code;
    char* nameA,*nameB;
    code = strtok(record, delim01);
    code = strtok(NULL, delim01);
    code = strtok(code, delim02);
    code = strtok(NULL, delim02);
    code = strtok(code, delim03);
    nameA = strtok(NULL, delim03);
    nameA = strtok(nameA, delim04);
    nameB = strtok(NULL, delim04);

    if (code ≠ NULL && nameA ≠ NULL && nameB ≠ NULL) {
        fprintf(save,"%s,%s,%s\n", code, nameA, nameB);
    }

    return;
}

void test() {
    FILE* fp;
    int c;

    fp = fopen("train.txt", "r");
    while (1)
    {
        c = fgetc(fp);
        if (feof(fp))
        {
            break;
        }
        printf("%c", c);
    }
    fclose(fp);
}

void train_conver() {
    FILE* file = fopen("train.txt","w+");
    String path = "E:\\CourseDesign\\database\\train.txt";
    char* line, * record;
    char* recordA[32] = { NULL,NULL };
    char buffer[1024];
    FILE* fp = NULL;
    int flag;   //
    if ((fp = fopen(path, "a+")) == NULL)return;
    train* collection = (train*)calloc(1, sizeof(collection));
    int i = 0; int j = 0;

```

```
    while ((line = fgets(buffer, sizeof(buffer), fp)) != NULL &&
i<20)
    {
        record = strtok(line, ")");
        while (record != NULL)
        {
            recordA[j] = record;
            record = strtok(NULL, ")");
            j++;
        }
        /*recordA[j - 1] = NULL;*/
        int length = j; j = 0;
        while (j<length) {
            store_train_code_name(collection, recordA[j],file);
            recordA[j] = NULL;
            j++;
        }
        j = 0;
        i++;
    }
    fclose(file);
    fclose(fp);
    printf("%d,",rename("train.txt", "train.csv"));
}
```

（用 C 语言生成简化文件）

—谭徽宇

2021年12月16日--（完善铁路信息数据集文件）

今天的任务是完善铁路数据集文件，使得我们可以根据该文件建立建立以下关系：

1. 站点-城市

2. 站点-站点
3. 城市-纬度-经度

目的是利用程序将城市信息与经纬度信息统合

> 2349 车次， 3182 个地区，总计 31588 条经停站数据

这些数据将用于第一步，无权无向图的构建。

第二步，通过城市经纬度数据与当前内容的联系，为 “站台” 与城市构建联系，从而达到对全国所有街道构成直接关系

第三步，根据数据中的铁路信息和载入的经纬度，构建有权无向图

第四步，根据有权无向图，借助

2021年12月17日——（kruskal 算法逻辑实现（C语言））

```
4 - 0 : 1
4 - 1 : 1
3 - 2 : 2
5 - 2 : 3
3 - 1 : 4
Spanning tree cost: 11
```

```c
#include "global.h"

#define MAX 30

typedef struct edge {
    int u, v, w;
} edge;
```

```
8
9  typedef struct edge_list {
10     edge data[MAX];
11     int n;
12 } edge_list;
13
14 edge_list elist;
15
16 int Graph[MAX][MAX], n;
17 edge_list spanlist;
18
19 void kruskalAlgo();
20 int find(int belongs[], int vertexno);
21 void applyUnion(int belongs[], int c1, int c2);
22 void sort();
23 void print();
24
25 void kruskalAlgo() {
26     int belongs[MAX], i, j, cno1, cno2;
27     elist.n = 0;
28
29     for (i = 1; i < n; i++)
30         for (j = 0; j < i; j++) {
31             if (Graph[i][j] != 0) {
32                 elist.data[elist.n].u = i;
33                 elist.data[elist.n].v = j;
34                 elist.data[elist.n].w = Graph[i][j];
35                 elist.n++;
36             }
37         }
38
39     sort();
40
41     for (i = 0; i < n; i++)
42         belongs[i] = i;
43
44     spanlist.n = 0;
45
46     for (i = 0; i < elist.n; i++) {
47         cno1 = find(belongs, elist.data[i].u);
48         cno2 = find(belongs, elist.data[i].v);
49
50         if (cno1 != cno2) {
51             spanlist.data[spanlist.n] = elist.data[i];
52             spanlist.n = spanlist.n + 1;
53             applyUnion(belongs, cno1, cno2);
54         }
55     }
56 }
```

```
57
58  int find(int belongs[], int vertexno) {
59      return (belongs[vertexno]);
60  }
61
62  void applyUnion(int belongs[], int c1, int c2) {
63      int i;
64
65      for (i = 0; i < n; i++)
66          if (belongs[i] == c2)
67              belongs[i] = c1;
68  }
69
70  void sort() {
71      int i, j;
72      edge temp;
73
74      for (i = 1; i < elist.n; i++)
75          for (j = 0; j < elist.n - 1; j++)
76              if (elist.data[j].w > elist.data[j + 1].w) {
77                  temp = elist.data[j];
78                  elist.data[j] = elist.data[j + 1];
79                  elist.data[j + 1] = temp;
80              }
81  }
82
83  void print() {
84      int i, cost = 0;
85
86      for (i = 0; i < spanlist.n; i++) {
87          printf("\n%d - %d : %d", spanlist.data[i].u, spanlist.data[i].v,
    spanlist.data[i].w);
88          cost = cost + spanlist.data[i].w;
89      }
90
91      printf("\nSpanning tree cost: %d", cost);
92  }
93
94  int randSeed = 0;
95  int MyRandom(int i) {
96      unsigned int times = (unsigned int)time(NULL);
97      srand(times * (i + 1));
98      randSeed = rand();
99      randSeed = randSeed % 100;
100     return randSeed;
101 }
102
103 int kruskaltest() {
104     int i, j, total_cost;
```

```
105
106     n = 6;
107     for (int i=0; i < n; i++) {
108         for (int j=0; j < n; j++) {
109             Graph[i][j] = MyRandom(j+i*j+MyRandom(i))%10;
110         }
111     }
112     kruskalAlgo();
113     print();
114 }
```

2021年12月19日——（将表导入数据库并完成相应语法）

任务一、利用 SQL 语法完成构图所需数据的返回

目标的 csv 格式表示：`唯一id,对应城市id，名字，pingyin，经度，维度`

```
1  select res2.id as id,res1.pos_id as pos_id,res2.pid as pid,res2.deep as
   deep,res2.`name` as name from(select
   res.id,res.pos_id,res.pid,res.deep,res.`name`,res.ext_id from (SELECT
   id,pos_id,pid,name,ext_id,deep from detaillevel de GROUP BY
   de.ext_name,de.ext_id) res HAVING res.deep=1)res1,(select
   res.id,res.pos_id,res.pid,res.deep,res.`name`,res.ext_id from (SELECT
   id,pos_id,pid,name,ext_id,deep from detaillevel de GROUP BY
   de.ext_name,de.ext_id) res HAVING res.deep>=2)res2 where res1.pos_id =
   LEFT(res2.pid,4);
```

建立临时表 `tar`

对应城市的关系建立：

```
1  select * from aotude ao where ao.ID
2  = (select tar.pos_id from tar where tar.name = '一元街');
```

任务二、利用 JDBC 将数据导入 Java 程序并输出

目标如下：

通过省 id 对一个省进行选择，将该省所有城市及其经纬度传入程序

```
1  select res.id,res.`name`,ao.longitude,ao.latitude from aotude ao,
   (select LEFT(de.id,4) as id,de.pid,de.name from detaillevel de where
   deep<2)res where res.id = ao.ID and LEFT(res.id,2) = 42;
```

```
4201    武汉 114.30525   30.59276
4202    黄石 115.0389    30.19953
4203    十堰 110.79801   32.62918
4205    宜昌 111.28642   30.69186
4206    襄阳 112.12255   32.009
4207    鄂州 114.89495   30.39085
4208    荆门 112.19945   31.03546
4209    孝感 113.91645   30.92483
4210    荆州 112.24069   30.33479
4211    黄冈 114.87238   30.45347
4212    咸宁 114.32245   29.84126
4213    随州 113.38262   31.69013
4228    恩施 109.48817   30.27217
```

```
public void Loadingposition(String id){
    //1 读取配置文件中4个基本信息
    InputStream is =

 JDBCUtil.class.getClassLoader().getResourceAsStream("jdbc.properties
");
    Properties pros = new Properties();
    try {
        pros.load(is);
    } catch (IOException e) { }
    String user=pros.getProperty("user");
    String password=pros.getProperty("password");
    String url=pros.getProperty("url");
    String driverClass=pros.getProperty("driverClass");
    //2 加载驱动
    try {
        Class.forName(driverClass);
    } catch (ClassNotFoundException e) { }
    //3 获取链接
    Connection conn = null;
    try {
        conn = DriverManager.getConnection(url, user, password);
        Statement statement = conn.createStatement();
        String sql = "select
res.id,res.`name`,ao.longitude,ao.latitude from aotude ao,(select
LEFT(de.id,4) as id,de.pid,de.name from detaillevel de where
deep<2)res where res.id = ao.ID and LEFT(res.id,2) = "+id+";";
        ResultSet rs = statement.executeQuery(sql);
        while(rs.next()){
                this.positions.add(new
Positions(rs.getString(1),rs.getString(2),Double.parseDouble(rs.getSt
ring(3)),Double.parseDouble(rs.getString(4))));
        }
    } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
    }
```

```
'十堰',4203,110.79801,32.62918
'孝感',4209,113.91645,30.92483
'黄冈',4211,114.87238,30.45347
'恩施',4228,109.48817,30.27217
'咸宁',4212,114.32245,29.84126
'宜昌',4205,111.28642,30.69186
'荆州',4210,112.24069,30.33479
'鄂州',4207,114.89495,30.39085
'随州',4213,113.38262,31.69013
'襄阳',4206,112.12255,32.009
'武汉',4201,114.30525,30.59276
'荆门',4208,112.19945,31.03546
'黄石',4202,115.0389,30.19953
```

id	pos_id	name	latitude	longitude
110101007	1101	朝阳门	116.40717	39.90469

任务三、在 Java 中实现 Kurskal 算法

目标如下:

实现结果类，并以以下形式输出

```
'北京市' - '朝阳门' : 0
'北京市' - '天津市' : 15.8051
```

```
public void union(Edge edge) {
        minSet.add(edge);
    }

    public boolean find(Set<Positions> nodeSet , Edge edge) {
        return nodeSet.contains(edge.ui) && nodeSet.contains(edge.vi);
    }



    public void getMinSpanTree(Graph g,int number){
        Set<Positions> nodeSet = new HashSet<>();
        System.out.println("\n kruskal 最小生成树算法");
        //检查每个边，两个端点 均不 在集合中时将 边 加入集合
```

```
        for(int i=0;i<number;i++){
            boolean isSameSet = find(nodeSet,g.edges.get(i));
            if(!isSameSet){
                nodeSet.add(g.edges.get(i).ui);
                nodeSet.add(g.edges.get(i).vi);
                union(g.edges.get(i));
            }
        }
    }
```

模拟边进行测试

```
[武汉--鄂州: 66.4166968890662, 武汉--黄石: 83.82678594641915, 黄石--武汉:
83.8267859464191, 宜昌--荆州: 107.4591386046547, 襄阳--宜昌: 107.95339891878953,
武汉--荆州: 230.51578110576585, 武汉--襄阳: 251.24612889924953, 荆州--黄石:
312.0939919366912, 宜昌--武汉: 336.70070816892905, 宜昌--黄石:
419.02826404076114]

kruskal 最小生成树算法
[武汉--鄂州: 66.4166968890662, 武汉--黄石: 83.82678594641915, 宜昌--荆州:
107.4591386046547, 襄阳--宜昌: 107.95339891878953]
```

2021年12月21日——（两棵最小生成树）

今天的任务是通过导入数据集"经停站表"，并对其上的城市生成最小生成树。

```
select * from (
select res.train_no,GROUP_CONCAT(res.city),COUNT(res.city) as city_num
from (
select ao.ID,SUBSTRING_INDEX(ao.city,'市',1)as
city,ao.longitude,ao.latitude,st.`name`,st.train_no
from aotude ao,stopposition st where LEFT(ao.city,2) =
LEFT(st.`name`,2))res where LEFT(res.ID,2) = 22 GROUP BY res.train_no)res
where res.city_num>1;
```

train_no	GROUP_CONCAT(res.city)	city_num
1003\1006	十堰,随州	2
1004\1005	随州,十堰	2
1007\1010	十堰,荆门	2
1008\1009	荆门,十堰	2
1125\1128	随州,咸宁	2
1126\1127	咸宁,随州	2
1157\1160	黄石,鄂州,孝感	3

2021年12月22日——（制作交互界面，使其通过下拉列表完成对"省"的选择）

临时任务

协助焦豪完成"用特定类型实现 kruskal 算法"

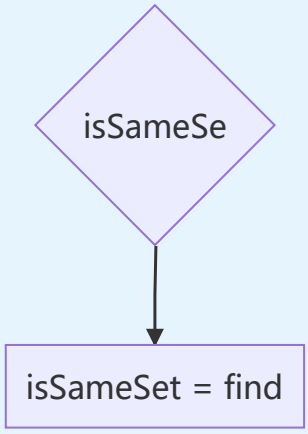
讲解算法流程并同时进行编写

1. 首先我们写出算法运行的主函数

最小生成树算法的目的是向算法传递一张网，然后根据这张网生成对应的最小生成树，所以，我们先传入一张图。

在这个算法中，我们需要判断当前的点是否被加载过，所以我们需要在该算法内载入一个结点的集合。然后因为有“判断”的需要，所以留下一个条件作为判断语句。然后进行循环遍历图的“边的集合”（注意，这个算法运行前需要对边的集合根据权值进行排序，否则结果不一定是最小生成树，也可以选择存储的边的结构使用二叉排序树）。

在循环中我们重复这样一个loop:




```
LinkList* getminTree(Graph* g) {
    LinkList* nodeSet;// 城市集合
    int i = 0;
    int isSameSet = 0;
    while (getEdges(g->edgesSet, i)) {
        isSameSet = find(nodeSet, getEdges(g->edgesSet, i));
        if(!isSameSet){
            addNode(nodeSet, getEdges(g->edgesSet, i)->ui);
            addNode(nodeSet, getEdges(g->edgesSet, i)->vi);
            addNode(g->minTree, getEdges(g->edgesSet, i));
        }
```

```
    }
    return g->minTree;
}
```

2.

协助李文才完成多图连接的数据存储结构和数据导入

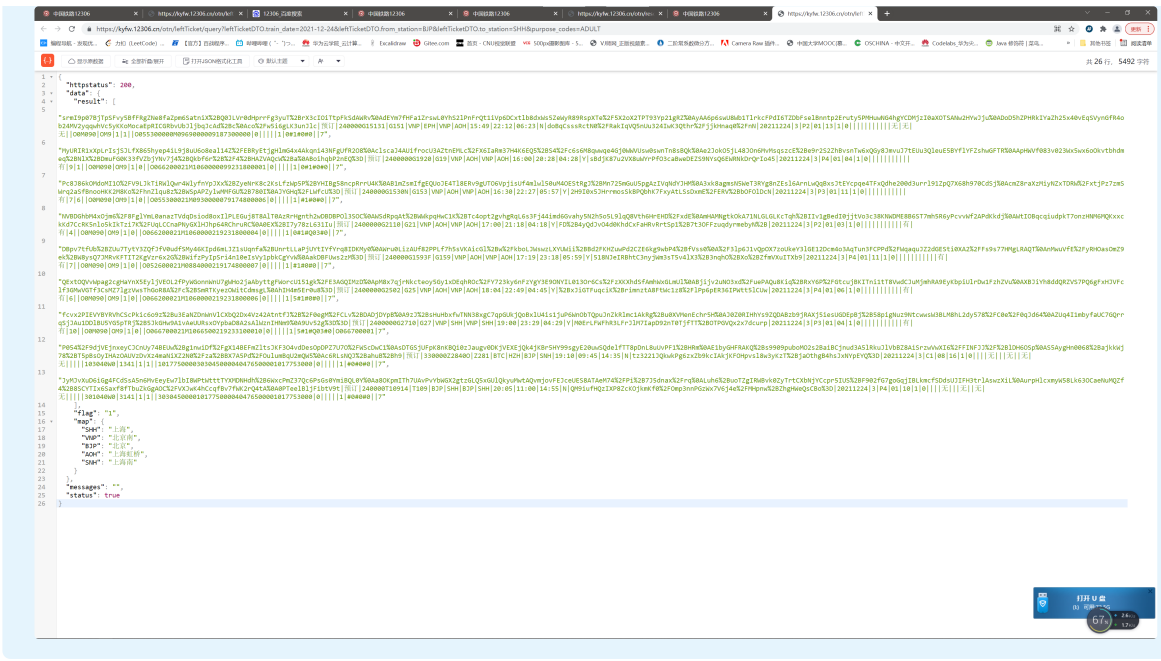
构建 `csv文件` 处理的通用函数 `API` 与文档,便于其他成员使用 `csv文件` 读取数据

2021年12月24日——（紧急任务，12306爬虫）

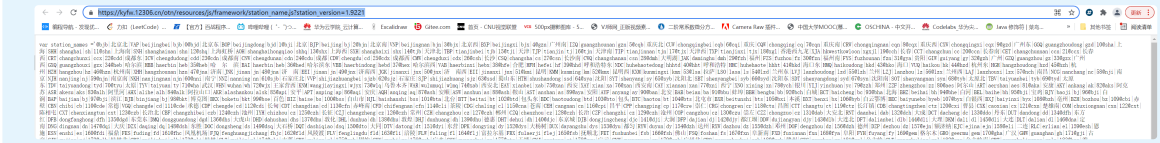
在算法测试中发现现有经停站数据集异常，存在缺站情况，需要增加对 12306 的站点爬虫取得正确数据。

```
https://kyfw.12306.cn/otn/leftTicket/init?linktypeid=dc&fs=%E5%8C%97%E4%BA%AC,BJP&ts=%E4%B8%8A%E6%B5%B7,SHH&date=2021-12-24&flag=N,N,Y
```

```
https://kyfw.12306.cn/otn/leftTicket/query?leftTicketDTO.train_date=2021-12-24&leftTicketDTO.from_station=BJP&leftTicketDTO.to_station=SHH&purpose_codes=ADULT

https://kyfw.12306.cn/otn/leftTicket/init?linktypeid=dc&fs=牡丹江,BBI&ts=莫尔道嘎,MRX&date=2021-12-24&flag=N,N,Y

https://kyfw.12306.cn/otn/leftTicket/query?leftTicketDTO.train_date=2021-12-24&leftTicketDTO.from_station=BBI&leftTicketDTO.to_station=MRXMRX&purpose_codes=ADULT

https://kyfw.12306.cn/otn/leftTicket/query?leftTicketDTO.train_date=2021-12-24&leftTicketDTO.from_station=SNN&leftTicketDTO.to_station=WHN&purpose_codes=ADULT
```



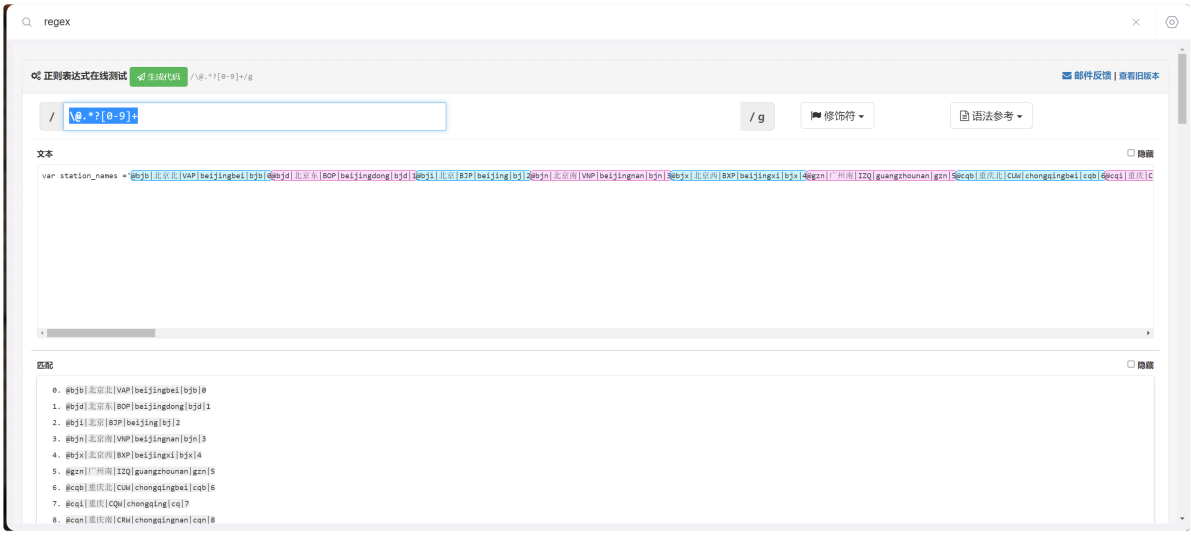
```
https://kyfw.12306.cn/otn/resources/js/framework/station_name.js?station_version=1.9221
```



匹配用的正则表达式

```
\@.*?[0-9]+
//匹配以'@'开头,数字结尾的数据
```

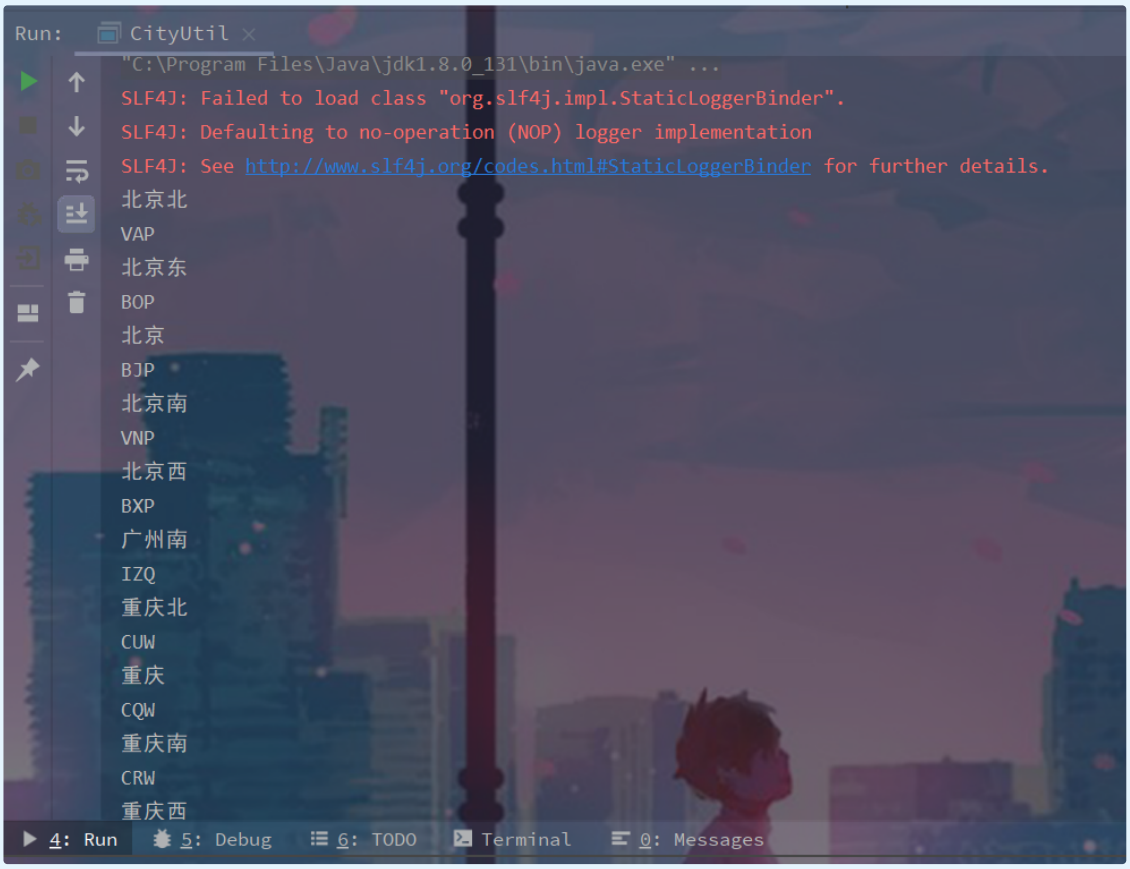
```java
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class RegexMatches {

    public static void main(String args[]) {
        String str = "";
        String pattern = "\\@.*?[0-9]+";

        Pattern r = Pattern.compile(pattern);
        Matcher m = r.matcher(str);
        System.out.println(m.matches());
    }

}
```




```
([a-z]+)|([\u4e00-\u9fa5]+)
//英文匹配|中文匹配
```

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class RegexMatches {

    public static void main(String args[]) {
        String str = "";
        String pattern = "([a-z]+)|([\\u4e00-\\u9fa5]+)";

        Pattern r = Pattern.compile(pattern);
        Matcher m = r.matcher(str);
        System.out.println(m.matches());
    }

}
```

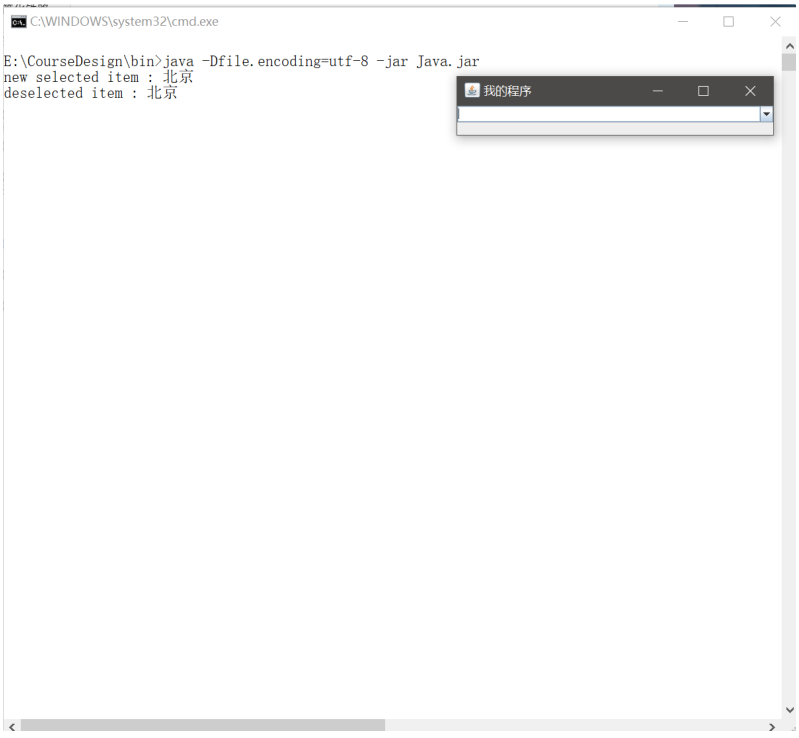
问题发生

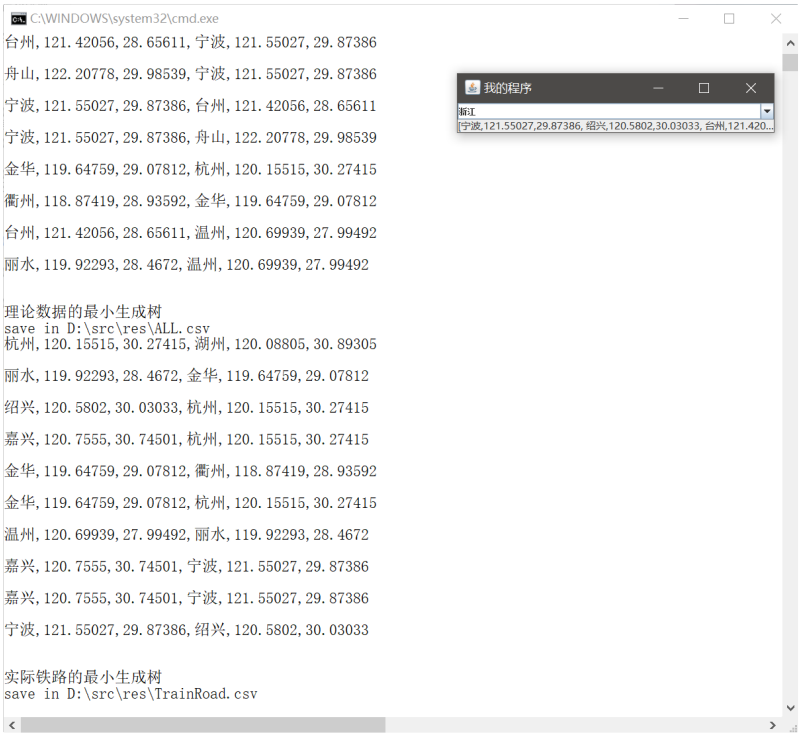



```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
```

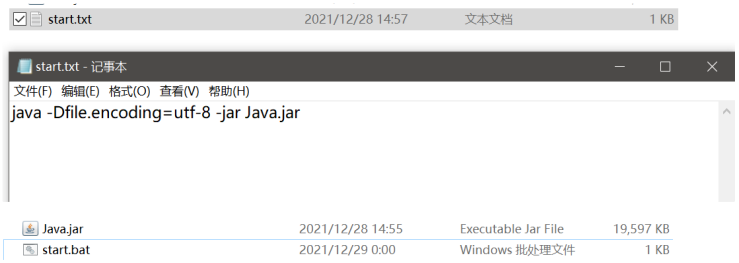
2021年12月23日——（完成22日原有工作）

完成效果





制作一个.bat 代替命令行输入



2021年12月24日——（制作csv文件处理工具）

今天的任务是制作一个csv文件的通用处理工具，用于与其他成员的独立程序沟通并产生反应结果。

.CSV 文件工具方法

2021年12月26日 15时xx分 到 17时38分，完成了对于 csv 文件按行列进行操作的 C语言 工具.
——遴阅.谭

结构组成

```c
typedef char* String;

typedef struct CSVDATAS_COLUMNS {
    String data;
    struct CSVDATAS_COLUMNS* next;
}CSV_COLUMN;

typedef struct CSVDATAS_LINE {
    CSV_COLUMN* columns;
    struct CSVDATAS_LINE* next;
}CSV_LINE;


typedef struct CSVDATAS {
    CSV_LINE* lines;
    int line_num;
    int column_num;
}DATA_CSV;

typedef struct CSVFILEs {
    String path;
    FILE* fp;
    DATA_CSV* data_csv;
}FILE_CSV;
```

1. `FILE_CSV` 类型

   csv 文件类，用于存储指向 csv 文件的指针及数据并保存路径。
2. `DATA_CSV` 类型

   csv 数据类，用于存储文件中每一行的数据
3. `CSV_LINE` 类型

   csv 数据类（行），用于存储一行的数据
4. `CSV_COLUMN` 类型

   csv 数据类（列），用于存储行中的一个数据单元

1. 加载 csv 文件—— (Fileloading_CSV)

```c
//FILE_CSV* Fileloading_CSV(String path, int column_num); //不能自动检查列数
FILE_CSV* Fileloading_CSV(String path);
```

使用方法:

```
//FILE_CSV* csv = Fileloading_CSV("D:\\src\\res\\ALL.csv", col_num);
FILE_CSV* csv = Fileloading_CSV("D:\\src\\res\\ALL.csv");
```

2. 获取行/列

```
CSV_LINE* getLineIndex(DATA_CSV* csv, int row_index);
CSV_COLUMN* getColumnIndex(CSV_LINE* line, int col_index);
```

使用方法：

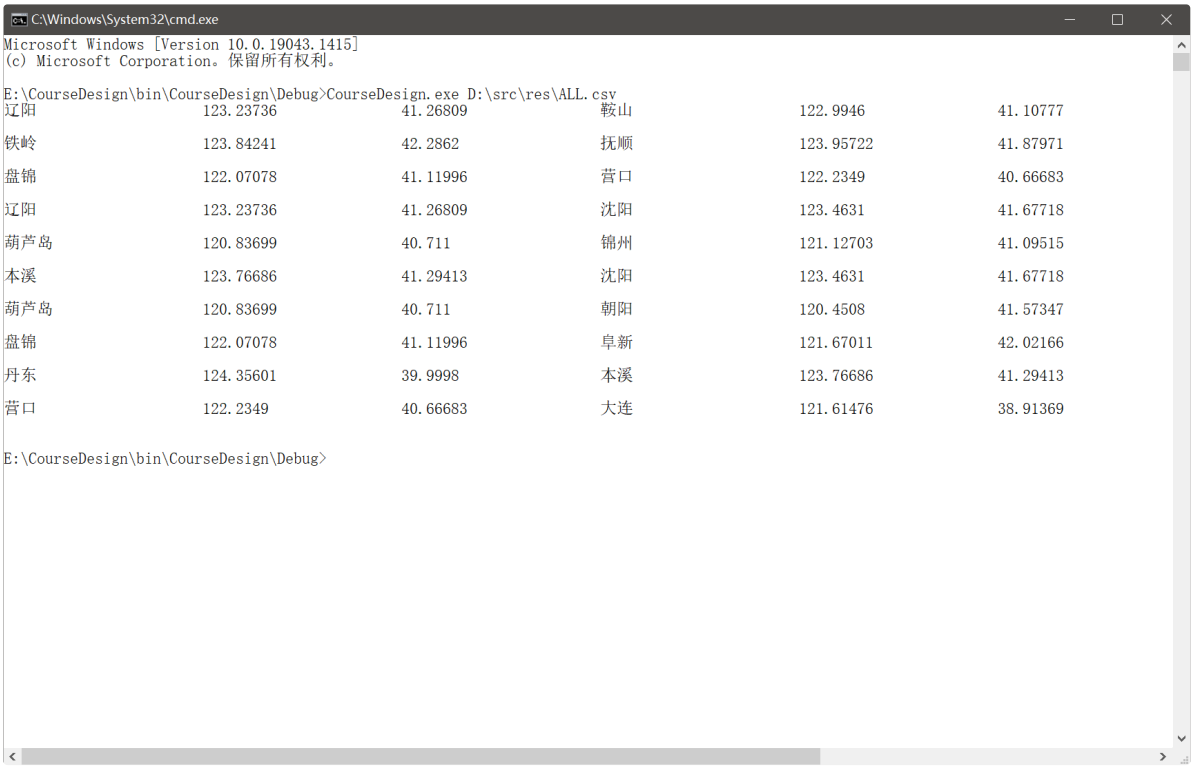
```
String getData(DATA_CSV* csv,int row, int col) {
     return getColumnIndex(getLineIndex(csv,row),col)->data;
}
```

3. 获取特定坐标的单元

```
String getData(DATA_CSV* csv, int row, int col);
```

4. 使用效果演示

```
int main(int argc,char* agrv[]) {
    if (argc > 1) {
        FILE_CSV* csv = Fileloading_CSV(agrv[1]);
        String str = NULL;
        if (csv != NULL) {
            for (int i = 0; i < csv->data_csv->line_num; i++) {
                for (int j = 0; j < csv->data_csv->column_num; j++) {
                    printf("%-23s\t", getData(csv->data_csv, i, j));
                }
                putchar('\n');
            }
        }
        return 0;
    }
    return 1;
}
```



5. 代码

```c
#pragma once /*CSV_Util.h*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#pragma warning(default:4996)

typedef char* String;

typedef struct CSVDATAS_COLUMNS {
    String data;
    struct CSVDATAS_COLUMNS* next;
}CSV_COLUMN;

typedef struct CSVDATAS_LINE {
    CSV_COLUMN* columns;
    struct CSVDATAS_LINE* next;
}CSV_LINE;


typedef struct CSVDATAS {
    CSV_LINE* lines;
    int line_num;
    int column_num;
}DATA_CSV;

typedef struct CSVFILEs {
    String path;
    FILE* fp;
```

```
    DATA_CSV* data_csv;
}FILE_CSV;
/*founction area*/
//FILE_CSV* Fileloading_CSV(String path, int column_num); //不能自动检查列数
FILE_CSV* Fileloading_CSV(String path);

DATA_CSV* Data_loading_CSV(FILE* fp, int column_num);
//FILE_CSV* Fileloading_CSV(String path);
CSV_LINE* addLine(CSV_LINE* tail, String line[], int num);
CSV_COLUMN* addcolumn(String e, CSV_COLUMN* pcolumn);
CSV_LINE* getLineIndex(DATA_CSV* csv, int row_index);
CSV_COLUMN* getColumnIndex(CSV_LINE* line, int col_index);
String getData(DATA_CSV* csv, int row, int col);

```

```
#include "CSV_Util.h" /*CSV_Util.c*/

CSV_COLUMN* addcolumn(String e, CSV_COLUMN* pcolumn) {
    String data = (String)malloc(25 * sizeof(char));
    strcpy(data, e);
    pcolumn->data = data;
    return pcolumn;
}

/// <summary>
/// 对已经产生的空间操作,在这里不使用会导致指针tail变化的方法
/// </summary>
/// <param name="tail"></param>
/// <param name="line"></param>
/// <param name="num"></param>
/// <returns></returns>
CSV_LINE*addLine(CSV_LINE* tail,String line[],int num) {
    if (tail == NULL)return NULL;
    String str = NULL;
    CSV_COLUMN* columns = (CSV_COLUMN*)calloc(1, sizeof(CSV_COLUMN));
    CSV_COLUMN* p = columns;
    int i = 0;
    for (; i < num; i++) {
        str = line[i];
        addcolumn(str, p);
        p->next = (CSV_COLUMN*)calloc(1, sizeof(CSV_COLUMN));
        p = p->next;
    }

    tail->columns = columns;
    return tail;
}
```

```
33
34  DATA_CSV* Data_loading_CSV(FILE* fp,int column_num) {
35      if (fp == NULL)return fp;
36
37      DATA_CSV* data_csv = (DATA_CSV*)calloc(1, sizeof(DATA_CSV));
38      data_csv->lines = (CSV_LINE*)calloc(1, sizeof(CSV_LINE));
39      CSV_LINE* pline = data_csv->lines;
40      CSV_LINE* pp = NULL;
41
42
43      String line = NULL; String record = NULL;
44      String* str = NULL;
45      str = (String*)calloc(column_num, sizeof(String));
46      int i = 0;
47      char buffer[1024];
48
49      char delims[] = ",";
50      char* result = NULL;
51      int line_num = 0;
52
53      while ((line = fgets(buffer, sizeof(buffer), fp)) != NULL)
54      {
55          record = strtok(line, ",");
56          while (record != NULL)
57          {
58              if (strcmp(record, "Ps:") == 0)
59                  return 0;
60              str[i] = record;
61              /*printf("%-23s\t", str[i]);*/
62              i++;
63              if (i == column_num) {
64                  addLine(pline,str,column_num);
65                  line_num++;
66                  pline->next = (CSV_LINE*)calloc(1, sizeof(CSV_LINE));
67                  /*pp = pline;*/
68                  pline = pline->next;
69                  i = 0;
70              }
71              record = strtok(NULL, ",");
72          }
73      }
74      /*free(pp->next);
75      pp->next = NULL;*/
76
77
78      if (data_csv != NULL) {
79          data_csv->column_num = column_num;
80          data_csv->line_num = line_num;
81      }
```

```
82
83      return data_csv;
84  }
85
86  FILE_CSV* Fileloading_CSV(String path) {
87      int column_num = 0;
88      FILE_CSV* csv = (FILE_CSV*)calloc(1, sizeof(FILE_CSV));
89      csv->path = path;
90      if (!(csv->fp = fopen(path, "a+"))) {
91          printf("ERROR: FILE OPENING FAILURE!\n");
92          return NULL;
93      }
94
95      char str[128] = { NULL };
96      String record = NULL;
97      if (fgets(str, 128, csv->fp) == NULL)return NULL;
98      int i = 0;
99      if (record = strtok(str, ",") != NULL)i = 1;
100     for (; (record = strtok(NULL, ",")) != NULL; i++);
101     column_num = i;
102     rewind(csv->fp);
103
104
105     csv->data_csv = Data_loading_CSV(csv->fp,column_num);
106     fclose(csv->fp);
107     return csv;
108 }
109
110 /// <summary>
111 /// 根据行号 i 获取行
112 /// </summary>
113 /// <param name="csv"></param>
114 /// <param name="i"></param>
115 CSV_LINE* getLineIndex(DATA_CSV*csv,int row_index) {
116     CSV_LINE* p = csv->lines;
117     int i = 0;
118     for (; i < row_index&&p!=NULL; i++) {
119         p = p->next;
120     }
121     if (p == NULL) {
122         printf("ERROR: The %d row does not exist \n(\t--
    getLineIndex(DATA_CSV*csv,int row_index))\n", row_index);
123         return NULL;
124     }
125     else {
126         return p;
127     }
128 }
129
```

```
/// <summary>
/// 根据列号 i 获取列结构
/// </summary>
/// <param name="line"></param>
/// <param name="i"></param>
CSV_COLUMN*getColumnIndex(CSV_LINE* line, int col_index) {
    if (line == NULL) {
        return NULL;
    }
    CSV_COLUMN* p = line→columns;
    int i = 0;
    for (; i < col_index&&p≠NULL; i++) {
        p = p→next;
    }
    if (p == NULL) {
        printf("ERROR: The %d row %d col does not exist \n(\t--
getColumnIndex(CSV_LINE* line, int col_index))\n",i, col_index);
        return NULL;
    }
    else {
        return p;
    }
}

/// <summary>
/// 获取第 row 行,第 col 列的 data
/// </summary>
/// <param name="csv"></param>
/// <param name="row"></param>
/// <param name="col"></param>
/// <returns></returns>
String getData(DATA_CSV* csv,int row, int col) {
     return getColumnIndex(getLineIndex(csv,row),col)→data;
}
```

2021年12月25日——（协助李文才调整图形化显示效果）

1. 决定定义位置关系的方式
2. 商议图形化显示的经纬度到特定分辨率的定位方式。

2021年12月26日——（将完成的图形显示程序接入算法程序）

1. 程序连接

```java
package com.ly.GUI;


import com.ly.Graph.Province;
import com.ly.JDBC.DbUtil;
import com.ly.Web.CityUtil;
import com.ly.Graph.Graph;
import sun.plugin2.message.SetChildWindowHandleMessage;

import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.*;
import javax.swing.*;
import javax.swing.event.PopupMenuEvent;
import javax.swing.event.PopupMenuListener;

/*
 * 演示JComboBox的基本用法，以及事件响应
 */
public class JComboBoxBasicUseDemo extends JFrame {
    private static final long serialVersionUID = -8161981948004677531L;
    int DEFAULT_WIDTH = 500;
    int DEFAULT_HEIGHT = 100;
    private JLabel label;
    private JLabel textCity;
    private JList jl1;
    private JList jl2;
    private JComboBox<String> faceCombo;
    private static final int FONTSIZE = 15;
    private static boolean flag = false;
    private static Process process = null;
    @SuppressWarnings("unused")
    public JComboBoxBasicUseDemo() throws SQLException {
        DbUtil db = new DbUtil();
        List<Province> provinces = new LinkedList<>();
        String id = null;
        final Graph[] graph = {new Graph()};
```

```
48
49          ResultSet rs = db.getResultSet("SELECT id,`name` FROM
`detaillevel` where deep=0;");
50          //loading <id,province>set into program;
51          while (rs.next()) {
52              provinces.add(new Province(rs.getString(1),
rs.getString(2)));
53          }
54          /*jl1 = new JList();
55          jl1.setFixedCellWidth(100);
56          jl1.setFont(new Font("Serif",Font.PLAIN,FONTSIZE));*/
57
58          setTitle("我的程序");
59          setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
60          //添加label
61          label = new JLabel("各省理论最小生成树与实际铁路数据的最小生成树对比图形
化");
62          label.setFont(new Font("微软雅黑", Font.PLAIN, 30));
63          add(label, BorderLayout.NORTH);
64
65          textCity = new JLabel();
66          //actionListener
67          ActionListener actionListener = new ActionListener() {
68              public void actionPerformed(ActionEvent e) {
69                  if (e.getSource() instanceof JComboBox) {
70                      @SuppressWarnings({"unchecked", "rawtypes"})
71                      JComboBox<String> comboBox = (JComboBox)
e.getSource();
72                      String fontName =
comboBox.getSelectedItem().toString();
73                      label.setFont(new Font(fontName, Font.PLAIN,
FONTSIZE));
74                      System.out.printf("%s%n", "actionPerformed called");
75                  }
76              }
77          };
78          //popupMenuListener
79          PopupMenuListener popupMenuListener = new PopupMenuListener() {
80              @Override
81              public void popupMenuCanceled(PopupMenuEvent e) {
82                  System.out.println("下拉菜单取消");
83              }
84
85              @SuppressWarnings("unchecked")
86              @Override
87              public void popupMenuWillBecomeInvisible(PopupMenuEvent e) {
88                  System.out.println("下拉菜单合上");
89                  JComboBox<String> source = (JComboBox<String>)
e.getSource();
```

```
            }

            @Override
            public void popupMenuWillBecomeVisible(PopupMenuEvent e) {
                System.out.println("下拉菜单弹出");
            }
        };
        //itemListener
        ItemListener itemListener = new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent arg0) {
                // TODO Auto-generated method stub
                if (ItemEvent.SELECTED == arg0.getStateChange()) {
                    String selectedItem = arg0.getItem().toString();
                    label.setFont(new Font(selectedItem, Font.PLAIN,
FONTSIZE));
                    System.out.printf("new selected item : %s%n",
selectedItem);
                    if (flag) {
                        for (int i = 0; i < provinces.size(); i++) {
                            if
(provinces.get(i).isSameProvince(selectedItem)) {
                                if((graph[0] =
Graph.minTreeLoading(provinces.get(i).getId()))≠null){
                                    String str =
graph[0].getPositions().toString();

 /*jl1.setListData(graph[0].getPositions().toArray());*/

 textCity.setText(graph[0].getPositions().toString());
                                    textCity.setFont(new Font("微软雅黑",
Font.PLAIN, FONTSIZE));
                                    add(textCity, BorderLayout.SOUTH);
                                    process =
openExe("E:\\CourseDesign\\bin\\Graph.exe
D:\\src\\res\\ALL.csv,D:\\src\\res\\TrainRoad.csv,1920,1080");
                                    flag = false;
                                }
                            }
                        }
                    }
                }
                if (ItemEvent.DESELECTED == arg0.getStateChange()) {
                    closeExe(process);
                    flag = true;
                    textCity.setText(null);
                    String selectedItem = arg0.getItem().toString();
                    System.out.printf("deselected item : %s%n",
selectedItem);
```

```
                }
            }
        };
        //添加一个JComboBox
        faceCombo = new JComboBox<String>();
        faceCombo.setEditable(true);
        faceCombo.addItemListener(itemListener);
        faceCombo.setEnabled(true);

        rs = db.getResultSet("SELECT `name` FROM `detaillevel` where
deep=0;");
        while (rs.next()) {
            faceCombo.addItem(rs.getString(1));
        }
        faceCombo.setLightWeightPopupEnabled(true);
        add(faceCombo, BorderLayout.NORTH);
        faceCombo.setSelectedIndex(-1);

    }

    public static Process openExe(String path) {
        final Runtime runtime = Runtime.getRuntime();
        Process process = null;

        try {
            process = runtime.exec(path);
        } catch (final Exception e) {
            System.out.println("Error exec!");
        }
        return process;
    }

    public static boolean closeExe(Process process) {
        if (process != null) {
            process.destroy();
            return true;
        }
        return false;
    }

    public static void main(String[] args) throws SQLException {
        // TODO Auto-generated method stub
        //创建窗体并指定标题
        JComboBoxBasicUseDemo frame = new JComboBoxBasicUseDemo();
        //关闭窗体后退出程序
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //自动适配所有控件大小
        //frame.pack();
        //设置窗体位置在屏幕中央
```

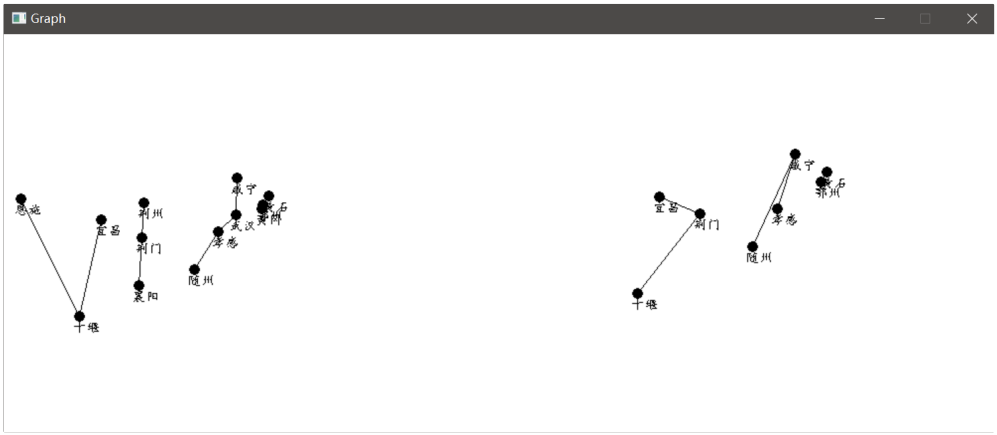
```
176             frame.setLocationRelativeTo(null);
177             //显示窗体
178             frame.setVisible(true);
179         }
180     }
```

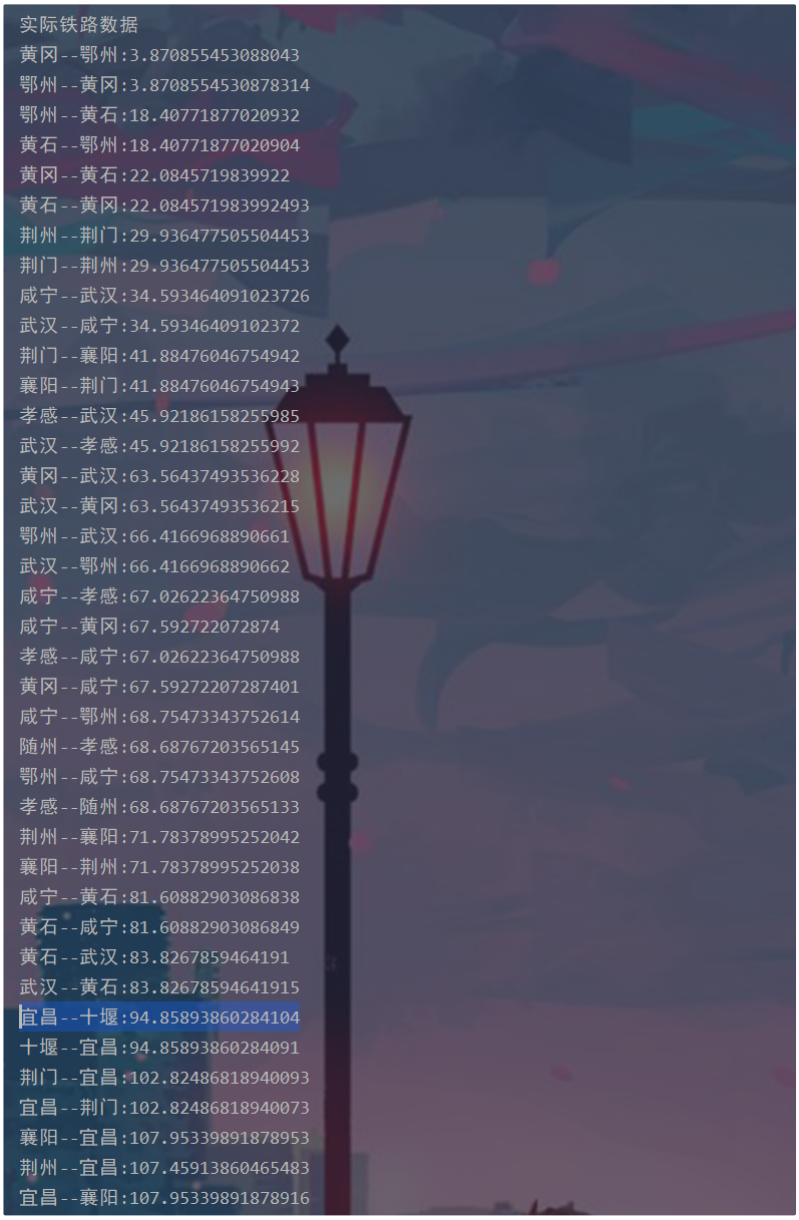
2. 测试

问题记录：测试结果异常，数据不构成最小树。

2021年12月27日——（解决测试出现的问题）

问题表现

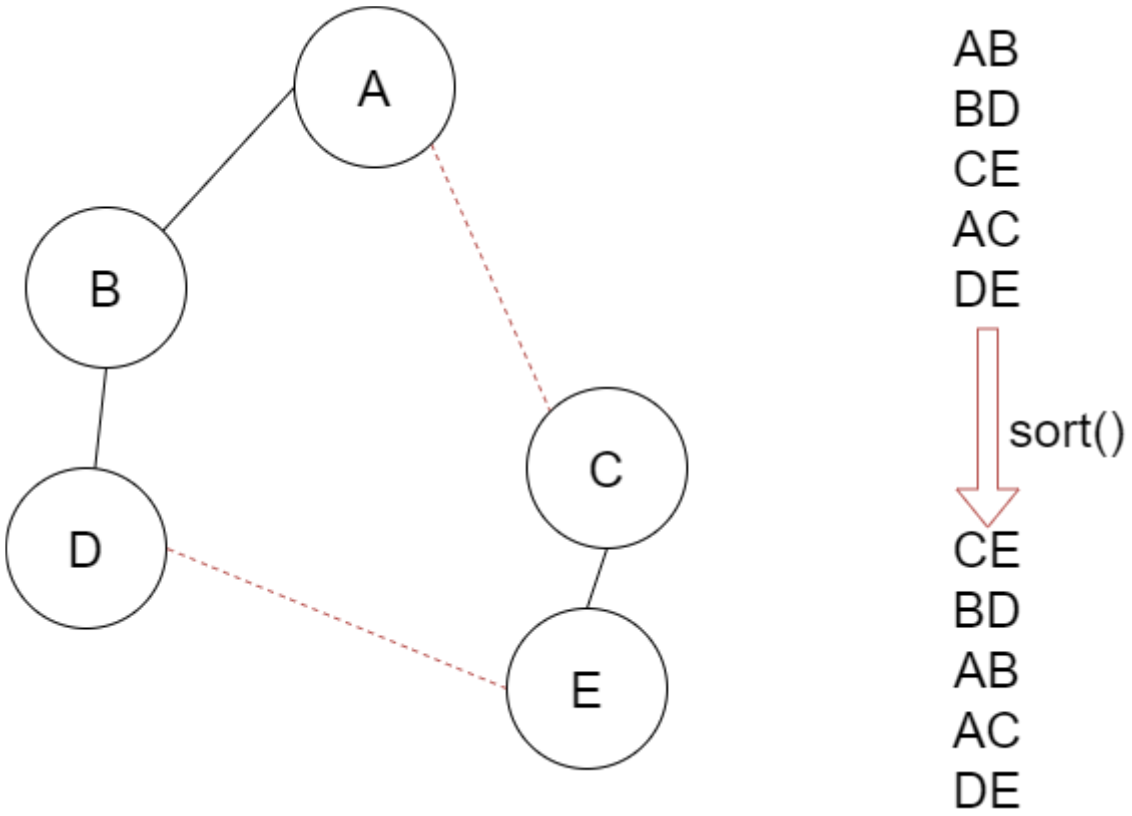




1． 上方的测试案例可以看出最小生成树的算法异常

查看后发现原因：

我编写该算法时采用了另一种思路，"点集合"与"边集合分离"，通过遍历排序过的边集合得到生成树，而这个做法使得判断一类情况时会对正确结果产生错误判断。



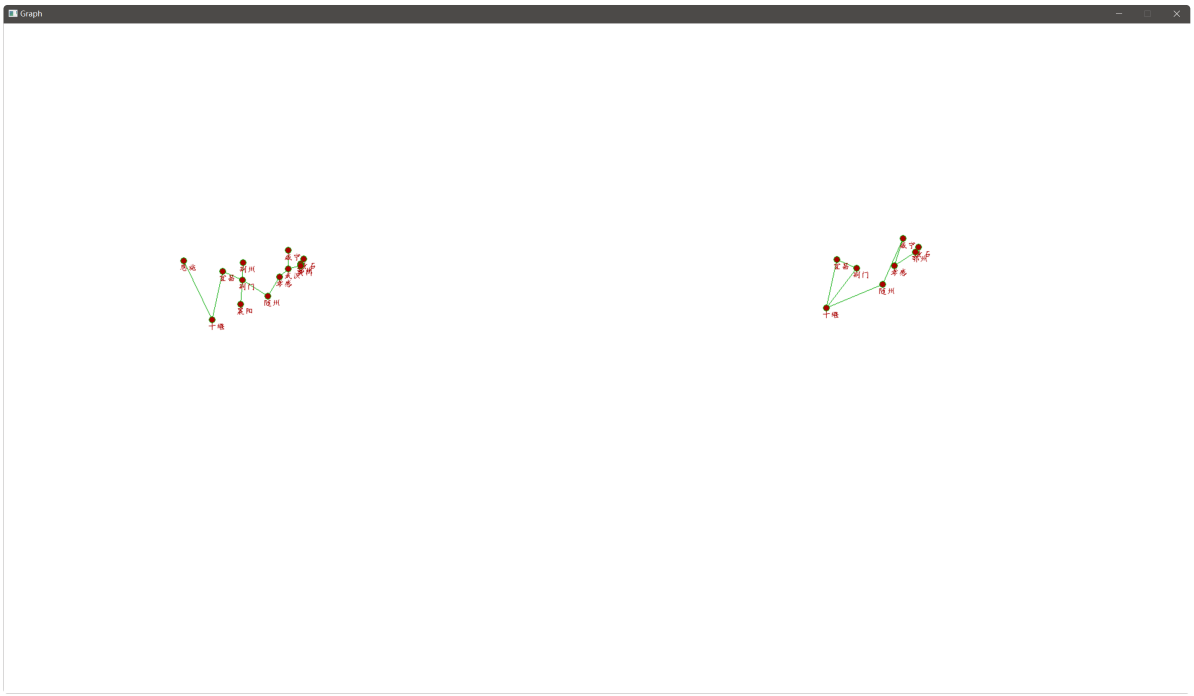
这个错误的 Kruskal 算法在进行时会得到图的所有最小连接，也就会产生错误中那样的因为最短边连接导致各路径的边无法连接。产生问题的原因是我在设计算法时认为“优先连接最小边更容易得到最小生成树”。我把这个问题称为AC问题，AC问题的发生是由于通过已排序边为添加顺序，同时将图中包含的点作为是否保留边作为条件，这样的操作导致了当图的两棵子树边各自加载权最小边后，由于所有点都处于点的并集内，所以无法为两棵树形成连接，以至于图形成了多段最小连接。为了解决这个问题，我提出了以下方案:

1. 当A、C都在点集中且AC边不在结果中时，假设AC已加入，是否构成环
2. 当A、C都在点集中且AC边不在结果中时，设置pA、pC，让它们从A、C出发，看是否相遇（利用分治简化方案一）
3. 加入AC问题的实质是A、C与当前结果集的所有点是否同时构成关系的问题，所以对这里使用 kruskal 算法，就能得到AC是否能添加的结果(true/false)

前两个方案由于边是以集合形式存在且相互之间没有数据上的直接关联,所以在本次任务中算法复杂度太高,我没有进行实践。对于第三个方案的设想，我做出了一些改变并直接应用于算法修改。接下来的内容是修改后的代码，测试结果和解释。

```
public LinkedList<Edge> getMinSpanTree(LinkedList<Edge> set) {
        Positions[] pos = new Positions[set.size()];
        Set<Positions> positions = getAllPositions(set);
        Iterator<Positions> it = positions.iterator();
        int i = 0;
        while(it.hasNext()){
            pos[i] = it.next();
            i++;
        }

        int[] parent = new int[set.size()];
        for(i=0;i<set.size();i++){
            parent[i] = 0;
```

```
        }

        Double sum = 0.00;
        for(Edge edge:set){
            int start = find(parent,getIndex(pos,edge.ui));
            int end = find(parent,getIndex(pos,edge.vi));

            if(start≠end){
                parent[start] = end;
                System.out.println(edge);
                minSet.add(edge);
                sum += edge.Weight;
            }
        }
        return minSet;
    }
```



问题解决思路

我想要在不融合数据的情况下使得这个结果以原有结构运行，也就是需要将当前的点集和算法内部的"key-value"数组建立联系。我将点在点集中的位置通过一段转换等价，从而保持 int 数组的 kv 关系不发生改变，从而使得整个算法不发生修改。

思考

我认为这个解决思路是对于结构类型相同数据的通解。但如果将数据地址与value关联，或许可以对任意数据与算法建立联系。