2015 级《数据结构》作业第二题

有向图最短路径与边界数计算

专业名称：地理科学与规划学院

学　　号：15303096

姓　　名：罗皓文

联系方式：luohw3@mail2.sysu.edu.cn

指导教师：李秋萍

完成时间：2017 年 7 月 11 日 星期二

目 录

1. 实验内容

实验数据：

节点数据：node.csv，3924 条记录；

道路数据：road.csv，5677 条记录。

实验步骤：

1.构建有向图，将数据读入到又像图中；

2.计算任意两点之间的最短距离；

3.计算任意一条道路的边界数；

4.使用边界数进行社区分割

2. 实验软硬件环境

系统环境：MacBook Air (13-inch, Early 2015)／macOS Sierra

处理器：1.6 GHz Intel Core i5

内存：4 GB 1600 MHz DDR3

IDE：Xcode Version 8.0 (8A218a)

语言：C++

3. 模块/函数说明

类：CGraph、CNode、CEdge

CNode 储存节点，CEdge 储存道路，节点之间以链表的形式储存，节点有数据成员 mpE 指向从 CNode 出发的道路，道路之间以链表的方式储存。

CGraph 为图，通过 CGraph 找到图的头指针，可以遍历全图，提供搜索功能，使用 search 函数可以通过 FID 或坐标找到任一节点。可以使用成员函数 beTwNess 计算边界数，使用成员函数 comStruct 可以实现社区分割。

CNode 为节点，使用函数 getPathGroup 可以计算以该点为起点到其他点的最短路径集合。函数 printNearestPath 输出到某点的最短路径详细信息。

CEdge 为边界，mnBTN 用于记录边界数。

核心函数为 getPathGroup，beTwNess，comStruct。

getPathGroup 使用迪杰斯特拉算法计算从某点出发到全图任意一点的距离与路径集合，每次循环通过 getNearNode 函数更新当前数组，并计算当前最短路径集合中距离最短的路径。基于此路径进行下一次循环。最后通过函数 printNearestPath 可以输出某最短路径的详细信息。算法时间复杂度为 N^2。

beTwNess 构建两个矩阵，利用弗洛伊德算法计算任意两遍的最短路径与其距离，然后基于这两个矩阵，统计每条道路的边界数，算法时间复杂度为 N^3。计算边界数时将路径不断拆分为两部分，如果为一步到达（只经过一条路的路径）就使得边界数增加当前记录的数，否则拆分为两条路径继续循环，直到所有路径都被计算完全为止，时间复杂度为 NlogN。

comStruct 使用阈值，依次从各个点出发，若出发点已标记则不处理，未标记则标记为新的一类，从某点出发到达邻接点且连接两点的路径边界数小于阈值，标记为同一类，然后对该点进行迭代操作，知道所

有点都标记为止，时间复杂度大于 N 小于 N^2。

4. 实验结果

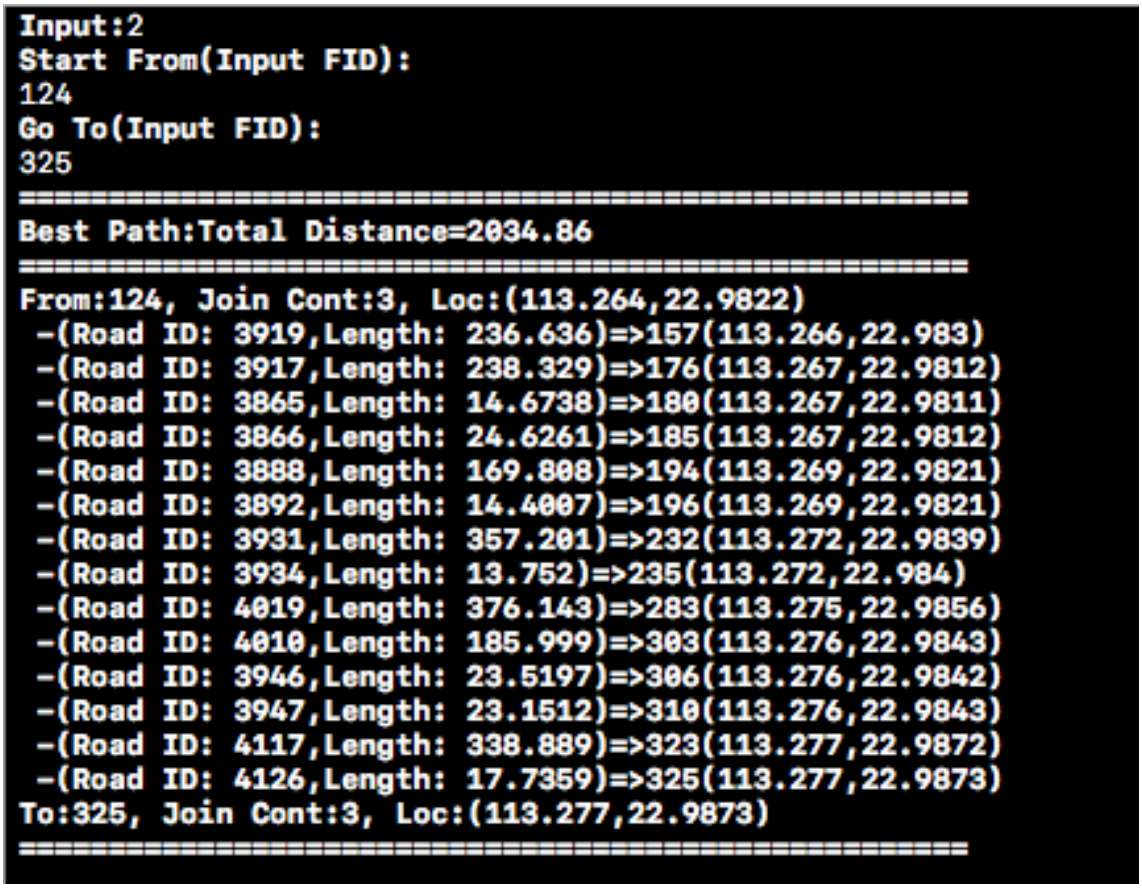
4.1. 实验流程

1.读入数据，显示数据。

2.计算任意两点最短距离

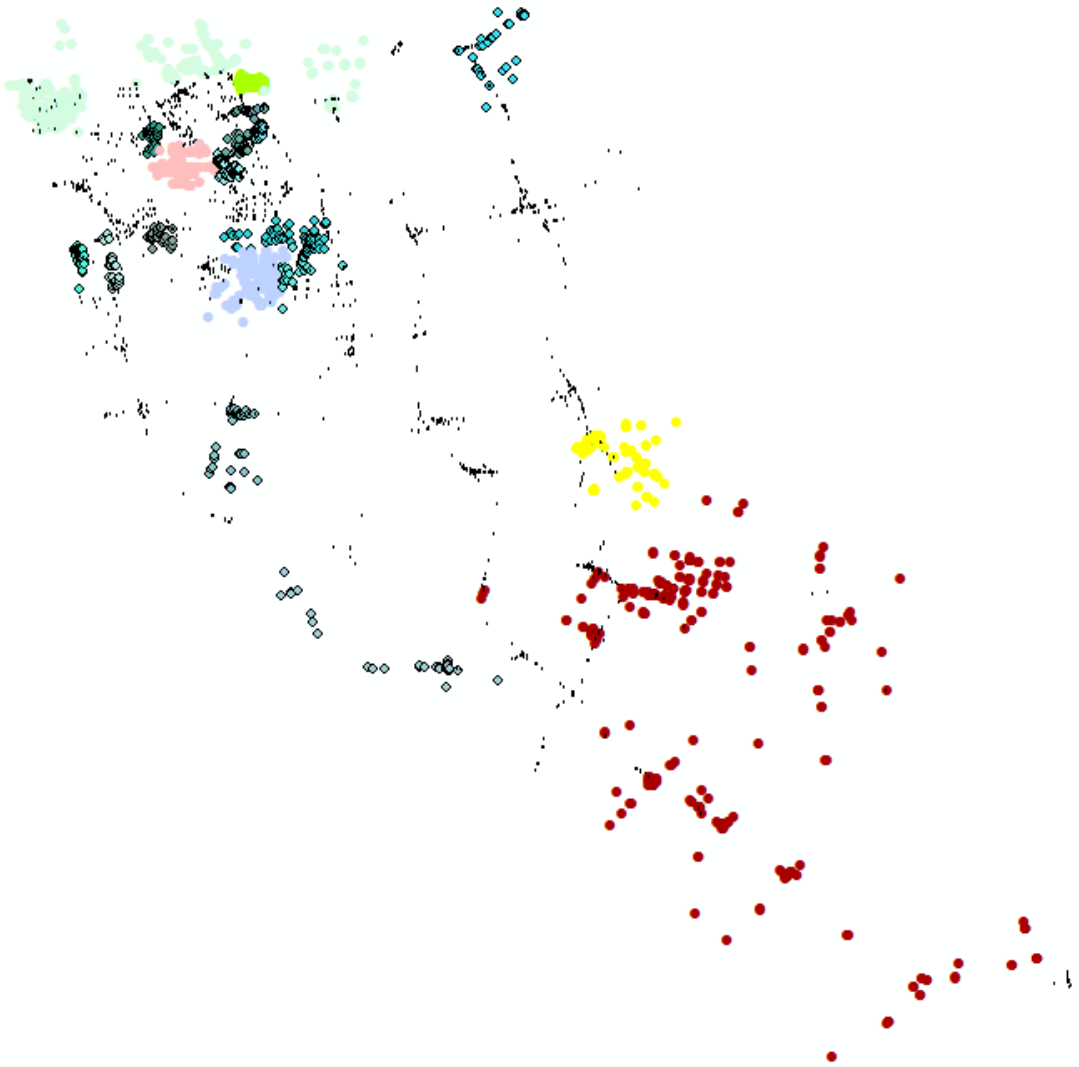
3.计算所有道路边界数

4.进行社区分割



上图为最短路径计算。

边界数结果输出为 csv 文件保存。



如图，社区分割将社区分成几个主要部分。

4.2. 核心函数运行时间

display：显示全图用时 0.104447s；

printNearestPath：计算任意两点最短路径用时 0.254806s

三次实验分别为：0.270535，0.246052，0.24783

beTwNess：任意两点最短距离计算时间 616.990287s（约 10 分钟）边界数统计用时 0.457783s，共 617.448070s.（第二次实验 596.197s）

comStruct：分割用时 0.019361s

5. 总结

本次实验，使用迪杰斯特拉算法计算了最短路径，并用弗洛伊德算法计算了最短距离矩阵，得出边界数，实现社区分割。

总的来讲，实验完成度高，部分性能较为优越，提出（改进）了动态统计边界数和阈值切断路径的算法，但对于弗洛伊德算法求任意两点最短路径时效率较低，可以考虑降低精度使用较快的算法得出有统计意义的结果。