数据结构2015PJ——时空数据管理

14307130318

刘超颖

一、开发环境

以Sublime作为编辑器，使用了HTML、JavaScript语言，调用了JQuery库。

二、所开发软件使用说明

1. 用浏览器打开click.html文件。

2. 通过下拉菜单选择要看的上海地图区域，并点击右侧draw按钮进行画图，界面如图所示。

3. 通过下拉菜单选择地图功能，提供“最短路查询”，“范围查询”，“近邻查询”等功能。

4. 最短路查询：在功能下拉菜单中选择“Shortest Path”，在地图上随意点击两个点，地图将为您画出两点之间的最短路程，此条最短路沿公路行驶，支持点击的起点或终点不在道路上，地图将为您找到离起点、终点最近的公路开始查询最短路。支持对单行道和路的方向性的处理。支持多条最短路查询，即每点两个点作为一组起点、终点。如果查询不到最短路（即可能发生两点不能连通、起点或终点在不正确的位置等情况），弹框跳出“找不到最短路”。

5. 范围查询：在功能下拉菜单中选择“Range Query”，在地图上随意点击一个点，地图将为您查询距离这一点周围以40像素为半径的圆内的所有兴趣点，即含有name标签的特殊节点。将结果弹窗跳出。效果如图：

6. 近邻查询：在功能下拉菜单中选择“Neighbour Query”，在地图上随意点击一个点，地图将为您查询距离这一点最近的兴趣点，即含有name标签的特殊节点。将结果弹窗跳出，且画出所在点到最近邻兴趣点的最短路。

7. 出租车轨迹查询：在出租车编号的文本框中输入所查出租车的编号，点击find按钮进行轨迹基本信息查询，将出租车轨迹信息用弹窗跳出。

8. 出租车拦招地点推荐：在功能下拉菜单中选择“Taxi Query”，在地图上随意点击一个点，找到距离该点一定范围内的所有易拦招出租车的点。

三、源代码的设计思路

1. 地图可视化：因为要手动实现通过给定osm（xml）文件画地图的功能，使用HTML 5的canvas进行图像制作。先通过对osm文件的bounds标签对应内容进行分析，确定canvas的height和width。然后将osm文件中的所有标签为node的点取出，将这些点的id和经纬度赋值给一个node数组。然后使用canvas内置的画图函数，将标签为highway的且在同一条路上的所有点连线。为了体现地图的拓扑完整性，将标签为relation的路线用另一种颜色的线画出。

2. 最短路：使用heap优化的Dijkstra算法实现单源起点到单源终点的最短路。首先，我面临的是查找离起点、终点最近的，公路上的点的工作。因为手动在地图上选取起点、终点，不可能刚好点到的点在公路上，所以我进行了最近道路点的处理，因为遍历一遍地图上所有点进行最近道路点查找的复杂度只有O(n)，并不会很慢，所以我简单粗暴地采取了这个方法，找到了离起点、终点最近的道路上的点。其次，

求道路起点到道路终点的最短路的算法，如果只是简单使用Dijkstra算法的话，复杂度是O(n^2)，当我将信息从上海地图中提取出来后，我发现地图上有约500,000个点，将路相连，就是将近500,000条边。如果用复杂度为O(n^2)算法，最短路查询会非常慢。我尝试将最短路径上的每隔五个点的信息打印出来，所用样例运行出结果需要6-7分钟之久。经过堆优化的Dijkstra是个非常普遍的最短路的算法，可以将Dijkstra的复杂度从O(n^2)下降到O(n*logn)，但让我写堆优化我是拒绝的，因为javascript并不像C++有stl之类的库。所以我试图将最短路算法换为SPFA，因为从某种角度上，SPFA比Dijkstra + heap要好写，且单从复杂度来看，SPFA的期望复杂度是O(n)，可能会比Dijkstra + heap要快，且SPFA对稀疏图的处理应比Dijkstra要快。但事实上，当我实现SPFA后，效果并不尽如人意，因为数据过大，且地图并不疏密，使用的样例需数秒才能得出结果。所以我还是回归了Dijkstra + heap，并手写堆优化，所用样例未经严格计时，不过通过实际操作可以得知，堆优化的Dijkstra是非常快的，在1秒之内可以在地图中画出两点之间的最短路。除此之外，在解决最短路问题的时候，我们还面临着一个问题，即我们可以认为每条公路都是一个图层，在画图操作时，十字路口等因素不受这个问题的影响，而在计算最短路时，所有的路与路之间的交点，不一定是两条路的公共点，即此时很容易找不到最短路，且找到的最短路往往不能转弯。故我们将地图上路与路之间的交点（约30,000个）全部加入之前存的node数组中，再进行最短路查询操作，即解决了这一问题。总的来说，在时间效率上，Dijkstra + heap > SPFA > Dijkstra。

3. 近邻查询：使用K-D树实现。首先，先解析osm文件，将文件中标签有类似于 <tag k="amenity" v="restaurant"/>，出现restaurant、supermarket等关键字的点的经纬度取出。若使用线性扫描法，即将数据集中的点与查询点逐一进行距离比较，也就是穷举，缺点很明显，就是没有利用数据集本身蕴含的任何结构信息，搜索效率较低。所以我们使用K-D树这种数据结构，将空间和数据集进行细分，直到每个空间中只包含一个数据点，每次细分之前空间内的点即跟节点。同时，将鼠标点击位置的经纬度取出。通过二叉搜索，顺着搜索路径很快就能找到最邻近的近似点，再通过回溯操作，找到最邻近点。KD树的创建算法的复杂度是O($n * \log(n)^2$)。

4. 范围查询：使用Brute-Force算法循环遍历实现，不多做赘述。

5. 出租车轨迹查询：首先，将csv文件导入数据库，数据库使用B树组织数据，然后按照出租车编号来选出对应的出租车轨迹信息。

四、源代码中涉及的数据结构和算法

1. Dijkstra + heap：从一个顶点到其余各顶点的最短路算法，解决有向图中最短路径问题。未经过堆优化前复杂度是O(n^2)，由于此处用到边数远小于点数的平方，故用堆这种数据结构进行优化，取出已有最短路径的复杂度降为O(1)，总复杂度降为O(n*logn)。

2. SPFA：Bellman-ford算法基础上加队列优化求单源最短路径的算法。期望复杂度为O(n)，，通过边增广，在稀疏图上快。

3. K-D树：每个节点都为k维点的二叉树，每个节点表示一个空间范围，用于最邻近查找。

五、其他

在源代码设计过程中，我采取了divide-and-conquer设计模式，把整个大问题切分成了数个小问题，针对每个模块，我分别设计了不同的算法，用模块化的方法进行高效编程。具体体现为我把最短路模块同时应用到了最短路查询和最近邻点查询两个功能中。

因为pj在画图过程中调用了jquery库，如果简单地将pj资源中给出的shanghai_map导入到环境中，会产生堆栈溢出的error。我尝试解决这一问题，列出了两种解决方案，其一是自己写一个类似jquery的东西解析xml文件，但这对于我来说实在是太困难了！所以我采取了第二种解决方案，即对整个

shanghai_map进行分析，划分为数个区域的map，在调用jquery的过程中果然没有产生之前的错误。但是这一解决方案也有一定问题，即区与区之间可能存在重复的部分，也可能存在两个区都不包含的部分。

每次模式更改时，对点击点数的计数器cnt置零。通过对cnt的维护修复了一些bug。

除此之外，在canvas的使用中，对鼠标点击的监听时，遇到了一些问题，即每次点击都会新开一个线程，导致一些bug，比如不慎在地图上点击一下后再换区域会导致错误最短路被画出。之后修改了这个bug，并进一步实现了多条最短路的查询。

在各项功能中，调用了sweetalert开源js库，弹出漂亮的ui界面。