图可视化大作业说明文档

团队分工

王泽宇

- 实现最短路、最大生成树和介数、紧密中心度的核心算法API。
- 图论算法需要用到的数据结构。
- 以用户为点建图的附加功能。

凌精望

- GUI和调用逻辑的接口。
- 计算连通分量API。
- 以电影为点建图和对稠密图显示的优化。

环境

操作系统：Windows 10

IDE：Visual Studio 2017， JetBrains WebStorm

编程语言：C++（核心算法），HTML，Javascript（可视化）

GUI架构：electron，使用node-ffi调用C++的dll

GUI使用说明

- 双击/bin/gui/graph-visualize.exe打开GUI界面。
- 点击左下角的连接切换算法
- 拖动、单击的点会高亮并在左面板显示出结点信息。
- 在最短路模式下：
  - 单击一个点，再点击左边的对应按钮，可以将选中点设为起点/终点。
  - 单击“计算最短路”，计算并显示最短路。
- 在中心度模式下
  - 可以在左面版切换染色方案，可视化不同的中心度
  - **如果所有点集聚在左上角，请单击或拖动一下点，方可在中央正常展开显示。**
  - 计算时间较长，请等待上方进度条完成。在C++计算期间单击左下方的链接将不会跳转页面。
- 在连通分量模式下输入边阈值后点击按钮重新计算连通分量。

核心算法API

配对堆

堆的一种。支持以下操作。

- $Merge(u,v)$：合并两个配对堆，直接将v加到u的孩子列表中，时间复杂度$O(1)$。
- $Insert(u,v)$：插入权值为v的节点，先新建一个堆，里面只有权值为v的节点，再将两个堆合并。时间复杂度$O(1)$。
- $Decrease\_value(u,\Delta)$：把节点u减小$\Delta,\Delta\geqslant 0$。如果权值减小后依然比父节点的权值小，则对不需要做任何变化。否则，将u从堆中分离出来，在加入到堆顶节点的孩子列表中（即$Merge$操作）。时间复杂度$O(1)$。
- $Pop()$：把根节点删除掉，并将孩子节点全部$Merge$起来。经证明，复杂度是$O(logn)$。

具体实现参见Graph-theory-dll项目下的Pairing_Heap.h和Pairing_Heap.cpp文件，算法学习参考文献见这里

单源最短路

使用**Dijkstra**单源最短路算法，并利用配对堆优化，时间复杂度$O(m+nlogn)$（n是图的点数，m是图的边数，下同）。具体实现见Graph-theory-dll项目下Graph类的成员函数ShortestPath()。

最大生成树

由于以电影建图的边权是两个电影的共同共同观影人数，故采用**最大生成树**算法。使用**Prim**算法求最大生成树，并利用配对堆优化，时间复杂度$O(m+nlogn)$。具体实现见Graph-theory-dll项目下Graph类的成员函数MST()。

中心度

介数中心度

介数中心的的定义式为$C_B(v)=\sum_{s\neq v\neq t\in V}\frac{\sigma_{st}(v)}{\sigma_{st}}$，其中$\sigma_{st}(v)$表示经过节点$v$的$s\rightarrow t$最短路径条数，$\sigma_{st}$表示$s\rightarrow t$的最短路径条数，直接用Floyd算法时间复杂度为$O(n^3)$。本项目采用的是Brandes在A Faster Algorithm for Betweenness Centrality一文中提到的算法，$C_B(v)$的计算可以简化为$C_B(v)=\sum_{s\neq v}\delta_s(v)=\sum_{s\neq v}\sum_{w:v\in P_s(w)}\frac{\sigma_{sv}}{\sigma_{sw}}(1+\delta_s(w))$，其中$\delta_s(v)=\sum_{t\in V}\frac{\sigma_{st}(v)}{\sigma_{st}}$，$P_s(w)$表示$s\rightarrow w$所有最短路径上的$w$的前驱结点构成集合，具体证明可见A Faster Algorithm for Betweenness Centrality，在此不做赘述。由此可以做对于所有起点s，我们都可以在做一遍单源最短路的情况下，遍历最短路图，可以求出$\delta_s(v),\forall v\in V$，求和即可得到所有的$C_B(v)$。利用之前的单源最短路，总的复杂度为$O(nm+n^2logn)$。具体实现见Graph-theory-dll项目下Graph类的成员函数Betweenness_Centrality()。

紧密中心度

紧密中心度的定义为$C_C(v)=\frac{N-1}{\sum_{u\neq v}d_{uv}}$，本项目采用**并行Floyd()算法**，线程池类的实现ThreadPool参考自这里。具体实现见Graph-theory-dll项目下Graph类的成员函数Closeness_Centrality()和Floyd()。

连通分量

使用dfs的算法实现找到并标记连通分量，输出连通分量的标号和**位于dfs树上的边**。

电影为结点的建图

电影边权说明：为同时看过这两部电影的用户的个数，如果用户数大于边阈值则认为有边。

优化显示

在最短路和中心度模式下，不能显示出所有的边。

采用的策略是：

- 建立一棵原图的最大生成树，将生成树上的边纳入显示。
- 对于每条边，如果它的两个端点的度数均为达到5，则将这条边加入显示。
- 将最短路的关键路径上的边纳入显示。
- 凡是不满足以上条件的边，不在GUI中显示。

网络构建

从给定的数据集/input/movie.csv和/input/user.csv中读取数据，用户需要自己指定边的阈值（为了保证边数不过大，阈值需大于等于2，且是整数），程序会根据阈值以用户为节点建立无向带权图（大约需要10~20s）。并计算连通分量connected_component()。最后程序会按照TEXT和JSON两种格式输出。

用户边权说明：两个用户间的共同看过的电影数作为两者边权的整数部分；将用户看过的电影的评分四舍五入为0~10的整数，用户评分向量的t维统计看过的电影得分为t的数量，并作归一化处理。两个用户的评分向量内积作为两者边权的小数部分。

TEXT格式输出：第一行为两个整数N, M，分别表示节点数（用户数）和边数。接下来M行，每行三个整数x, y, w,表示节点x，y之间有一条边权为w的边。

JSON格式输出：有两个一级键："nodes"和"links"。分别表示图中的点和边。"nodes"是一个点集数组，每个点包含两个键："name"表示用户的名字，"group"该用户所属联通分量的编号。"links"是一个边集数组，每条边包含三个键："source"和"target"表示一条边的两个端点，"weight"表示边权。

运行方式：：双击/bin/Network.exe，输入阈值，会在/bin/output目录下生成graph.txt和graph.json文件。

引用代码

- GUI部分的HTML和CSS采用了W3School的Apartment Rental Template。https://www.w3schools.com/w3css/tryw3css_templates_apartment_rental.htm
- 使用了nlohmann/json作为C++的json库。https://github.com/nlohmann/json
- 多线程池实现参考。https://github.com/progschj/ThreadPool

功能亮点

- 用多线程优化Floyd算法，以计算紧密中心度。
- 采用Brandes' algorithm计算介数中心度，降低时间复杂度。
- 用配对堆优化Prim和Dijkstra。
- electron和d3实现的可交互GUI。
- 4个图论算法均进行了实现。
- 以用户为结点建图的附加功能。