图分析大作业 报告文档

—— 软件 62 冯昊 2016013255

软件 61 卢北辰 2016013242

一、 主要内容描述

利用电影层面的电影名、影片分类统计数据和用户层面的用户名、评论电影名、评分统计数据，构建以电影为节点、共同观影用户为边的无向图。以图为基础，实现最短路径算法、最小生成树算法、图的连通分量算法，并据此分析图的性质。

此大作业已经放在了 GitHub Pages（ http://fengh16.github.io/graph ）上面，可以直接扫码进入（由于数据文件较大，可能打开较慢，请耐心等待）。由于 Chrome 浏览器安全检查较为严格，采用 tsv 文件（为了避免电影名用户名等一些特殊符号的干扰）而导致本地直接打开会报错（Firefox 浏览器打开没有问题，微信直接扫码可以打开）。

****Chrome 需要访问服务器网站，Firefox 以及 Edge 可以直接打开本地文件操作。****

二、 核心算法（模块）描述及实现方法

1、电影信息读取模块（必做）

主要功能：读取电影信息文件，并将其转换为一个电影节点集合。每个电影节点包括电影名、电影类型、电影评分、电影编号信息。

实现方法：逐行读取文件，之后逐行分析，划分每行的信息，并以此创建或更改节点。

2、用户信息读取模块（必做）

主要功能：读取用户信息文件，并将其转换为一个用户信息集合，同时完成邻接矩阵的建立。每个用户信息单元包括用户名和该用户看过的电影信息。

实现方法：逐行读取文件，之后逐行分析，划分每行的信息，并以此创建或更改节点。此模块实现在电影信息读取模块之后，因此可以以“被同一用户观看”为标准在图中建立边，从而完成图的构建。

3、最短路径算法（必做）

主要功能：接收 2 个电影节点的编号，输出这两个电影节点间的最短路径及其权值。

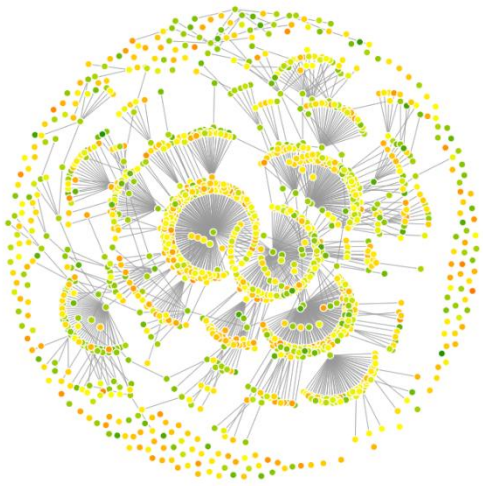
算法实现：Dijkstra 算法。以含起点的集合为初始已知集合，逐个向已知集合中添加距离起点最近的节点，直至所求终点节点被添加（则该条道路即为最短路），或没有

节点可以被添加（即这两点之间不连通，不存在最短路径）。

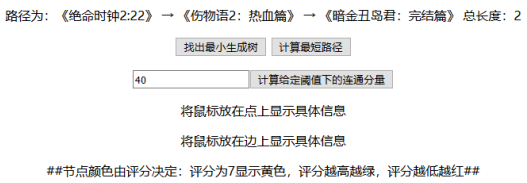
4、最小生成树算法（必做）

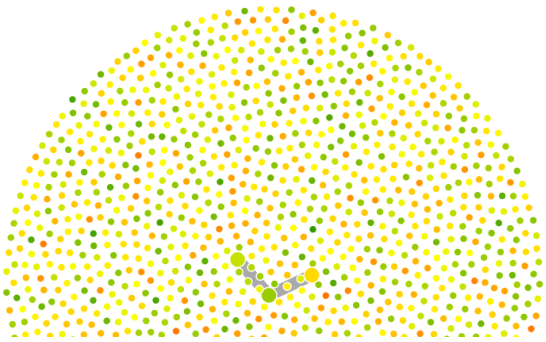
主要功能：在原图的基础上，求得原图的最小生成树。

算法实现：Prim 算法。以含起点的集合为初始已知集合，逐个向已知集合中添加距离已知集合最近的节点，直至所有节点都被添加。由于原图不一定连通，而只有连通图具有最小生成树，因此此算法在计算前将不存在的边假定为边权为一极大整数的临时边，由此得到最小生成树后，再将临时边消去，因此得到的其实是一个“最小生成森林”。



最小生成树



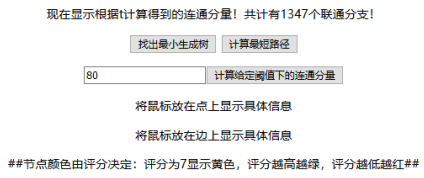


最短路径

5、连通分量算法（必做）

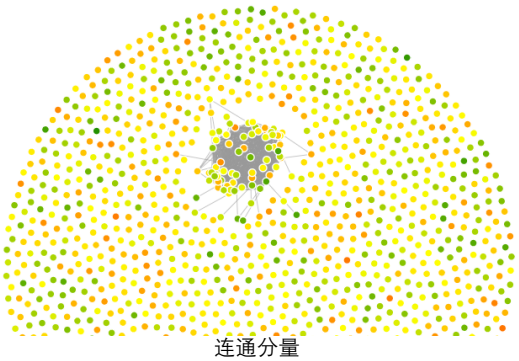
主要功能：①在原图的基础上，根据阈值确定边的有效性，将原图划分为若干个连通分支，并得到每个连通分支的点集。②在图形界面中，显示出给定阈值下的有效边形成的连通分支。

算法实现：深度优先搜索算法。选取一个尚未处于任何连通分支的节点，将其放入一个新的连通分支，再对该结点进行深度优先搜索，将属于该连通分支的所有未分配节点加入，之后再次选点，重复上述过程，直到图中所有节点都已被分配入连通分支。



6、图形界面与可视分析模块（选做）

主要功能：将图以可视化的形式展现在网页上。对图进行最短路径、最小生成树、连通分量计算后，可将运算结果以图形的形式表达，从而直观地展现出图和计算结果，方便分析图的性质，实现可视分析。(具体各算法的实现结果已在对应算法介绍之后给出)。在图形之中，用颜色来直观显示电影的评分（绿色代表 10 分，黄色代表 7 分，分数较低时为红色，较高时为绿色），在鼠标移上去的时候可以显示出来对应节点或者边的信息。



连通分量

实现方法：利用 D3（基于数据的文档操作 javascript 库），把数据和 HTML、SVG 等结合起来，从而创造出可交互的数据图表。响应鼠标移上去和移出去的事件，对对应的节点、边进行大小、宽度的修改。（移上去的时候，节点以及直接相连的边和点会放大显示，同时修改文本框的内容。如果在最短路径显示过程中，由于路径少，所以就加大、加宽了对应的节点和边）

7、网页数据爬取（选做）

主要功能：使用 py 里的 requests 库以及 BeautifulSoup 库进行网页数据（豆瓣）的爬取。主要获取数据包括：电影名称、电影评分、用户名称、用户对应的评分数据。

实现方法 :使用自己的账号模拟登陆，并且间隔一段时间进行刷新操作（同时采用 token 模拟的方法使得服务器误认为自己是正常登陆）。由于反爬虫限制，我试图采用代理服务器的方法进行爬取（虽然因为没有好的代理服务器源，并且很多代理服务器已经被豆瓣封掉了，效果并不是很好）。在我提供的源码中，有前后几版不同的源码，包括爬取长评论版、爬去短评论版、代理服务器伪装 IP 版，以及最后处理数据用到的 DealWithText.py（从网页中摘取评论数据）和 CreateFile.py（从爬下来的数据获取同助教提供的两个 csv 文件以及我最后使用的两个 out.tsv 和 value.tsv）两个文件。

三、 遇到的问题及解决方法

1、最早的数据读取是将数据转化为节点或信息实例后，将其存放在一个向量（vector）中，但实际运行过程中发现，由于 vector 类中元素的查找是通过顺序查找进行，因此算法时间复杂度相当高，需要大量时间进行计算。后来，将容器替换为集合（set），由于节点和信息都具有唯一性，恰好满足集合的使用条件，而集合自动将信息储存为一棵红黑树，查找效率很高，从而解决了问题。

2、在 d3.js 使用过程中，一直使用 Firefox 进行调试，后来用 Chrome 打开的时候发现其安全级别过高导致无法用 d3 自带的数据读取函数进行读取。因此将程序搬运到了服务器上，使得 Chrome 能够顺利打开。

3、在 d3.js 运行过程中，发现自己写的未封装的 force 图无法直接添加边（会报错：某些节点坐标无效），因此采用每次循环重新构建 svg 的方法进行操作，解决了该问题。

4、在爬取数据过程中，由于反爬虫限制，而且短评数目过多，所以转而采取爬取短评的方法。由于豆瓣服务器经常会强迫关闭连接，或者返回空数据，因此需要有断点续传的功能。在我的爬虫程序中，采用了以标记确定爬取开始位置的方法，借此来实现较好的爬取效率。同时，由于宿舍断电（写爬虫的时候是 10 月，当时还要熄灯）所以写了两个版本的（Windows 版和服务器（Ubuntu）版）。

5、由于 Chrome 坑爹的安全设置，无法直接通过 d3 的读取文件方式读取本地存储的 tsv 文件（因为 tsv 文件采用的 tab 分隔方式不会影响数据，电影名和用户名中可能会有引号逗号等影响因素，所以采用 tsv 文件），所以将文件放在服务器上，让 Chrome 浏览器能够正常读取（Firefox 和 Edge 读取没有问题）。

附件

1、网页网址：http://fengh16.github.io/graph

2、代码文件（包含程序的所有源代码）、网页文件、数据文件：见压缩包中。