算法课期末大作业

一、实验简介

本实验涉及如何实现一个图的库，包括图的存储、读写、图结构挖掘算法的实现以及图的可视化。我们希望在实现对应功能时能使用便捷的接口化形式，例如你使用python实现该作业，我们希望你的示例程序如下列格式：

```
# 以下为一个示例，具体函数名以及结构可与下方不同
# 读入文件
g = Graph("输入文件")
g.save("输出路径")

# 实现图结构挖掘算法
g.k_cores("输出结果1")
g.ds("输出结果2")
# .. 以及其他你实现的算法

# 可视化
g.show() # 展示图
g.show_coreness() # 展示coreness结构
# .. 其他你实现的可视化样例
```

二、实验内容

1. 图的读写（20分）

```
# 图格式如下
n   m #  表示点数和边数
u   v # 表示一条u到v的连边
...
```

- 学生需要实现一个图的存储结构，支持图的创建、节点和边的添加与删除。
  - c++图的实现可以使用SNAP库https://snap.stanford.edu/index.html，也可以自己实现
    - 此外也可参考部分论文中对图结构的设置
      - 1）https://github.com/LijunChang/Cohesive_subgraph_book
      - 2）https://github.com/Xiejiadong/Quantifying-Node-Importance-over-Network-Structural-Stability
  - python可以考虑使用NetworkX，igraph, Snap库等，也可以自己实现
  - 注：
    - 自己实现图结构容易拓展到不同算法的相应数据结构，但需花费额外时间。
    - 使用库会增加上手难度，大家可以酌情选择，无论是否自己实现均不会影响最终成绩
- 实现图的读取功能，能够从文件（如：.txt等格式）中读取图的节点和边信息。
  - 为了方便处理，我们可以将所有的图都处理为无向简单图，因此你需要做下列步骤：
    - 图中可能含有重边和自环，你需要对应去除
    - 图中的无向边(u, v) 可能表示为 (u, v) 和 (v, u) 的有向边形式，你应当对应处理

    - 图中顶点可能并不是$1-n$的全映射，即输入$n=5$时顶点的实际序号可能为$(0,1,4,6,9)$, 针对该情况，你需要合理处理，保证输入和输出的顶点能够一一对应
- **实现图的基础指标计算**
    - 如图的密度，图的平均度等
- 实现图的写入功能，能够将图的信息输出到文件中。

2. 图结构挖掘算法实现 （60分）

- 实现图的结构挖掘算法
    - 1）k-core分解（10分）：
        - 需要对读入的图能够计算每个顶点的coreness值，并可以用下面的格式存放到对应输出文件中

```
# output.txt
xxx.xxs # 运行时间
1   coreness[1]
2   coreness[2]
...
```

        - 计算过程可以参考YOJ 1127社交网络，也可以查看参考文献 [1]
    - 2）最密子图（15分）：
        - 精确算法：对输入的图求出最密密度，且输出最密子图对应的子图，如有多个最密子图，可以输出任意一个，输出形式可如下：

```
# output.txt
xxx.xxs # 运行时间
density  # 最密子图对应的密度
1 2 3 4 5 6 # 最密子图对应的子图
```

          可参考 YOJ 1128 管理公司 题解
        - 近似算法：用一个2-近似算法求出2-近似密度子图，即$\rho(S)\geq\frac{\rho(S^*)}{2}$, 所求子图的密度要大于等于最密子图的一半。输出形式可如下：

          实现方法可参考参考文献 [2] 中 4.2 节

```
# output.txt
xxx.xxs # 运行时间
density  # 2-近似密度子图对应的密度
1 2 3 4 5 6 # 2-近似密度子图对应的子图
```

    - 3）k-clique分解 (15分）：
        - 输入对应的k，使用BK算法求k-clique，并将所有极大团输出。输出形式可如下， 其中一行对应一个极大团：

```
# output.txt
xxx.xxs # 运行时间
1 2 3 4 5 6 # 极大团1
7 8 9 10 # 极大团2
```

        - 可参考 YOJ 1125 朋友 和 YOJ 1126 最大团的相应题解

- 5）**实现下列任一即可获得20分**
  - **以下算法论文中大多有参考代码，可对应参考具体实现**
  - LDS（局部密集子图）

    ——使用 [3] 中基于网络流和使用 [4] 中基于凸优化的算法均可，我们目标是求top-k LDS，其中k是人工输入的参数，输出格式可如下，其中一行为一个LDS

    可参考https://github.com/chenhao-ma/LDScvx中实现

    ```
    # output.txt
    xxx.xxs # 运行时间
    density 1 2 3 4 # top-1 LDS的密度和对应顶点
    density 7 8 9 10 # top-2 LDS的密度和对应顶点
    ```

  - k-clique最密子图

    ——参考[4] 中具体实现（论文中任一实现均可），需要人工指定k，输出格式可如下：

    可参考https://github.com/btsun/kclistpp中实现

    ```
    # output.txt
    xxx.xxs # 运行时间
    density 1 2 3 4 # 最密的k-clique子图密度，以及其对应的顶点
    ```

  - k-core动态维护：

    ——在实现k-core的基础上，支持对图进行边的插入和边的删除，并能快速计算在边插入和删除后顶点k-core的变化

    可参考文献 [5] 和 [6]，实现任一即可

    每次输入一条边(u, v), 同时输入其为插入边或删除边，插入和删除后的输出结果和k-core相同

    ```
    # output.txt
    xxx.xxs # 运行时间
    1   coreness[1]
    2   coreness[2]
    ...
    ```

  - k-vcc分解：
    - k-vcc定义：1）移除图G中任意小于等于k-1个顶点，G都不会断开连接 2）该图是极大的，即不存在一个图G'是图G的超集，并且G'也满足性质1）
    - 可参考论文：

      Enumerating k-Vertex Connected Components in Large（ICDE 2019）（精确算法）、这篇论文提出了自上而下基于图划分的k-vcc查找方式（难度⭐⭐⭐）

      Towards k -vertex connected component discovery from large networks（WWW）（近似算法）、这篇论文提出了自下而上基于种子扩展的k-vcc查找方式（难度⭐⭐⭐）

    以上两种方式选择其一即可
    - 输入对应的k，使用最大流算法求k-vcc，并将所有k-vcc输出。输出形式如下，其中一行对应一个k-vcc：

```
  # output.txt
xxx.xxs # 运行时间
  1 2 3 4 5 6 # k-vcc1
7 8 9 10 # k-vcc2
```

3. 图可视化 （20分）

3.1 总体要求

- 实现一个简单的图可视化功能，能够将图以图形的方式展示在屏幕上。
- 可视化应支持节点和边的样式设置，如颜色、大小、标签等。
- 可视化可以考虑实现良好的交互性，如缩放、平移、布局调整等。
- 对各个图挖掘算法的结果也可以进行相应的可视化展示。

3.2 推荐图可视化库

c++可选（难度：⭐⭐⭐）

Graphviz，官网链接https://graphviz.org/；Graphviz（Graph Visualization Software）是一个由AT&T实验室启动的开源工具包，专门用于绘制图形的布局和渲染。它广泛应用于自动图形布局处理，支持多种类型的图形表示，包括有向图和无向图。

官方教程：https://www.graphviz.org/pdf/libguide.pdf

Library Usage：https://graphviz.org/docs/library/

官方库：https://gitlab.com/graphviz/graphviz

参考博客：https://blog.csdn.net/root_clive/article/details/122395524

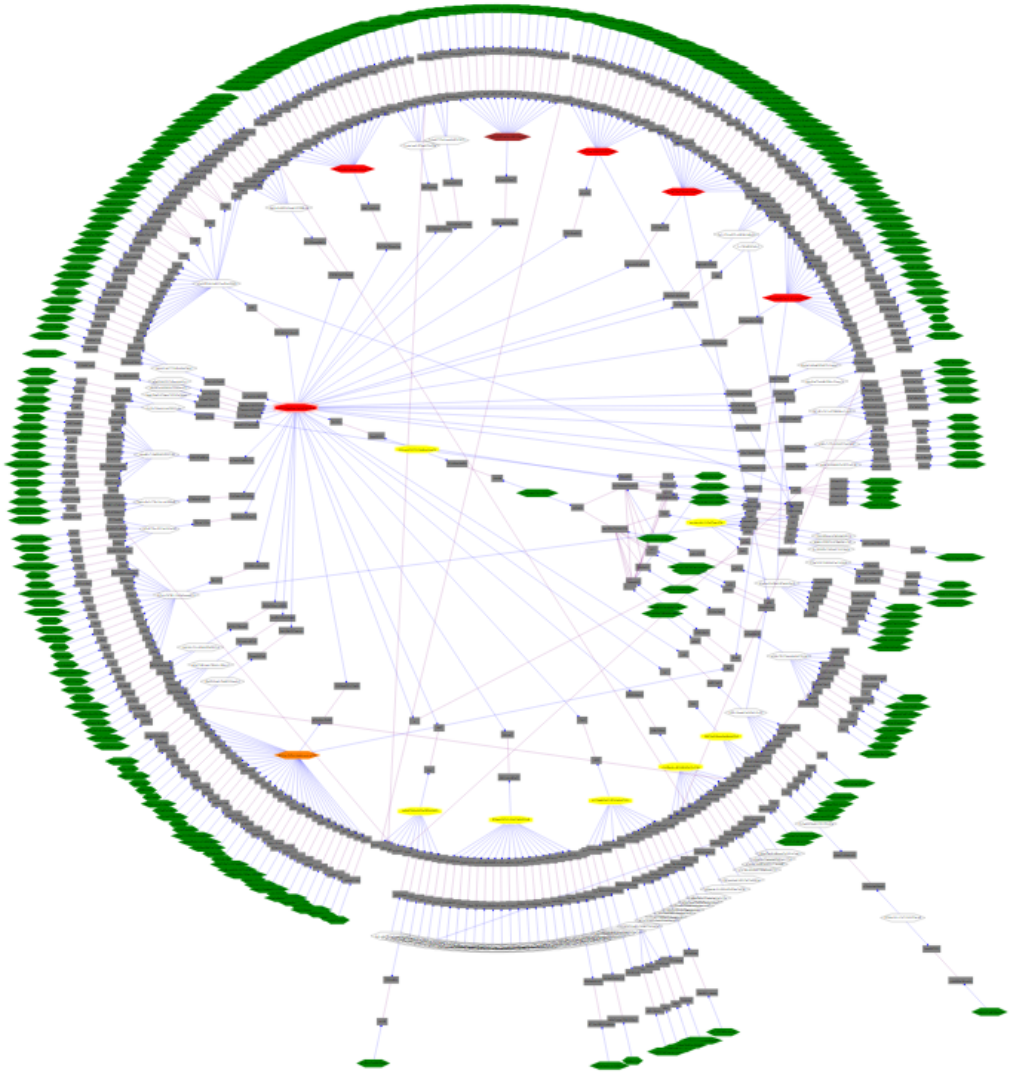
一些例子：

```
## Sample Programs using Graphviz

* [demo.c](https://www.graphviz.org/dot.demo/demo.c)
* [dot.c](https://www.graphviz.org/dot.demo/dot.c)
* [example.c](https://www.graphviz.org/dot.demo/example.c)
* [simple.c](https://www.graphviz.org/dot.demo/simple.c)
* [Makefile](https://www.graphviz.org/dot.demo/Makefile)
```

https://graphviz.org/Gallery/twopi/twopi2.html

效果展示：



python可选（难度：⭐⭐）

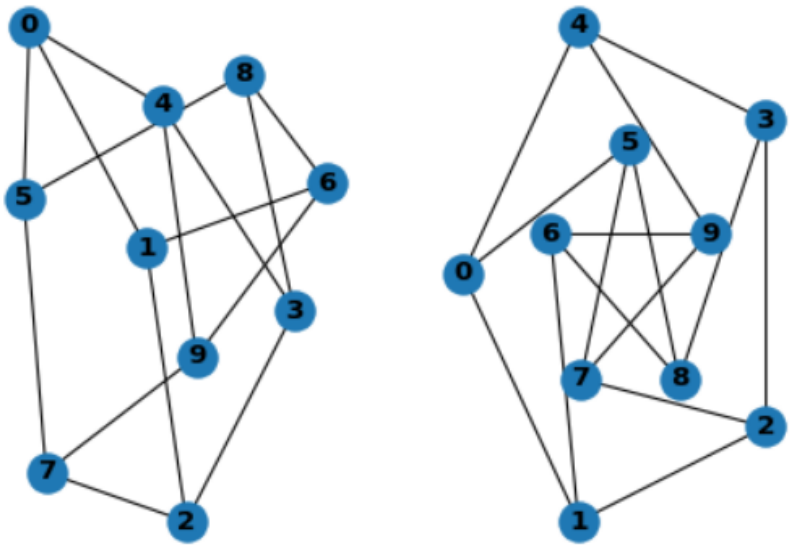
networkx，官方教程：https://www.osgeo.cn/networkx/tutorial.html#drawing-graphs

参考博客：https://zhuanlan.zhihu.com/p/381645334

https://zhuanlan.zhihu.com/p/36700425

https://www.cnblogs.com/luohenyueji/p/16991239.html

效果展示：



三、实验要求

- 使用C++或Python语言进行编程。
- 代码应具有良好的可读性，注释清晰。
- 需要提交完整的代码和实验报告，报告中应包含实验的设计思路、算法实现、测试结果和分析等内容。
- 鼓励进行创新，可以尝试实现额外的图算法或优化现有算法。

四、提交材料

- 完整的源代码文件。
- 实验报告，包括实验目的、实验内容、实验步骤、实验结果及分析等。
  - 可以贴部分代码并解释代码逻辑
- 运行结果文件：
  - 其中包含你所实现的图挖掘算法在下方三个数据集上的运行结果

五、提供材料

- **数据集**：社交网络，基于位置的网络等

  以下三个数据集均为未处理的版本，你需要参考第一部分中的描述对数据集加以处理

  - Gowalla.txt
  - Amazon.txt
  - CondMat.txt

dataset	nodes	edges	Average Clustering Coefficient	Diameter (Longest shortest path)
Gowalla	196591	950327	0.2367	14
Amazon	334863	925872	0.3967	44
CondMat	23133	93497	0.6334	14

参考文献

[1] Batagelj V, Zaversnik M. An o (m) algorithm for cores decomposition of networks[J]. arXiv preprint cs/0310049, 2003.

[2] Subgraph C. Cohesive Subgraph Computation over Large Sparse Graphs[J].

[3] Qin L, Li R H, Chang L, et al. Locally densest subgraph discovery[C]//Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2015: 965-974.

[4] Sun B, Danisch M, Chan T H H, et al. Kclist++: A simple algorithm for finding k-clique densest subgraphs in large graphs[J]. Proceedings of the VLDB Endowment (PVLDB), 2020.

[5] Zhang Y, Yu J X, Zhang Y, et al. A fast order-based approach for core maintenance[C]//2017 IEEE 33rd International Conference on Data Engineering (ICDE). IEEE, 2017: 337-348.

[6] Li R H, Yu J X, Mao R. Efficient core maintenance in large dynamic graphs[J]. IEEE transactions on knowledge and data engineering, 2013, 26(10): 2453-2465.