算法课期末大作业 - 图系统实现

简介

本项目旨在实现一个基本的图系统，包括图的存储、读写、图结构挖掘算法（k-core分解、最密子图、k-clique分解、k-clique最密子图）以及图的可视化功能。所有功能均使用 Python 实现。

环境配置

本项目依赖于以下 Python 库：

- `matplotlib`
- `networkx`
- `scipy`

您可以使用 `pip` 安装这些依赖：

```
pip install matplotlib networkx scipy
```

文件结构

- `graph_system.py`：包含 `Graph` 类及其所有图处理和算法实现。
- `dummy_graph.txt`：用于演示的小型示例图数据。
- `Amazon.txt`，`CondMat.txt`，`Gowalla.txt`：提供的实际数据集。
- `output/`：运行结果输出目录。

运行方式

主程序入口为 `graph_system.py`。您可以通过命令行运行它，并指定输入图文件和输出结果的前缀。

```
python graph_system.py <input_file> <output_prefix>
```

- `<input_file>`：待处理的图数据文件路径（例如：`Amazon.txt`，`CondMat.txt`，`Gowalla.txt`）。
- `<output_prefix>`：输出结果文件的前缀。所有算法的输出文件将以此前缀命名，并添加相应的后缀（例如：`_kcores.txt`，`_densest_approx.txt` 等）。

示例：

运行程序处理 `Amazon.txt` 数据集，并将结果输出到 `output/Amazon_` 开头的文件中：

```
python graph_system.py Amazon.txt output/Amazon
```

注意：

- 程序会自动创建 `output` 目录（如果不存在）。
- 对于大型图（如 `Amazon.txt` 和 `Gowalla.txt`），部分算法（尤其是 `k_clique` 和 `k_clique_densest_subgraph`）可能运行时间较长甚至超时。这是由于这些算法本身的计算复杂度较高。在实际测试中，您可能需要根据机器性能和数据集大小调整 `k_clique` 和 `k_clique_densest_subgraph` 方法中的 `time_limit` 参数，或者只在较小的数据集上运行这些算法。

实现功能

1. 图的读写与处理

- **`Graph` 类**: 负责图的存储（邻接表表示）、节点映射和基本操作。
- **`read_graph(file_path)`**: 从指定文件读取图数据。支持处理原始数据中的重边、自环，并将有向边转换为无向边。同时，对节点ID进行连续映射，方便内部处理。

2. 图基础指标计算

- **`_calculate_density(subgraph_nodes)`**: 计算给定子图的密度。

3. 图结构挖掘算法

- **`k_cores(output_path)`**: 实现 k-core 分解算法，计算图中每个节点的 coreness 值。输出文件包含运行时间以及每个节点的原始ID和对应的 coreness 值。
- **`densest_subgraph_approx(output_path)`**: 实现最密子图的 2-近似算法（Charikar's greedy algorithm）。输出文件包含运行时间、近似最密子图的密度以及子图中的节点原始ID。
- **`densest_subgraph_exact(output_path)`**: 精确最密子图算法的占位符。目前直接调用了近似算法的结果，因为精确算法通常需要复杂的最大流最小割实现，超出了本次作业的范围。
- **`k_clique(k, output_path, max_cliques=10000, time_limit=60)`**: 使用 Bron-Kerbosch 算法查找图中的所有极大团。**注意：** 此算法在大型图上可能非常耗时。为了避免无限运行，增加了 `max_cliques` 和 `time_limit` 参数来限制输出的极大团数量和运行时间。输出文件包含运行时间以及找到的每个极大团的节点原始ID。
- **`k_clique_densest_subgraph(k, output_path, time_limit=60)`**: 查找 k-clique 最密子图。此实现采用了一种启发式方法，通过迭代检查潜在的 k-clique 并计算其密度来寻找最密子图。**注意：** 对于大型图，此方法可能仍然较慢，且不保证找到全局最优解。输出文件包含运行时间、最密 k-clique 子图的密度以及子图中的节点原始ID。

4. 图可视化

- **`show()`**: 使用 `matplotlib` 和 `networkx` 库对图进行可视化，并将结果保存为 `graph_visualization.png` 文件。对于大型图，可视化可能需要较长时间或生成较大的图片文件。

实验结果与分析

由于大型数据集（如 `Amazon.txt` 和 `Gowalla.txt`）上 `k_clique` 和 `k_clique_densest_subgraph` 算法的计算复杂度较高，在有限的沙盒环境中可能无法在合理时间内完成。因此，建议在测试这些特定算法时使用较小的数据集或调整 `time_limit` 参数。

运行结果文件示例 (dummy_graph.txt):

运行命令：`python graph_system.py dummy_graph.txt output/dummy`

- `output/dummy_kcores.txt`：

```
 0.00s
1 2
2 2
3 2
4 2
5 1
```

- `output/dummy_densest_approx.txt`：

```
 0.00s
1.5
1 2 3 4
```

- `output/dummy_densest_exact.txt`：

```
 0.00s
1.5
1 2 3 4
```

- `output/dummy_maximal_cliques.txt`：

```
 0.00s
1 2 3
1 2 4
```

- `output/dummy_kclique_densest.txt` (k=3):

```
 0.00s
1.5
1 2 3 4
```

注意： 实际运行时间可能因机器性能而异。

总结

本项目实现了一个功能较为完整的图系统，涵盖了图的读写、多种图结构挖掘算法以及可视化。在处理大型图时，部分算法的性能是一个挑战，这反映了图算法本身的复杂性。未来的

工作可以专注于进一步优化这些算法，或探索更适合大规模图处理的分布式算法。