AI607: Graph Mining and Social Network Analysis (Fall 2020)

Homework 1: The Kronecker Graph Generation

Release: September 11, 2020,
Due: September 25, 2020, 11:59pm

1 Introduction

This assignment will help you understand the principle of the Kronecker model. The Kronecker model is a model for generating graphs that obey patterns observed in real-world networks. In this assignment, you will implement the Kronecker model and analyze its results using different seed graphs.

2 Implementation

We will use Numpy [1] and Scipy [2] for this assignment. Given a Kronecker initiator P_1 and an exponent k, you should compute P_k, the k-th Kronecker power of the initiator. The Kronecker initiator P_1 is the $N \times N$ adjacency matrix of a graph with N nodes, and it is given in the input file in the following format:

```
        1    1    0
        1    1    1
        0    1    1
```

Note that the size N can vary. In this example, the Kronecker initiator is an adjacency matrix of 3 nodes. (See the attached example input files.) The computed P_k becomes the adjacency matrix of the generated Kronecker graph.

2.1 Kronecker product computation

First, load the initial matrix in the intializer of the class `Kprods`. Then, implement the `produceGraph` function in `Kronecker.py` to compute the k_{th} Kronecker power of the initiator matrix. You may use a recursive function to efficiently compute the kronecker product. The computed result (P_k) should be returned by the `produceGraph` function.

[1]`https://numpy.org/install/`
[2]`https://scipy.org/install.html`

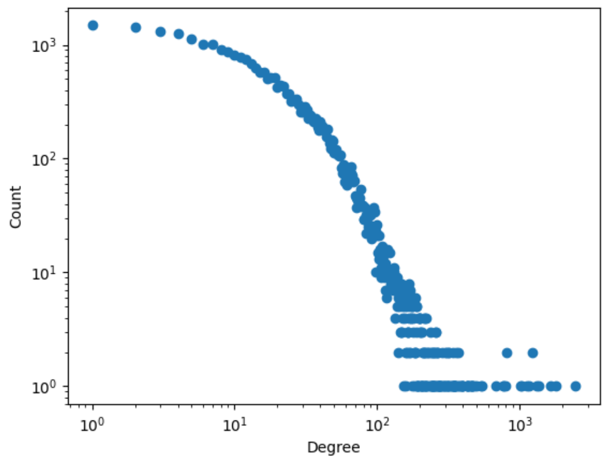

Figure 1: Degree distribution example

2.2 Analysis

Visualize the degree distribution of the generated Kronecker graph. We provided the two Kronecker initiators with exponents:

```
        Case 1 (k = 8):
             1    1    1    1
             1    1    0    0
             1    0    1    0
             1    0    0    1

        Case 2 (k = 6):
             1    1    0    0    0
             1    1    1    1    0
             0    1    1    0    0
             0    1    0    1    1
             0    0    0    1    1
```

Implement `computeDegDist` function in `Analysis.py` to compute the degree distribution. Then, visualize the distribution using `plotDegDist` (provided) or any plotting method. Any plotting methods are allowed as long as the distribution is similar enough.

Figure 1 shows the degree distribution of a citation graph consisting of some High-Energy Physics Theory research papers from ArXiv. If you have implemented everything properly, the degree distribution of the generated Kronecker graph in **Case 1** should be similar to that given in the figure.

- Visualize and compare the degree distributions of **Case 1** and **Case 2** provided above.
- The x-axis and y-axis should represent the node degrees and the number of nodes corresponding to the degree, respectively.

- Both axis should be log-scaled.
- Provide the figures in your report (**report.pdf**).

3 Test

You can test your implementation by executing the `main.py` with some options:

```
usage: main.py [-h] [-f FILENAME] [-k K] [-o OUTPUTNAME]

Generating Kronecker graphs.

optional arguments:
  -f FILENAME, --fileName FILENAME
                        Select the input file: ex. inputA.txt
  -k EXPONENT, --k EXPONENT
                        Select the exponent of the Kronecker product
  -o OUTPUTNAME, --outputName OUTPUTNAME
                        A file name for an output matrix
```

For example, you can run `main.py` using the provided example as:

```
    python main.py -f inputA.txt -k 8 -o inputA.png
```

4 Notes

- Your implementation should run on TA's desktop within **10 minutes**.
- You may encounter some subtleties when it comes to implementation. Come up with your own design and/or contact Taehyung Kwon (taehyung.kwon@kaist.ac.kr) and Geon Lee (geonlee0325@kaist.ac.kr) for discussion. Any ideas can be taken into consideration when grading if they are written in the *readme* file.
- Do not modify the code outside of the "TODO" regions. For example, do not import additional python libraries outside "TODO" regions.

5 How to submit your assignment

1. Create hw1-[your student id].tar.gz, which should contain the following files:
   - **main.py, Kronecker.py, Analysis.py**: these should contain your implementation.
   - **report.pdf**: this should contain your answers in Section 2.2 (Analysis).
   - **readme.txt**: this file should contain the names of any individuals from whom you received help, and the nature of the help that you received. That includes help from friends, classmates, lab TAs, course staff members, etc. In this file, you are also welcome to write any comments that can help us grade your assignment better, your evaluation of this assignment, and your ideas.

2. All 5 files should be included in a **single folder**.
3. Make sure that no other files are included in the tar.gz file.
4. Submit the tar.gz file at KLMS (`http://klms.kaist.ac.kr`).