The code contains a lot of data processing part, which is a little bit tedious due to time-issue. I will briefly explain the general workflow and talk about something that I haven’t done yet.

General logic of the data process:

**IMPORTANT**: If your graph data only has endpoints of an edge but doesn’t have weights, remember to add “-f simple” to the command of running the searching algorithm, otherwise it wouldn’t give you the correct results.

**General logic:**

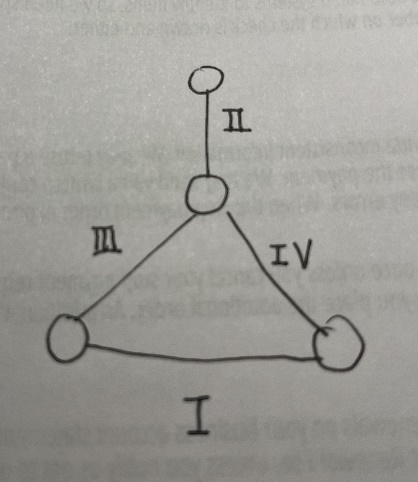
Change graph data from separated by comma into separated by \t

->Run the searching algorithm

->filter the results so that only wanted subgraph is kept (in this case, the 0001001101011110

one)

->Finding the edge IDs of each edge in subgraph lists

->Sort within each subgraph so that it fulfills certain order (in this case, it’s the order below)

->According to flows assign to each graph, search and generate flow data for each graph (20 samples)

->Run bootstrap to get CIs for each sample

->Compare the CIs to test the hypothesis

**Things that I haven’t included**:

1. Tests for other graphlets. Since the chosen graphlets of size 4 is easy to implement, efficient in running time, and meet all requirements, I just use this graphlet for a demo. I haven’t tried other yet, the process would be more or less the same, but is expected to cost more time.
2. Tests for other comparison methods. I only do the full-comparison one.
3. CIs concerning edges with same topological environment inside subgraph. E.g. In the graphlet I am implementing right now, it should be edge pair (II-III), (II-IV). We should group them together as the same CI to compare.
4. I only ran 1 part of the Erdos graph settings, however, for all other parts, the process should be the same since they share exactly same graph structures. All you need is space and time.😊 You will probably run this 5\*60 times and colab doesn’t support me to do so. And by the way, if you would like to run all data from Erdos family (or other families), I suggest you optimize the bootstrapping part a little bit since that is the bottleneck of the running time.
5. Also, for the off-diagonal entries, I only check whether they are overlapping with each other or not, I haven’t check whether rho belongs to [0, 0.5], since it requires an assumption or a range for sigma^2, but I am 100% sure about whether using the overlap of all sigma^2 Cis is a good choice of range for substituting or not. But this part should be fairly easy to add.