Suggestions from Fleming

First off, to clarify the assignment, you need to take the name of the graph definition file as a command line parameter. Don't use prompts and ask the user to type in the file name. This is very important. We're going to test these programmatically, so if you do something like "Please enter the file name" your program will fail all test cases and you'll get a zero. It should be run as "myProgram nodeFile.txt". I'm attaching and example nodefile.txt to this message. Note that there are tabs between the node # and the weight, though most libraries will take care of this for you pretty easily, since a tab is whitespace.

There are many ways to do project 2 part 2, but I thought I'd share some thoughts on how to get started. In part 2, you need to spawn separate processes and exchange information between them. To do this, I'd suggest the following approach:

1. In Python, you can spawn a new process with the subprocess library https://docs.python.org/3/library/subprocess.html#module-subprocess

使用多进程，而不是多线程

使用 subprocess 库，可以使用 multiprocessing 库吗？

2. One of the things you can do with this library is pass in command line arguments to the programs you run.

？？？

3. You need to establish conventions on who starts processes, how processes choose listening ports, and who connects to who. For example you have 3 nodes in your graph. Who starts the three processes? What 3 ports will they use? And for a connection, will node 1 connect to node 2 or node two connect to node 1? Connections are bidirectional, so you shouldn't make two connections.

新建多个进程之后，每个进程使用 Socket 去监听不同的端口，通过端口发送信息（本机）

需要规定从哪个 Node 开始新建进程，谁连接谁，无向图，所以连接只需要一次

4. My suggestion is that you make the syntax for starting a command "myProgram <nodeFile name> <nodeNumber>", with the last parameter being optional. Then, when the program starts, check the number of parameters. If there is no node number, then this is node 1. Then have node 1 read the

nodeFile and start the other nodes, but passing in the node number to the other nodes. For example you would start the second node as "myProgram nodeFile.txt 2".

(file_name, node_number)作为函数输入，去启动这个程序，可以默认从 Node1 开始启动，读取 Node1 本身的数据，新建其他进程

5. Establish a convention for what ports nodes listen on. All the processes run on the same machine so an easy one would be something like "Node N's listening port is 1000+N". This means node 1 listens on 1001, node 2 listens on 1002, etc etc.

为每个进程设置独一无二的监听端口

6. Establish a convention for who connects to who. The simplest is the smaller number in a pair connects to the larger number node. For example, is there is a connection from node 2 to node 3, then node 2 initiates a TCP connection to node 3. This follows the file format, and makes things simpler. A node can simply read its part of the file and connect to whatever nodes are in its section.

进程之间需要通信，需要规定连接由谁发起，比如用 node 数字小的发起 TCP 连接

7. Realize that you may need to wait a bit to connect until everything is started up. For example if you start node 2 and immediately try to connect to node 3, node 3 may not be running yet. You either need to handle this failure and retry, or just sleep a couple of seconds and wait for everything to start up. Remember to start your listened before you sleep or it's pointless...

需要加入异常处理机制，如果连不上怎么办（sleep 几秒，再连一次）

死锁咋办

8. When you connect to a node, it won't know what your node number is. Probably the first thing you should send is some fixed length string or integer and tell the node who you are. For example, node 2 connects to node 3, and you send "2" as the first message so node 3 knows this connection is from node 2. If you make this fixed length, it's easier. For example 4 bytes.

线程间通信（通过 socket）