CX 4010 / CSE 6010
Assignment 6
Epidemic Network Model
Extra requirements for graduate students highlighted in green

Due Date: 11:59pm on Thursday, October 29
Submit a single zipfile as described herein to Canvas

In this assignment, you will work in small groups to develop a program to model the spread of an epidemic on a network (graph). There will be two main tasks: generating the network and performing the modeling using that network. Once the full program is developed, you should perform several experiments, output information, and perform some analysis as described below.

Network generation

Generate a nearest-neighbor network that is a generalization of a basic ring network in which each node connects to its two neighbors (indicated here by indices). In the case of two neighbors, you would connect each node j to its neighbors (j+1) and (j-1), with suitable "wrapping around" at the edges (e.g., node 0 would connect to node 1 and to the last node). For a general number of neighbors, progressively expand out from the neighbors whose indices are 1 away. For example, if the number of nearest neighbors is 6, then you should connect each node j to its neighbors j±1, j±2, and j±3.

The number of nearest neighbors `numNeighbors` should be treated as a parameter specified in a specified in the relevant .h file. This parameter should be even; if an odd number is specified, you should increase it by 1 and print a warning to the screen indicating this has been done. If the number of nearest neighbors is larger than the number of individuals in the population, the code should set the number of nearest neighbors to the population size and print a warning to the screen indicating that is the case.

Graduate students only: Convert your nearest-neighbor network to a small-world network. To do so, for each edge that would be added in constructing a nearest-neighbor network, modify the edge to connect to a randomly selected node instead according to a specified probability. For example, starting from a baseline network with two nearest neighbors and a probability `pReplaceRandom` for connecting to a random node of 0.5, a common behavior might be to connect node j to j+1 and, instead of to j-1, to a randomly selected node. The parameter `pReplaceRandom` should be specified in the relevant .h file. If `pReplaceRandom` is greater than 1, it should be set to 1, and if it is less than 0, it should be set to 0; in both cases the code should print a warning to the screen describing what has been done. Note that when `pReplaceRandom` is 0 you should recover the original nearest-neighbor network, so you should not submit a separate program for that case.

Epidemic modeling

For the epidemic, you will use an agent-based model. In such a model, individuals are modeled separately, with each individual labeled as being in either a susceptible, infected, or recovered state. The states are updated iteratively with each iteration corresponding to some fixed time period (e.g.,

one day). Each susceptible individual has a specified probability `pInfection` of becoming infected from contact with each infected individual to which they are connected via edges; if they are infected, their state must be changed from susceptible to infected. Once infected, individuals recover according to another specified probability `pRecovery` that is independent of neighbors; if they recover, their state must be changed from infected to recovered. If `pInfection` or `pRecovery` is greater than 1, it should be set to 1, and if it is less than 0, it should be set to 0; in both cases the code should print a warning to the screen describing what has been done.

When updating the states, be sure to make any changes take effect only in the next iteration, so that individuals updated later in a given iteration do not see changes to the states relative to what individuals updated earlier in the iteration see.

As part of the modeling effort, calculate the following three quantities and output them to the screen (with appropriate labeling):

- The maximum number of infected individuals in a single iteration.
- The iteration number at which the maximum number of infected individuals occurs.
- The sum of the number of currently infected and recovered individuals after the last iteration.

Graduate students only: Extend your model such that the network itself may be modified. When a node is switched from susceptible to infected, disconnect each of the edges connected to that node according to a specified probability `pDisconnect`. Remember to make these changes take effect in the next iteration, not the current iteration, and be sure to maintain symmetry (if edge (i,j) is removed, edge (j,i) also should be removed). If `pDisconnect` is greater than 1, it should be set to 1, and if it is less than 0, it should be set to 0; in both cases the code should print a warning to the screen describing what has been done. Note that when `pDisconnect` is 0 you should recover the case without network modification, so you should not submit a separate program for that case.

Experiments and analysis

Begin with a single infected individual (choose the first, so that your program will have an infected individual regardless of population size).

Undergraduate students:

Run your program 10 times and report the median, maximum, and minimum of each of the three calculated quantities listed above with your results. (You can do this offline or write your program to run 10 times and perform the calculations—your choice. If you have your program do these calculations, you should still output the relevant quantities from the 10 individual runs.)

Graduate students:

Run your program 10 times for each parameter combination below and report the median, maximum, and minimum of each of the three calculated quantities listed above with your results. (You can do this offline or write your program to run 10 times and perform the calculations—your choice. If you have your program do these calculations, you should still output the relevant quantities from the 10 individual runs.)

- `pReplaceRandom = 0,` `pDisconnect = 0`
- `pReplaceRandom = 0.25,` `pDisconnect = 0`
- `pReplaceRandom = 0,` `pDisconnect = 0.5`
- `pReplaceRandom = 0.25,` `pDisconnect = 0.5`

Suggested parameter values

Below are suggestions for the parameter values. If you feel the results you are getting are not interesting, you may choose a different value for one or more parameters—please make a note in that case.

Parameter	Meaning	Value
`numAgents`	Number of individuals (population size)	500
`numIterations`	Number of iterations (days)	100
`numNearestNeighbors`	Number of nearest neighbors connected to each node by edges in the nearest-neighbor network	10
`pInfection`	Probability of a susceptible individual becoming infected from each contact with a currently infected individual in a single iteration	0.05
`pRecovery`	Probability of a currently infected individual recovering in a single iteration	0.1
Graduate students only:		
`pReplaceRandom`	Probability of substituting a nearest-neighbor edge with a different edge (to a randomly selected individual)	0 and 0.25
`pDisconnect`	Disconnection probability	0 and 0.5

Division of labor

Most groups will have two members. One should write the code to generate the network and the other should write the code to perform the epidemic modeling on that network. You will need to coordinate to ensure that these two parts of the code work together properly. Work together to run the program and report the results.

You should submit the code as a single program with appropriate division into modular functions (including proper identification of who contributed what), rather than as two separate programs. If you are assigned to a group of three, two of you should develop separate implementations for one of the tasks (your choice of network generation or epidemic modeling) such that each can be combined with the work of the third group member. Overall, each team of three then will have two implementations to solve the problem, each with one portion developed by a single student and combined with the other portion developed by one of the other two students.

Group assignments will be available in Canvas.

Submission information

You should submit to Canvas **a single zipfile** that is named according to the Georgia Tech login of someone in your group—the part that precedes @gatech.edu in your GT email address. To receive full credit, your code must be well structured and documented so that it is easy to understand. Be sure to include comments that explain your code statements and structure.

The zipfile should include the following files:

(1) your code (all .c and .h files), with the main code that handles arguments, function calls, error handling, etc. named **main.c** and any helper functions in files named **network.h/network.c** and **model.h/model.c**. If you are using linux or Mac OS, we recommend you use a **makefile** to compile and run your program, and you should include it if so.

(2) a **README** text file (not formatted in a word processor, for example) that includes the compiler and operating system you used for compiling and running your code along with instructions on how to compile and run your program. If you are in a group of three, you should include instructions for both complete implementations in the same README.

(3) a series of slides composed in PowerPoint or similar software, saved either in PowerPoint or as a PDF and named **slides.pptx** or **slides.pdf**, in the following order:

- 1 slide: your names and a brief explanation of how you developed/structured your program. This should not be a recitation of material included in this assignment document but should focus on the main structural and functional elements of your program (e.g., the purpose of any loops you used, the purpose of any structs you may have used, the purpose of any functions you created, etc.).
- 1 slide: some sanity checks or other evidence in support of the correct operation of your code. (There is not a specific requirement beyond an attempt to address this in some way.)
- 1 slide, undergraduate students: report the results of your experiments and briefly comment on their meaning.
- 1 slide, graduate students: report the results of your experiments and briefly comment on their meaning, including the roles of the parameters `pReplaceRandom` and `pDisconnect`.
- 1 slide: a description of the division of labor across the team for this assignment (who did what) with specific references to the code.