KIT205 Data Structures and Algorithms
Assignment 3

Due: Friday 29th May, 11:55pm

Introduction

Epidemiology is the study of the spread of disease within a population. Epidemiologists often use a computer model to predict and mitigate the spread of disease. For example, it might be desirable to know the most effective strategy for selective vaccination when vaccine availability is limited. For this assignment, you will model the spread of a disease and the effect of selective interventions (e.g. vaccination, or isolation). You will also investigate the consequences of different type of social networks.

If the topic causes you any anxiety that may affect your ability to complete the assignment, please contact your unit coordinator as soon as possible.

Assignment Specification

The model

Every individual will be modelled as a vertex in small world network using an undirected graph. Frequent contacts of each individual will be modelled as edges connecting to other vertices.

Each individual will be in one of three groups: susceptible, infected, or recovered (this is known as the SIR model). Susceptible individuals will have a chance of moving to the infected group if one or more of their neighbours are in the infected group. Infected individuals will have a chance of moving to the recovered group. Recovered individuals cannot infect anyone or become reinfected. We will also use the “recovered” group to model immunised individuals.

The spread of the infection will be modelled over a number of time-steps. At each timestep:

- For each infected individual
    - For each susceptible neighbour there will be a probability, inf_rate, that the neighbour will become infected
    - There will also be a probability, rec_rate, that the individual will recover

Implementation

The graph of individual connections will be implemented as an adjacency list using our standard data structures:

```c
typedef struct edge{
    int to_vertex;
    float weight;
} Edge;

typedef struct edgeNode{
    Edge edge;
    struct edgeNode *next;
} *EdgeNodePtr;

typedef struct edgeList{
    EdgeNodePtr head;
} EdgeList;

typedef struct graph {
    int V;
    EdgeList *edges;
} Graph;
```

Note that the weight variable will not be used for this assignment, and that undirected edges will be represented as directed edges in both directions.

In addition to the graph data structure we will also have an array of vertex states, and parameters for infection rate and recovery rate.

```c
int* state; // 0 = susceptible, 1 = infected, 2 = recovered
```

Part A Constructing the Graph and Plotting Spread

Base Code

You have been provided with some base code. The main function:

- creates a small world graph using the Watts & Strogatz method
- runs 50 steps of the simulation, plotting the results
- calls a vaccinate function
- resets the simulation and runs another 50 steps

Scale Free Network

For this part of the assignment you need to replace the small world network with a scale free network using the method by Barabási & Albert.

The function prototype is provided in graph.h, but the function currently does nothing.

You do not have to complete this part first, since you can use the small world network to test the other parts of the assignment.

The Simulation

You need to write a function to advance the simulation one time step. The function prototype is provided in main.h

```c
void step(Graph G, int* state, float inf_rate, float rec_rate);
```

This function needs to calculate the new (random) infections and recoveries. For example, if individual u is infected, the they will recover (move to state 2) if:

```
 ((float)rand()/RAND_MAX < rec_rate)
```

Note that `(float)rand()/RAND_MAX` will produce a random number in the range 0..1. The cast to float is important otherwise you will get integer division and the answer will always be 0. `RAND_MAX` is a constant defining the maximum number returned by `rand()`.

Part B Flattening the Curve

Next you want to try and flatten the curve. You may choose to vaccinate (or isolate) a certain number of individuals. You could do this randomly, but to get full marks you need to choose some sort of rationale (based on connections) for choosing which individuals to vaccinate.

You should create a new function with the following prototype:

```
void vaccinate(Graph G, int* state, int num);
```

This function should (only) change the state of some individuals to the recovered (int value 2) state.

The function is called after the baseline simulation. Then the simulation is reset and rerun. If you have correctly implemented a good strategy, you should see a noticeable flattening of the curve (you will probably get a greater effect for the scale free network than the small world network).

Assignment Submission

Assignments will be submitted via MyLO (an Assignment 3 dropbox will be created). You should use the following procedure to prepare your submission:

- Make sure that your project has been thoroughly tested using the School's lab computers
- Choose "Clean Solution" from the "Build" menu in Visual Studio. **This step is very important** as it ensures that the version that the marker runs will be the same as the version that you believe the marker is running.
- Quit Visual Studio and zip your entire project folder along with a completed assignment cover sheet
- Upload a copy of the zip file to the MyLO dropbox

History tells us that mistakes frequently happen when following this process, so you should then:

- Unzip the folder to a new location
- Open the project and confirm that it still compiles and runs as expected
    - If not, repeat the process from the start

KIT205 Data Structures and Algorithms: Assignment 3

Synopsis of the task and its context

This is an individual assignment making up 15% of the overall unit assessment. The assessment criteria for this task are:

1. Implement generic graph data structures and algorithms
2. Apply graph data structures and algorithms to solve a specific problem

Match between learning outcomes and criteria for the task:

Unit learning outcomes	
On successful completion of this unit…	***Task criteria:***
On successful completion of this unit, you will be able to:	
1. Transform a real world problem into a simple abstract form that is suitable for computation	1-2
2. Implement common data structures and algorithms using a common programming language	1-2
3. Analyse the theoretical and practical run time and space complexity of computer code in order to select algorithms for specific tasks	-
4. Use common algorithm design strategies to develop new algorithms when there are no pre-existing solutions	2
5. Create diagrams to reason and communicate about data structures and algorithms	1-2

Criteria	HD (High Distinction)	DN (Distinction)	CR (Credit)	PP (Pass)	NN (Fail)
	You have:	**You have:**	**You have:**	**You have:**	**You have:**
1. Implement and apply graph data structures ***Weighting 60%***	• Scale free graph constructed correctly • Step function implemented correctly • Correct initialisation, usage, and termination (all memory freed)	• Scale free graph constructed correctly *-and-* Step function implemented correctly	• Scale free graph constructed correctly *-or-* Step function implemented correctly	• Partially correct graph construction and/or step function	• Genuine attempt to implement construction or step functions
2. Develop an algorithm to solve a specific graph problem ***Weighting 40%***	• Correctly implemented a vaccination function that utilises connection information to choose vaccination strategy	• Partially correct vaccination function that utilises connection information to choose vaccination strategy		• Implemented a vaccination function that is not based on connectivity	• Genuine attempt to implement vaccination function