Project: Simulating Covid 19

To do this project, we need to learn some prerequisites first :-)

Priority Queues

This project involves you to do some "extracurricular" reading about priority queues. A priority queue is like a regular queue that we discussed in the class, except that there is a "priority" associated with each element. There is also a notion of serving the elements in the queue, with higher priority elements being served first. There are several ways of implementing this ADT–I will *not* worry about how you implement it. A naive implementation using linked lists is also fine. Often, a data structure called *the heap* is used. I strongly urge you to read this as well.

You can find tons of resources, some of which are listed below:

1. `https://en.wikipedia.org/wiki/Priority_queue`
2. `http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html`
3. `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/MIT6_006F11_lec04.pdf`
4. `https://www.youtube.com/watch?v=ImSRt7LxQnY` (NPTEL)

   In particular, and relevant to us, priority queues are useful in implementing "discrete event simulations"; please see

   (a) `http://www.cis.umassd.edu/~ivalova/Fall07/cis360/rDemos/PQueues/PQueue.html`

   (b) `https://link.springer.com/content/pdf/bbm%3A978-3-319-50806-1%2F1.pdf` (See Section A.1.2; you will be implementing a simpler version of this)

   (c) `https://www.ias.ac.in/public/Volumes/sadh/022/05/0611-0627.pdf` (further reading if you have interest)

   We will use this application of the priority queue to study the epidemic.

Graphs and Adjacency Matrices

The next thing that you would need to learn to do this project is a *very basic idea* of *graphs* or *networks*. A network is a collection of nodes or vertices, which are connected by *edges*. Therefore, if we want to model individuals in a population and their contacts, one way to do is to model a person using a node and if two persons are in close contact with each other, we will model that by placing an edge between them. If they are not, then there is no edge between them. Obviously, if two nodes are connected by an edge, they are said to be *adjacent* to

each other. Such a structure of nodes and edges is called an (undirected) graph. Never mind about the undirected part for this project. One way of representing a graph in a computer is using an *adjacency* matrix. This can be implementing using linked lists.

- `https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)`
- `https://en.wikipedia.org/wiki/Adjacency_matrix` (see in particular the Undirected Graph example)
- ... and tons of others

Some Computational Epidemiology

An epidemic like Covid-19 is an SIR epidemic. An individual is initially *Susceptible* (S). Then s/he *might* get *Infected* (I) and finally Recover (R). However, as one can expect, each transition is probabilistic.

You have to implement a simple variant of the algorithm in section of A.1.2 of:

`https://link.springer.com/content/pdf/bbm%3A978-3-319-50806-1%2F1.pdf`

and obtain the infection curves[1].

Reproduced here is a part of the algorithm, with some modifications.

Algorithm 1 Part of Fast SIR

```
while Q is not empty do do
  Event ← earliest remaining event in Q
  if Event.action is transmit then
    if Event.node.status is susceptible then
      process_trans_ SIR(G, Event.node, τ, γ) [See Below]
    else {Event is Recovery}
      process_rec_SIR(Event.node, Event.time) [See Below]
    end if
  end if
end while
```

An event struct in the priority queue has to necessarily have the following fields– a) the time when the event occurs, b) the type of event (transmit or recover) and c) which node it concerns. Other fields are as per your programming convenience. Also, maintain lists of S, I and R people.

process_trans_ SIR

If the event is a transmit event and the corresponding node v is susceptible, delete v from the S list and add it to the I list. Create events for

[1]The source code is available in python in this document. You are encouraged to read that code (learn python in the process if you haven't already.

- **The node v transmitting the infection to its neighbors, along with the time at which this happens.** You can use the following subroutine to determine these times. For each neighbor, toss a coin with probability τ for each day until you get a *heads*. For the day you get a heads, put a transmit event for that neighbor. (Think how you can (biased) toss coins in C).

- **v's recovery**. Toss a biased coin with γ probability of it showing *heads*. Keep tossing until you get a heads. If you get a heads on the ith trial, v will recover after i days. (So, we create an event for that and insert in the priority queue.)

process_rec_ SIR

If the event is a recover event, then remove that node from the I list and add it to the R list.

So what do we have to do?

Perform this simulation for a maximum of 300 days, use τ as 0.5 and γ as 0.2. Construct a graph of 10000 nodes using: `https://www.sanfoundry.com/c-program-generate-random-undirected-graph-given-number-edges/`, with MAX_VERTICES as 10000 and MAX_EDGES as 3000.