GitHub Repository Link: https://github.com/alkray/Probability-Statistics-for-CSC-Final-Project

From what I understand, a Monte Carlo simulation takes a scenario and by inputting random numbers and doing a lot of trials, somewhat accurate results could be produced from otherwise impossible calculations. This would be a big task with multiple moving parts so I chose an object oriented programming language, Python. I needed to break up this project into pieces.

First would be to simulate one trial. This piece was threefold. I had to initialize the network with one random computer infected with all the variables of the simulation. I decided using a 2d array would be best for my case. I used the top row to signal if a computer was infected and the second to count how many times it had been infected. Next I coded a day and its events. In a day there are two events. The virus spreading and the IT cleaning. For the virus spreading I used a method that took in the probability of spreading and the number of computers infected. This is important because the number of computers infected changes the probability of a computer getting infected.

$$P(A \cup B) = 1 - P(notA) \times P(notB)$$

Above is the formula that is used to find the probability that at least one of two computers infects a uninfected computer. This formula can be altered to probability of at least one of n number of infected computers infects an uninfected computer given that the probability of one computer infecting an uninfected one is constant. The formula is below:

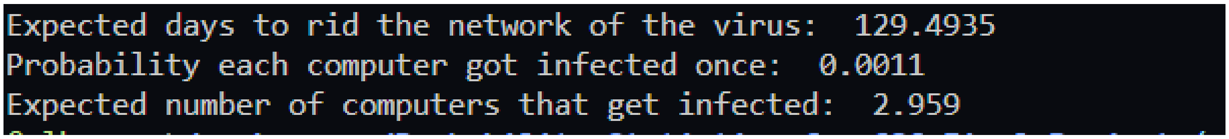
$$P(I_1 \cup I_2 \ldots \cup I_n) = 1 - P(notI)^n$$

Given the updated probability, I use random.random() to generate a number between 0 and 1. I would return 1 to signal that the infection spread if the number was less than the updated probability.

For the IT cleaning it was a bit more tricky. I pointed the top row of the 2d array to a regular array to use the extensive array library. Using this library I generated a new array named infected_indices that looped through the array, saving the indexes of all the infected computers. I then created an array, computers_to_be_cleaned that used random.sample which takes in an array and how many to sample. I used the min method to account for the scenario where the number of infected computers is less than the number of computers IT removes in a day. Using a for loop, the computers_to_be_cleaned array and the top row array, the IT event had been simulated. Now that I had a day simulated I needed to find a way to make the trail run until all computers had been cleaned. I used a while loop and a method to check if the top row was all zeros(no more virus).

Next step would be to figure out a way to run it x number of trials and to save the data from each trial. To do that I had go to through all my code, move them into methods and to take 3 inputs. They were the number of computers in a network, the probability that the virus would spread to a computer, and the number of computers IT would clean in a day. In order to get the data from all the trials I had the trials return an array with the number of days it took to rid the network of the virus in the first index, whether or not each computer had been infected at least once in the second index, and the number of computers in the network that had been infected in the third index. I would then total these values inside of the for loop I used to run the x number of trials. At the end I divided the totals by the number of trials to get the expected time it takes to remove the virus from the whole network, the probability that each computer gets infected at least once, and the expected number of computers that get infected.

After running the simulation 10000 times here are the results:

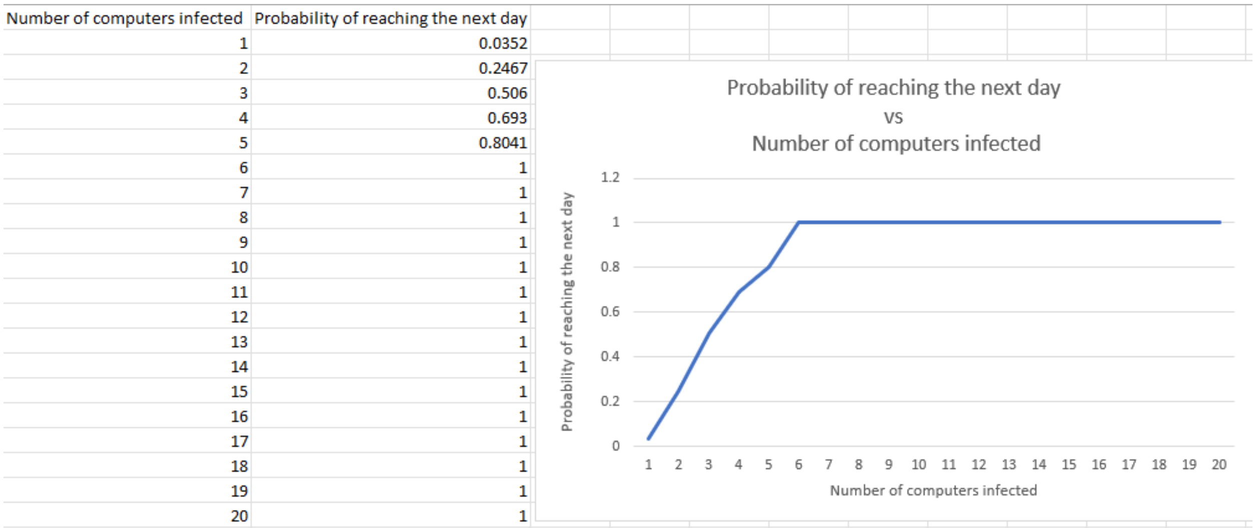


```
Expected days to rid the network of the virus:  129.4935
Probability each computer got infected once:  0.0011
Expected number of computers that get infected:  2.959
```

The probability each computer got infected once and the expected number of computers that got infected are about what I expected when working on this project but the expected days to rid the network of the virus was much higher than I thought it would be. The average number of days it takes to clean the network is 129.4935. I was not expecting this high number because to make it to the next day, at least 5 computers needed to be infected to allow the simulation to reach the next day. That means you needed at least a 1 in 10 chance to happen 5 times. To find this its best to use the compliment event where less than 5 computers get infected.

$$1-\left(\left(\frac{1}{10}\right)^{19}+\binom{19}{1}\left(\frac{1}{10}\right)\left(\frac{9}{10}\right)^{18}+\binom{19}{2}\left(\frac{1}{10}\right)^{2}\left(\frac{9}{10}\right)^{17}+\binom{19}{3}\left(\frac{1}{10}\right)^{3}\left(\frac{9}{10}\right)^{16}+\binom{19}{4}\left(\frac{1}{10}\right)^{4}\left(\frac{9}{10}\right)^{15}\right)$$

After calculating this we get .0352. This is very low so it was curious why the average number of days was so high when just the probability to reach the next day after day 1 was so low. After digging through the results I found the culprit. While making it to the next day was very slim, if there was more than 1 computer that survived the first day, the probability of making it to the second day was much higher. The graph below is the comparison between number of computers infected and the probability that the virus will make it to the next day.



Number of computers infected	Probability of reaching the next day
1	0.0352
2	0.2467
3	0.506
4	0.693
5	0.8041
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1

*note: Networks that have more than 5 computers infected are guaranteed to move on to the next day.

As a result there reaches a critical point where the virus overcomes the IT removal rate of 5 computers a day and it takes tens of thousands, sometimes hundreds of thousands of days to eliminate the virus.

The probability that each computer gets infected at least once, essentially if all the computers in the network have been infected once is 0.0011. So out of 10000 only 11 got completely infected. This lines up with the established idea in the previous couple paragraphs that it is rare for a network to get completely infected considering how rare it is to make it past the first day.

The average number of computers infected is slightly above 2.9 which is also expected. Consider that the simulation starts off with 1 infected. 2.9 – 1 is 1.9 and with the remaining number of computers that could be infected being 19 and the probability that a computer gets infected is .1, the value of 2.959 appears to be accurate.