COSC326 Etude 12:

Part 3:

> “Suppose a magic power is contagious. If one contacts a person with the magic power, they are likely to get the magic power. More contacts mean more likely to get the magic power. For a group of persons with the magic power with their times of empowerment, find out all other persons who are likely to get the power.”

This magic power doesn’t have much information around how contagious / rate of spread so we have implemented a method that will show if any user has been in contact with a powered person. The method will initialize with a list of all users who have the magic power, it will then add to this list, everyone that meets either patient zero or someone who has also been in contact with patient zero. This covers a wide array of people and is a broader way of mapping the potential infection.

Another way to map the infection would be using a probability of becoming infected, e.g., infection chance = 20% and Pn = a contact, then that contact will have a chance of contracting the powers for person Pn - 1. We can map the chance of Pn with the formula Pn chance = ∑ Pi’s chance * 0.2 (for each pi belongs to s, the set of contacts from pn).

Example being. MP = infected person, which means their chance is 100%. So, if A contacts MP, A now has the infection rate (20%) chance of contracting the powers. If B now contacts A, B will have A’s_chance * infection_rate chance of contracting the powers. If B then also contacts MP, B’s chance now equals B’s_chance + infection_rate from MP. Now, if C contacts, A, B and MP, C’s chance = A’s_chance * infection_rate + B’s_chance * infection_rate + MP's_chance * infection_rate, where the infection_rate is 20% and MP's_chance is 100%

[Start of general formula]

Let P be a healthy person.

Let infection_rate = 20%

P lives in a community S of (n+1) people where i = 0 to n = number of people.

Si = 1 if P meets person.

= 0 otherwise

For any single meeting with someone from S, F the probability of a healthy person getting infected.

F = 1 if I am the base case, otherwise

= f for an infectious person (other than the base case)

= 0 for a healthy person

The probability of person i being infectious is Ci

Ci = 1 if person i has ever met the bases case, otherwise

= 0 if person i has never met an infectious person

= f^k where k is the number of infectious people I has met

P is the probability of catching the virus

P = 1 if has met the base case, i = 0

= sum (i = 1 to n) Si * Ci * infection_rate

When p is the same for all meeting constant c, the probability of getting infected by an infectious person. Assume that c is the same for all the i someone that you meet. that a person gets infected.

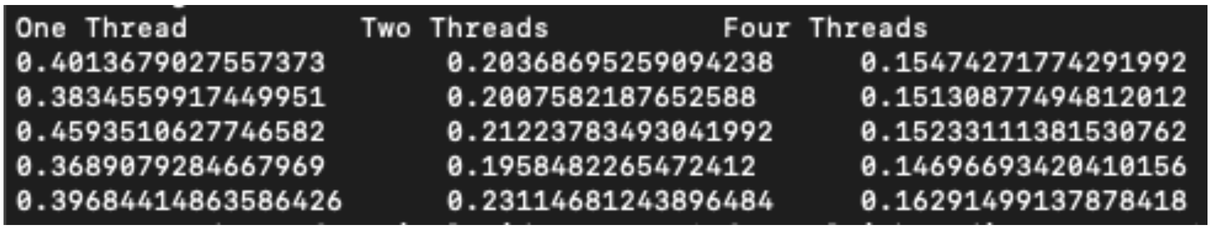
P = [sum over i] S(i) * C(i)

If someone who has already been in contact with an infected person / indirect contact then we add that probability to that individuals chance and if their probability is greater than 100 we can assume they are infected.

Part 4:

Multithreading is used to make tasks like searching through a large data set quicker by splitting the set and using different threads of the CPU to search the smaller sets. This in turn makes it quicker and for this example, we split the dictionary of contact traces into halves and quarters to make the task quicker.

Performance Table:



```
One Thread          Two Threads          Four Threads
0.4013679027557373       0.20368695259094238     0.15474271774291992
0.3834559917449951       0.2007582187652588      0.15130877494812012
0.4593510627746582       0.21223783493041992     0.15233111381530762
0.3689079284667969       0.1958482265472412      0.14696693420410156
0.39684414863586426      0.23114681243896484     0.16291499137878418
```

We are quite happy with our performance table, as it shows increased performance with more threads. One thing to note is a dictionary with 1000 entries was not enough to make multithreading efficient but when we used a data set of 1,000,000 entries like the table above, you can see the results are much better. Considering these results are on a MacBook Air, I'm sure that with a better CPU you would be able to get much better results with larger data sets and also use more threads. In the table we can see that one thread has a time between 0.36 to 0.45 seconds, and then we look at two threads which has halved that time to a range of 0.19 to 0.23 and then with four threads we can see a further decrease in time for four threads with the range being 0.14 to 0.16 which is what we expected and we are very pleased with these results