Graph Data Mining Project 5 - Virus Propagation

Wen-Han Hu (whu24)		Yang-Kai Chou (ychou3)		Yuan Xu (yxu48)

Option 1: Virus Propagation on Static Networks

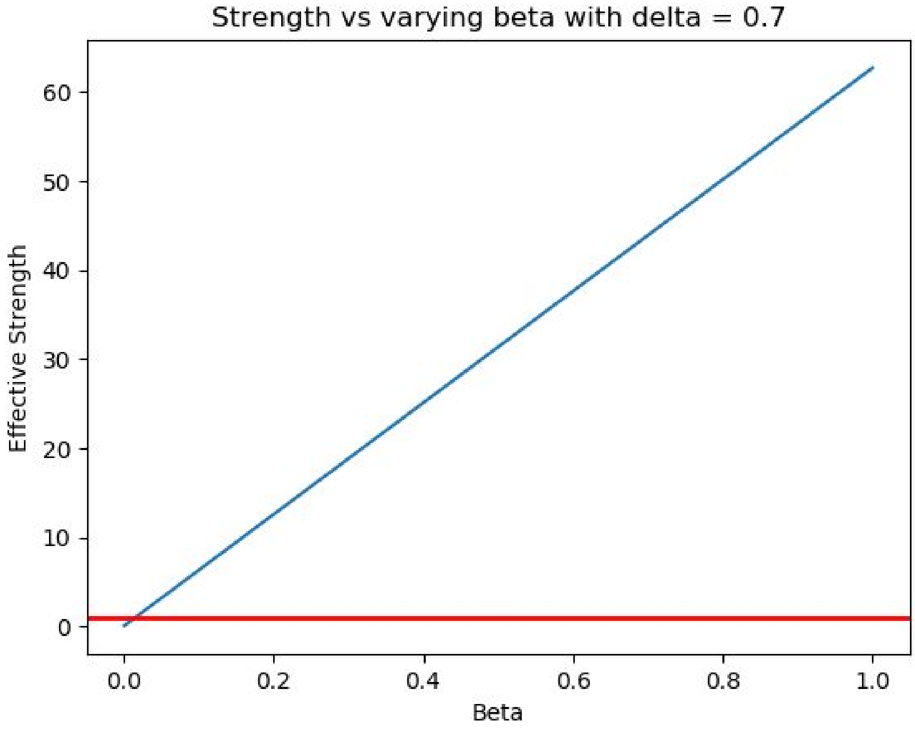
1.Calculate the effective strength (s) and answer the following questions.

a. Will the infection spread across the network (i.e., result in an epidemic), or will it die quickly?
Ans:

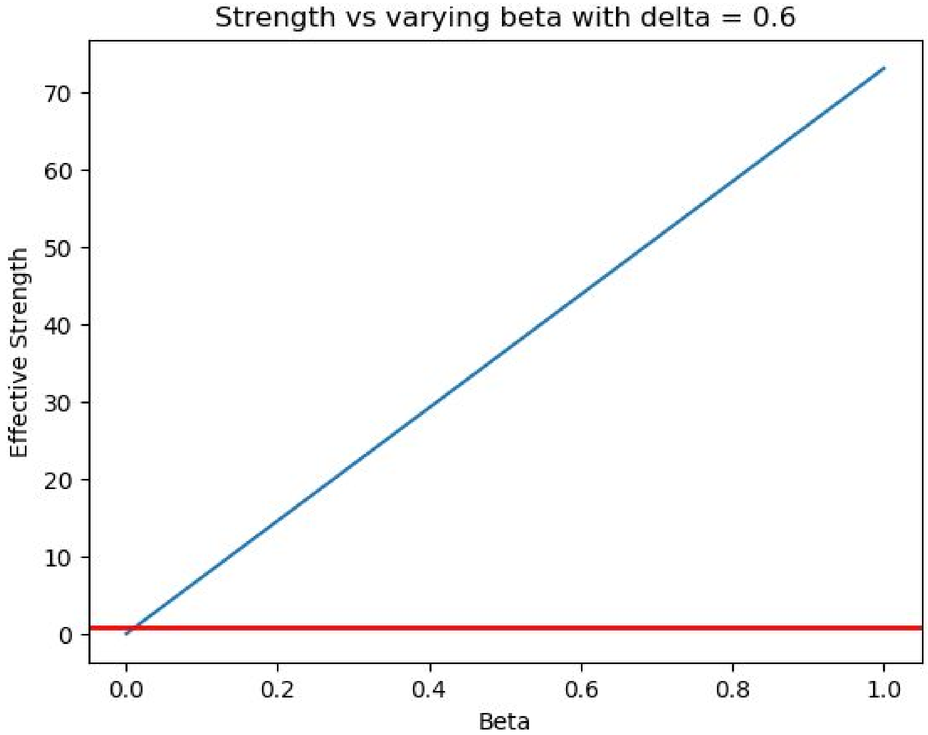
1. For β = β1 (0.20) and δ = δ1 (0.70), the effective strength is 12.529913.
   Since effective strength is greater than 1, the infection will result in an epidemic.
2. For β = β2 (0.01) and δ = δ2 (0.60), the effective strength: 0.7309116
   Since effective strength is less than 1, the virus will not result in an epidemic and die quickly.

b. Keeping δ fixed, analyze how the value of β affects the effective strength of the virus (suggestion: plot your results). What is the minimum transmission probability (β) that results in a network-wide epidemic?
Ans:
For this question, we tuned the value of β from 0.001 to 1 and increment by 0.001 and the results as following.

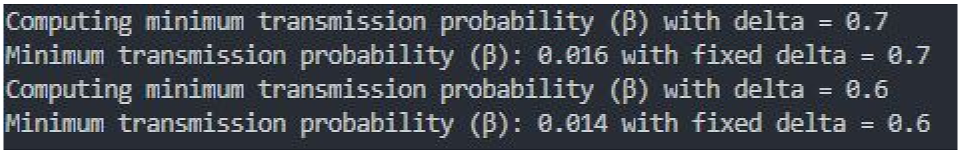
1. Fixed δ = δ1 (0.70), the minimum transmission probability (β) is 0.016.



2. Fixed δ = δ1 (0.60), the minimum transmission probability (β) is 0.014.



.

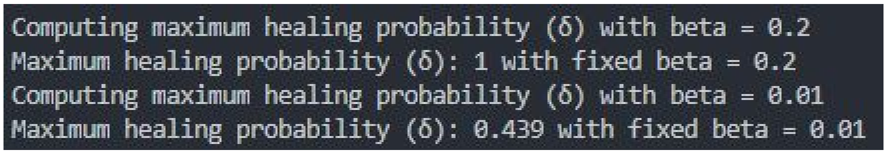


```
Computing minimum transmission probability (β) with delta = 0.7
Minimum transmission probability (β): 0.016 with fixed delta = 0.7
Computing minimum transmission probability (β) with delta = 0.6
Minimum transmission probability (β): 0.014 with fixed delta = 0.6
```

c. Keeping β fixed, analyze how the value of δ affects the effective strength of the virus (suggestion: plot your results). What is the maximum healing probability (δ) that results in a network-wide epidemic?

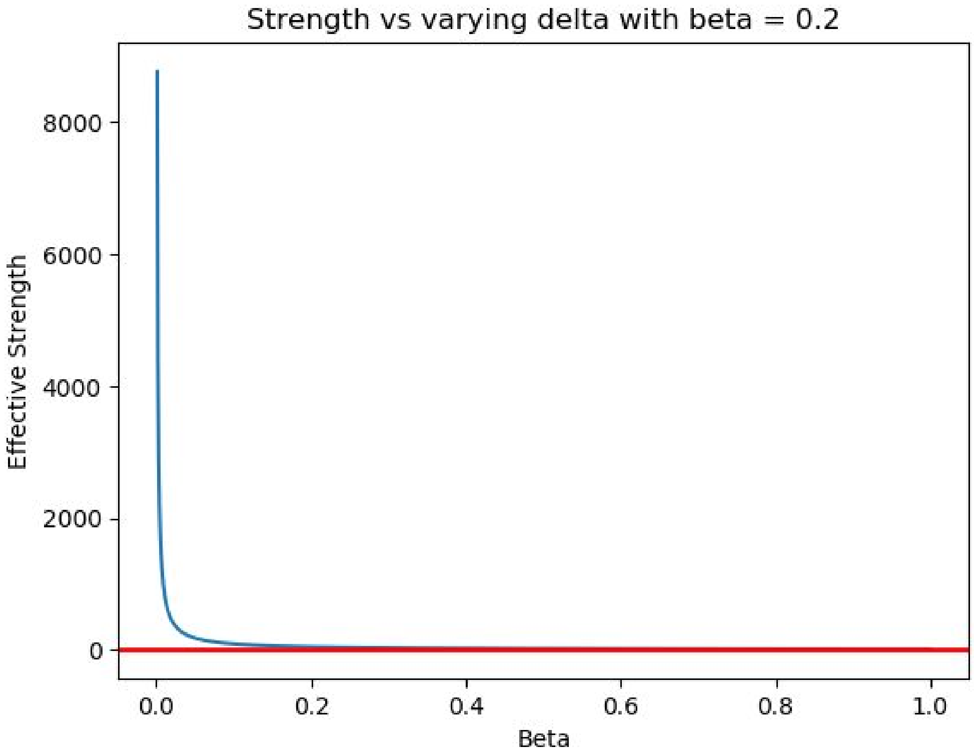
Ans:

For this question, we tuned the value of δ from 0.001 to 1 and increment by 0.001 and the results as following.

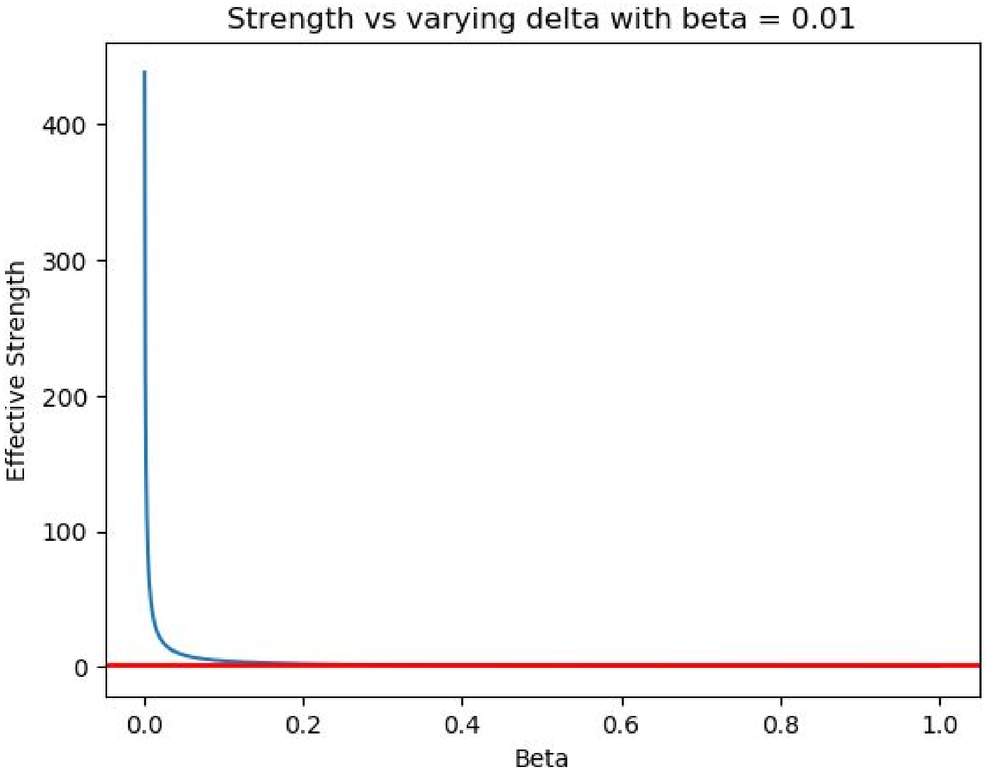


```
Computing maximum healing probability (δ) with beta = 0.2
Maximum healing probability (δ): 1 with fixed beta = 0.2
Computing maximum healing probability (δ) with beta = 0.01
Maximum healing probability (δ): 0.439 with fixed beta = 0.01
```

1. Fixed β = 0.2, the maximum healing probability (δ) is 1. Which means that there is definitely resulting in network-wide epidemic with this value of β.



2. Fixed β = 0.01, the maximum healing probability (δ) is 0.439

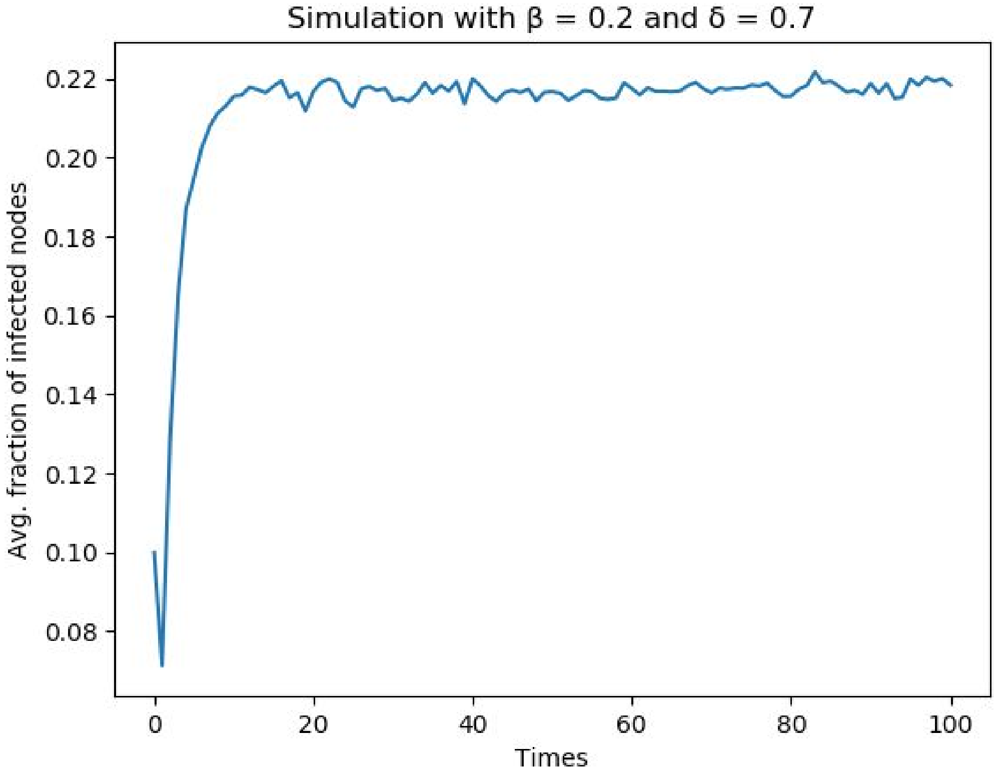


2. Simulates the propagation of virus with the SIS VPM and answer the following questions.

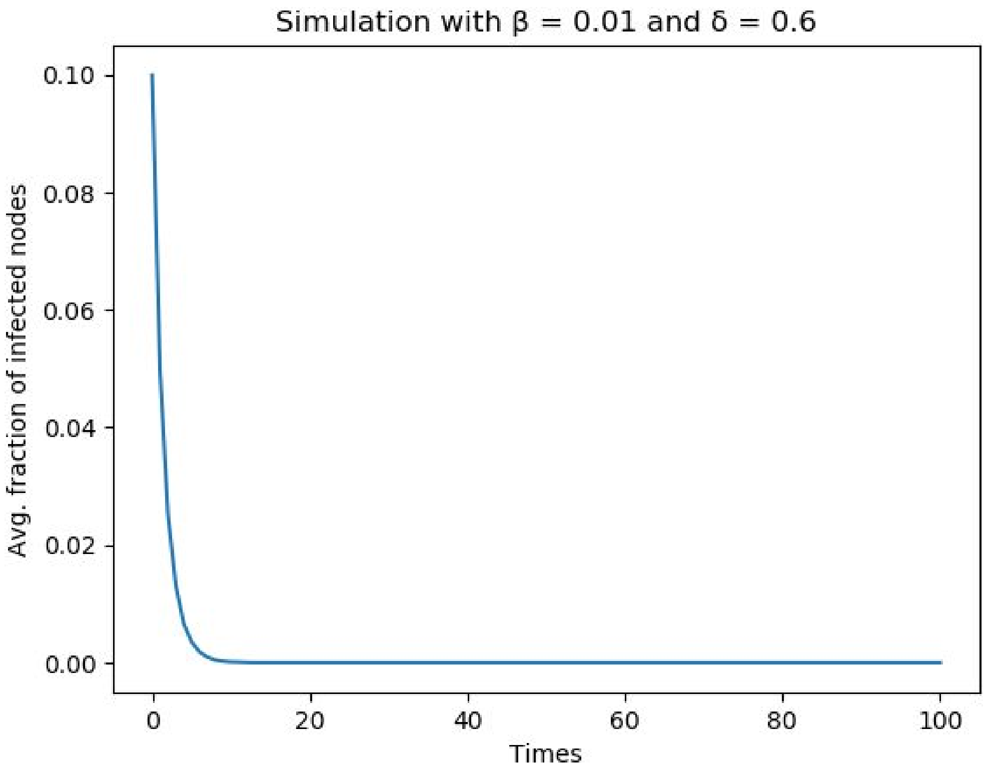
b. Plot the average (over the 10 simulations) fraction of infected nodes at each time step. Did the infection spread across the network, or did it die quickly? Do the results of the simulation agree with your conclusions in (1a)?

Ans:

1. For β = β1 (0.20) and δ = δ1 (0.70), we can observe that it results in an epidemic. The result from simulation is consistent with the result we had on 1(a).



2. For β = β2 (0.01) and δ = δ2 (0.60), we can observe that virus die quickly and it does not result in an epidemic. The result from simulation is consistent with the result we had on 1(a).



3. Implements an immunization policy to prevent the virus from spreading across the network. Answer the following question.

a. What do you think would be the optimal immunization policy? What would be its time complexity? Would it be reasonable to implement this policy? Justify.

Ans:

The optimal immunization policy will be finding the most significant subset of k nodes among all n. Indicators of centrality, such as eigenvalue, degree, pagerank...etc, are great indexes to identify the importance of node among the network. Plus, the time complexity should be within polynomial time or the solution is computationally intractable and the immunization policy is not possible to apply.

For each heuristic immunization policy (A, B, C, and D) and for the static contact network provided (static.network), answer the following questions:

b. What do you think is the intuition behind this heuristic?

Ans:

- Policy A : The intuition is to immunize random nodes in the network and expect that immunizing these nodes will prevent the virus from spreading across the network. If the nodes of network is large and number of k is small then it is the least effective way to do immunization.

● Policy B : This intuition is to target the nodes with the highest degree. In other words, this approach aims to control the virus spread by top most effective k nodes which is more effective than random nodes selection.

● Policy C : The intuition is to constantly track the most effective nodes instead of determining all the immunized nodes in the initial state. Since the degree of nodes will change when we immunize the node each time.

● Policy D : The intuition here is that the nodes with high eigenvector score means that a node is connected to many nodes who themselves have high scores. It is also a way to find the significant nodes and effectively avoid virus propagation.

c. Write a pseudocode for this heuristic immunization policy. What is its time complexity?
Ans:

Policy A:
Input: adj: adjacency matrix of graph, k: number of vaccines
Output adj: Updated adjacency matrix with specific node with immunization.

```
def policyA(adj, k):
    immun = set() //initial set to store immunized nodes
    while len(immun) < k: // get k random numbers
        r = random.randint(0, total_nodes - 1)
        if r not in immun:
            immun.add(r)

    for node in immun:
        for i in range(len(adj[node])):
            adj[node][i] = 0
            adj[i][node] = 0
    return adj
```

Time complexity analysis:
Original adjacency matrix generation: $O(N^2)$
Select k random numbers: $O(k)$
Update adjacency matrix: $O(kN)$
Thus, the total time complexity of policy A is $O(N^2 + k + kN) = O(N^2)$

Policy B:

Input: adj: adjacency matrix of graph, k: number of vaccines

Output adj: Updated adjacency matrix with specific node with immunization.

```
def policyB(adj, k):
    immun = set() //initial set to store immunized nodes
    degree = sorted(list(g.degree()), key=lambda x: x[1], reverse=1) // sort degree list
    for i in range(k): // get k largest degree nodes
        immun.add(degree[i][0])

    for node in immun:
        for i in range(len(adj[node])):
            adj[node][i] = 0
            adj[i][node] = 0
    return adj
```

Time complexity analysis:

Original adjacency matrix generation: $O(N^2)$

Sort degree list: $O(N logN)$

Select k largest degree nodes : $O(k)$

Update adjacency matrix: $O(kN)$

Thus, the total time complexity of policy B is $O(N^2 + NlogN + k + kN) = O(N^2)$

Policy C:

Input: k: number of vaccines

Output adj: adjacency matrix with specific node with immunization.

```
def policyC(k):
    immun = set()  //initial set to store immunized nodes
    while len(immun) < k:
        degree = sorted(list(g.degree()), key=lambda x: x[1], reverse=1) // sort degree list
        immun.add(degree[0][0])
        g.remove_node(degree[0][0])

    adj = [[0 for _ in range(total_node)] for _ in range(total_node)]
    for i in nx.edges(g):
        adj[i[0]][i[1]] = 1
        adj[i[1]][i[0]] = 1
    return adj
```

Time complexity analysis:

Sort degree list: $O(kN logN)$

Remove node from graph : $O(kN^2)$

Adjacency matrix generation: $O(N^2)$

Thus, the total time complexity of policy C is $O(kNlogN + kN^2 + N^2) = O(kN^2)$

Policy D:

Input: adj: adjacency matrix of graph, k: number of vaccines

Output adj: adjacency matrix with specific node with immunization.

```
def policyD(adj, k):
    val, vec = np.linalg.eig(adj)
    eig_set = [(val[i], vec[i]) for i in range(len(val))]
    eig_set = sorted(eig_set, key=lambda x: x[0], reverse=1)
    largest_vec = eig_set[0][1]
    largest_vec = np.absolute(largest_vec)
    target = [u[0]
              for u in sorted(enumerate(largest_vec), reverse=True, key=itemgetter(1))[:k]]
    for i in target:
        g.remove_node(i)

    // generate new adjacency matrix
    adj = [[0 for _ in range(total_node)] for _ in range(total_node)]
    for i in nx.edges(g):
        adj[i[0]][i[1]] = 1
        adj[i[1]][i[0]] = 1
    return adj
```

Time complexity analysis:

Original adjacency matrix generation: $O(N^2)$

Computing eigenvalues: $O(N^3)$

Sort eigenvalues: $O(NlogN)$

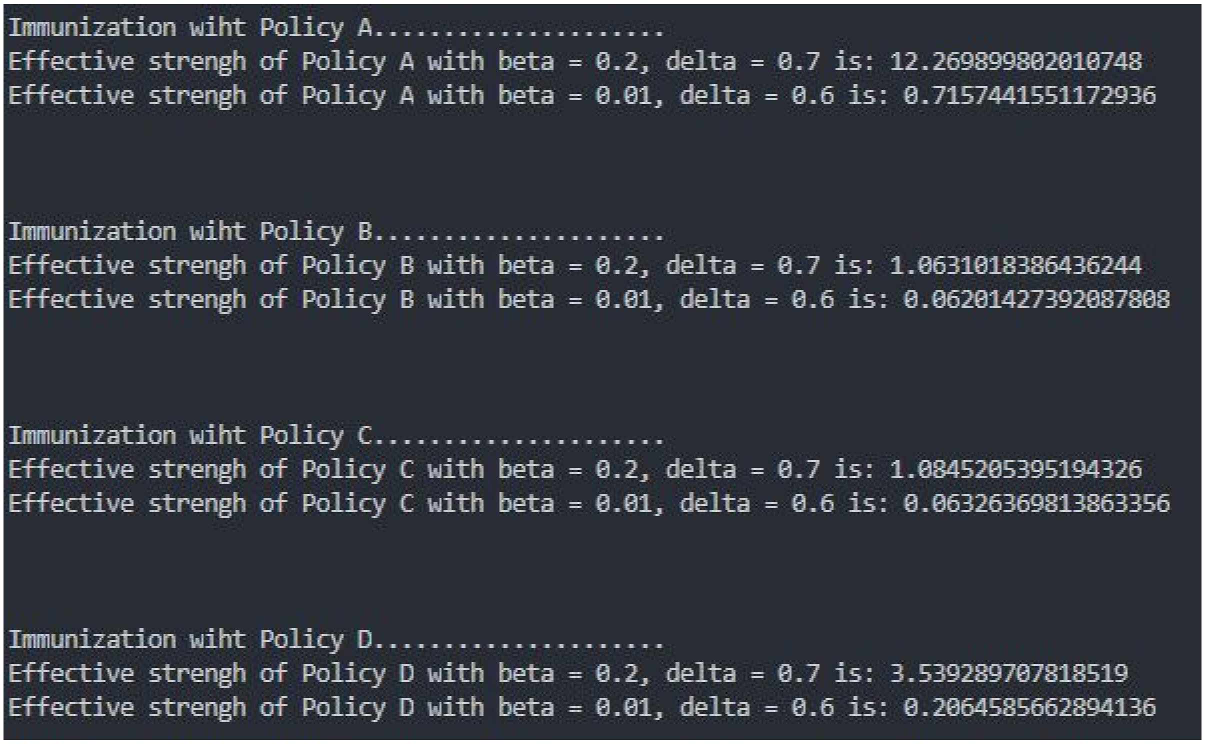
Remove node from graph : $O(kN^2)$

Adjacency matrix generation: $O(N^2)$

Thus, the total time complexity of policy C is $O(N^2 + N^3 + NlogN + kN^2 + N^2) = O(N^3)$

d. Given k = k1 , β = β 1 , and δ = δ 1 , calculate the effective strength (s) of the virus on the immunized contact network (i.e., contact network without immunized nodes). Did the immunization policy prevented a network-wide epidemic?

Ans:



```
Immunization wiht Policy A....................
Effective strengh of Policy A with beta = 0.2, delta = 0.7 is: 12.269899802010748
Effective strengh of Policy A with beta = 0.01, delta = 0.6 is: 0.7157441551172936



Immunization wiht Policy B....................
Effective strengh of Policy B with beta = 0.2, delta = 0.7 is: 1.0631018386436244
Effective strengh of Policy B with beta = 0.01, delta = 0.6 is: 0.06201427392087808



Immunization wiht Policy C....................
Effective strengh of Policy C with beta = 0.2, delta = 0.7 is: 1.0845205395194326
Effective strengh of Policy C with beta = 0.01, delta = 0.6 is: 0.06326369813863356



Immunization wiht Policy D....................
Effective strengh of Policy D with beta = 0.2, delta = 0.7 is: 3.539289707818519
Effective strengh of Policy D with beta = 0.01, delta = 0.6 is: 0.2064585662894136
```

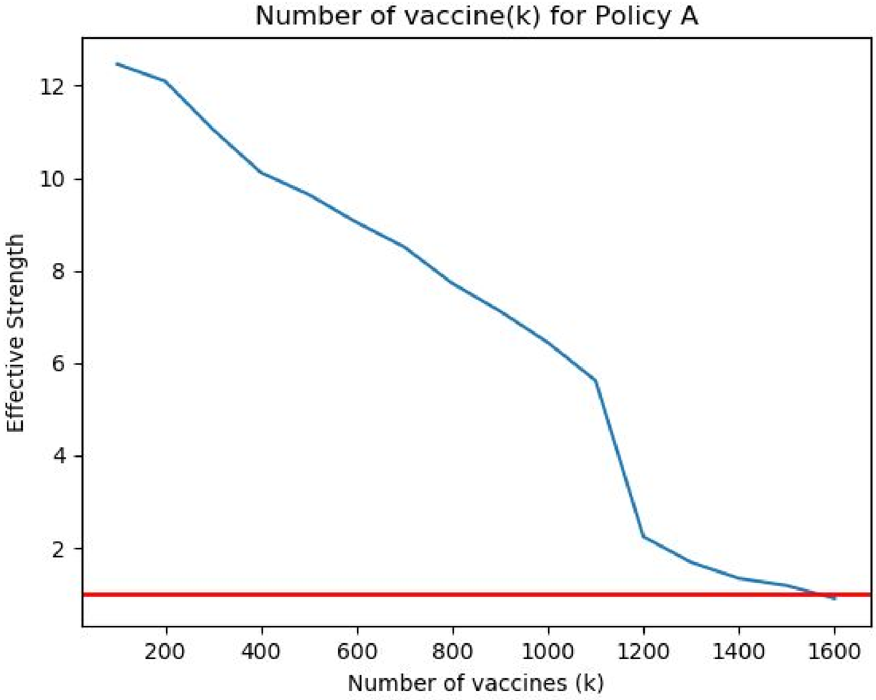
With the results above, we can obviously find that policy B, C have better performance to prevent a network-wide epidemic but all of them are greater than 1. We can only conclude that policy A and D will not prevent a network-wide epidemic and policy B and C may have chance to prevent a network-wide epidemic.

e. Keeping β and δ fixed, analyze how the value of k affects the effective strength of the virus on the immunized contact network (suggestion: plot your results). Estimate the minimum number of vaccines necessary to prevent a network-wide epidemic.

Ans:

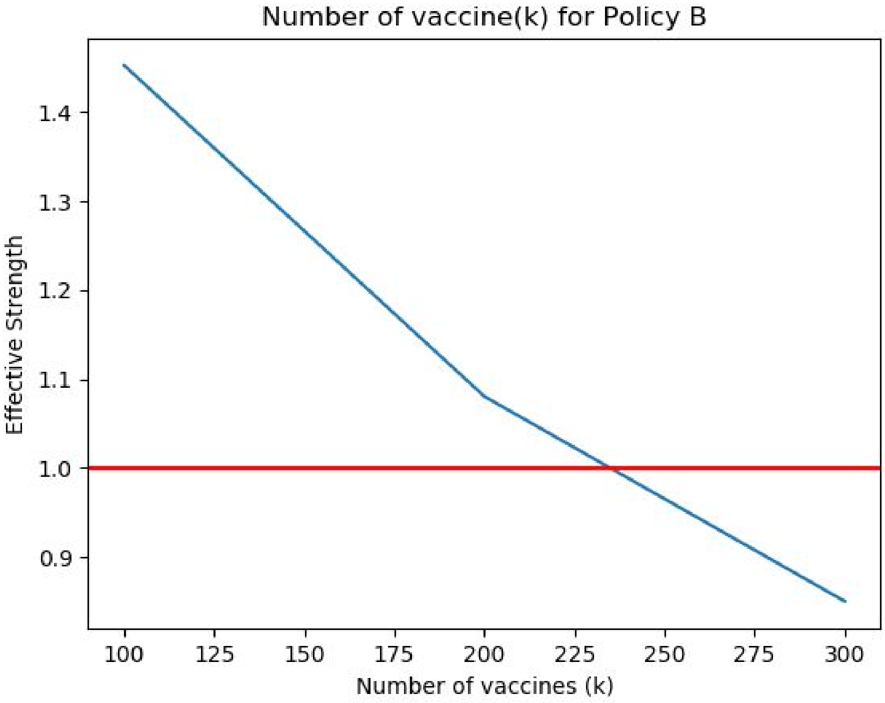
For this question, we first fixed β = β 1 , and δ = δ 1 and tune the value of k starting from 100 nodes, incremented by 100 nodes until the strength is less than 1 to estimate the minimum number of vaccines necessary to prevent a network-wide epidemic.

Policy A:



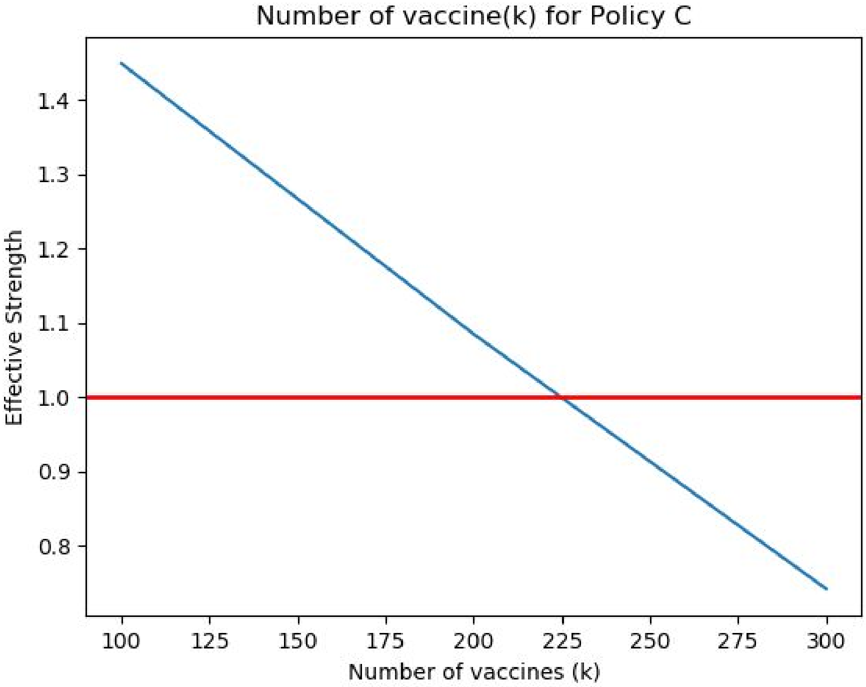
When the number of vaccines greater than 1600, we have effective strength less than 1which will prevent a network-wide epidemic. Since we increment k by 100, the minimum number of vaccines is within 1500 to 1600.

Policy B:



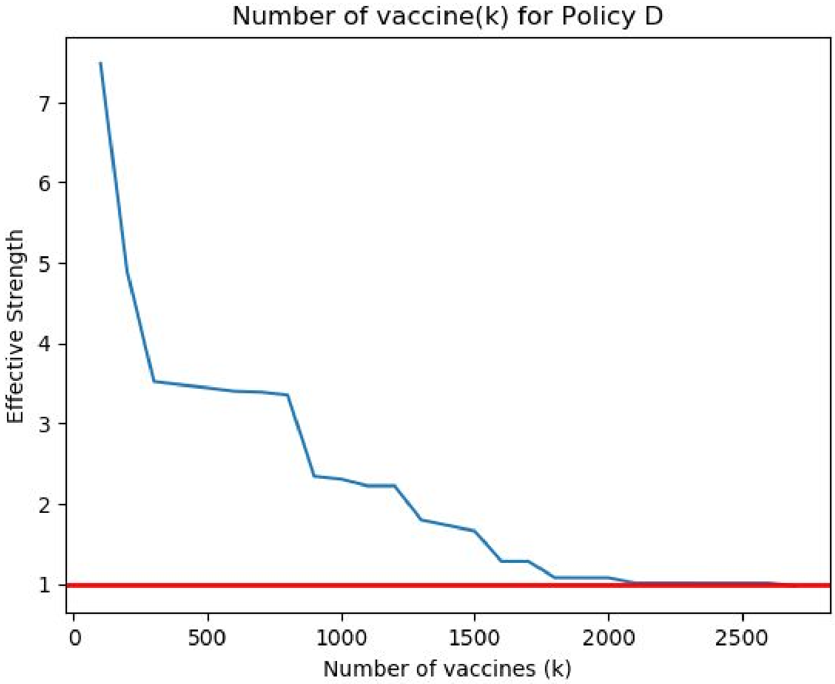
When the number of vaccines greater than 300, we have effective strength less than 1 which will prevent a network-wide epidemic. Since we increment k by 100, it is possible the minimum number of vaccines is within 200 to 300.

Policy C:



When the number of vaccines greater than 300, we have effective strength less than 1 which will prevent a network-wide epidemic. Since we increment k by 100, minimum number of vaccines is within 200 to 300.

Policy D:

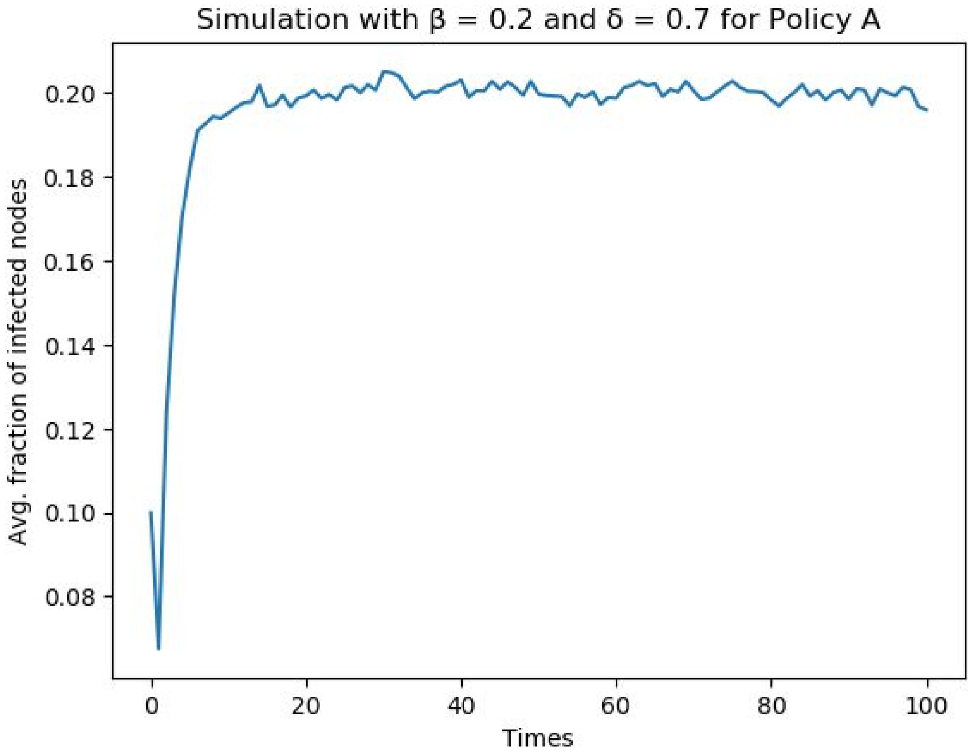


When the number of vaccines greater than 2800, we have effective strength less than 1 which will prevent a network-wide epidemic. Since we increment k by 100, the minimum number of vaccines is within 2700 to 2800.

f. Given k = k1 , β = β 1 , δ = δ 1 , c = n/10, and t = 100, run the simulation from problem (2) for the immunized contact network 10 times. Plot the average fraction of infected nodes at each time step. Do the results of the simulation agree with your conclusions in (3d)?

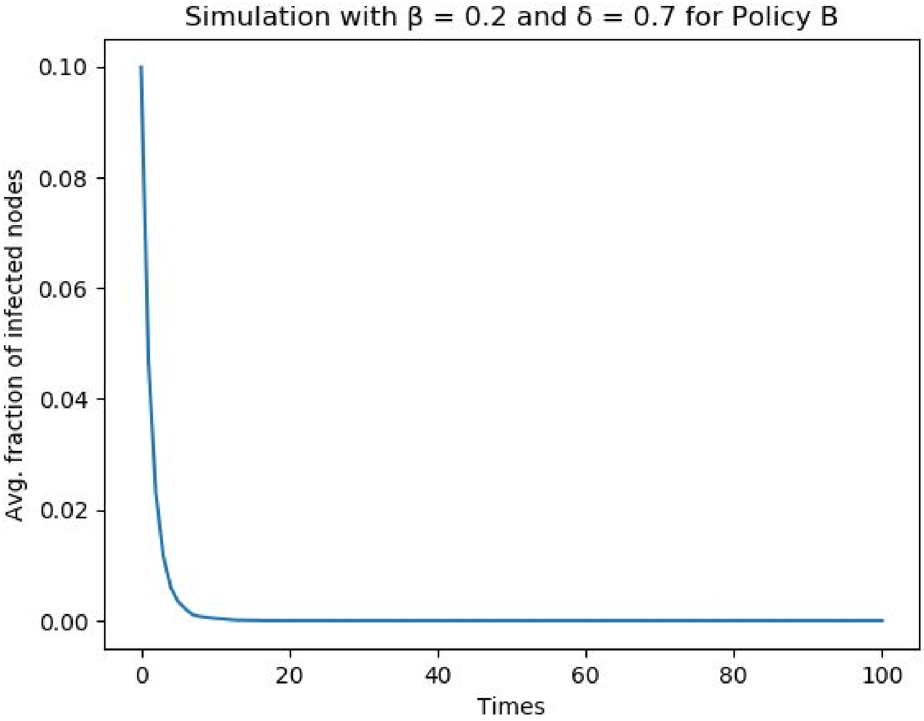
Ans:

Policy A:



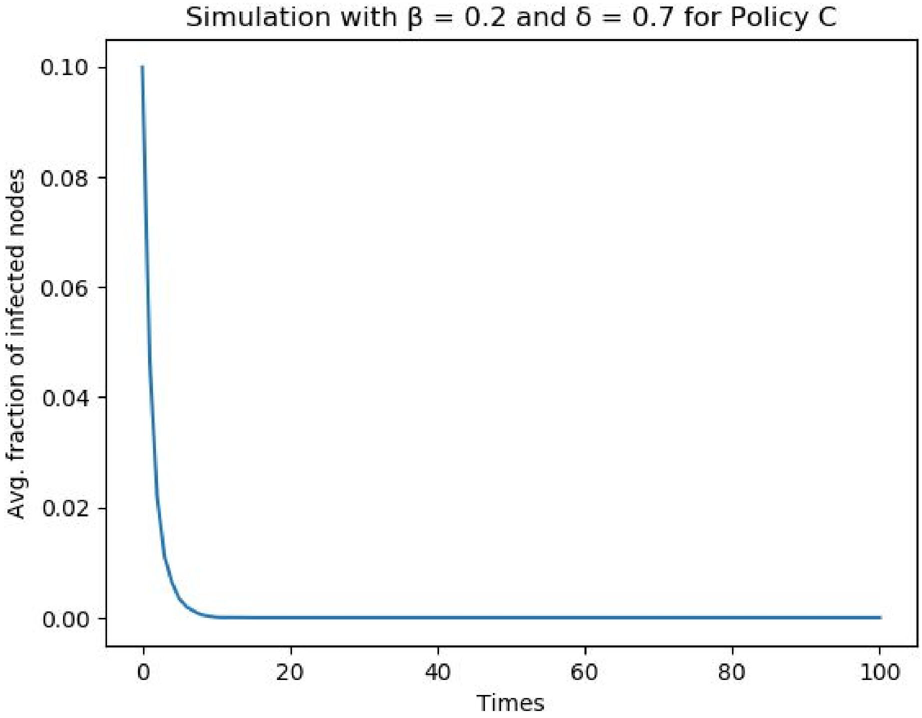
The result of simulation indicates that Policy A still results in an epidemic. The result is consistent with the conclusion from (3d).

Policy B:



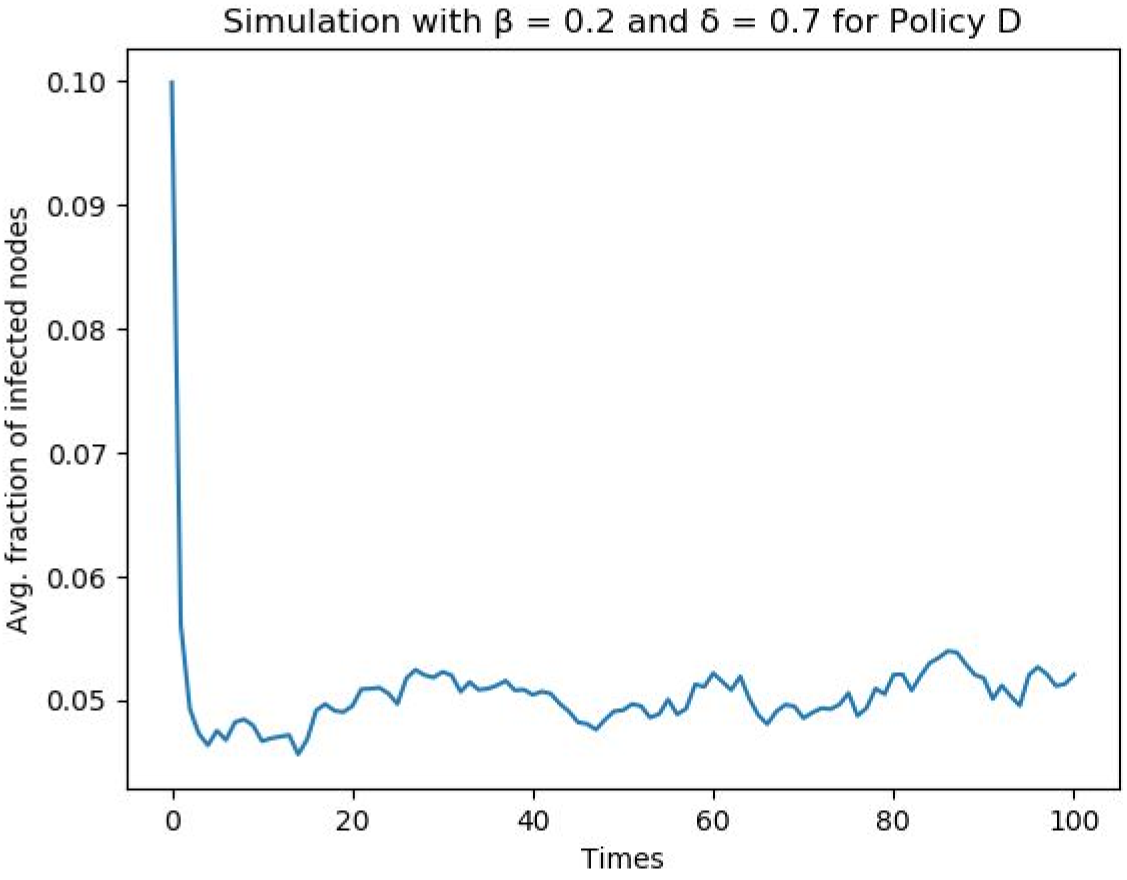
The result of simulation indicates that Policy B will not result in an epidemic. The result is consistent with the conclusion from (3d).

Policy C:



The result of simulation indicates that Policy C will not result in an epidemic. The result is consistent with the conclusion from (3d).

Policy D:



The result of simulation indicates that Policy D may or may not result in an epidemic. The result is consistent with the conclusion from (3d).