Theory Assignment-3

Roll No: 2021055 **Section: A**

Name: Jyotirmaya Singh **Specialisation: CSE**

1) <u>Answer to Question 1</u>:

Formulating the problem as a Network Flow Problem:

Let c_1 and c_2 be any two towns/villages on the island that have a railway line between them. Since the railway line is bidirectional (i.e., we can travel from c_1 to c_2 or from c_2 to c_1 using the railway line connecting them), we use two directed edges to represent the railway line between c_1 and c_2 (one directed edge from c_1 to c_2 and another directed edge from c_2 to c_1). We define the capacity of each of these directed edges to be 1.
In case there are multiple railway lines between c_1 and c_2, we still follow the same method described above.

We have chosen Tinkmoth to be the source and Doweltown to be the sink without loss of generality.
Since our graph was given to be undirected, the value of the max flow from Tinkmoth to Doweltown will have the same value as the max flow flow from Doweltown to Tinkmoth.
Hence, we can choose either of Tinkmoth or Doweltown to be the source or sink since their min cuts will have the same capacity.

How maximum-flow value in the network is equivalent to the solution to the original problem:

Let our network be represented by G(V, E).
A cut is a node partition (S, T) such that s (source) is in S and t (sink) is in T.
Capacity of the cut (S, T) is defined as the sum of capacities of edges from any vertex in S to any vertex in T.

Since we have to prevent the spread of disease to Doweltown while minimizing expense, we are required to put the minimum number of traffic blocks on the railway lines such that no train can reach from Tinkmoth to Doweltown.

Theory Assignment-3

Roll No: 2021055 **Section: A**

Name: Jyotirmaya Singh **Specialisation: CSE**

The min-cut theorem states that the maximum flow in a network is equal to the capacity of the min-cut for that network. Thus, if we can find the maximum-flow in our network, we will get the minimum number of outgoing edges from partition S (which contains the source) to partition T (which contains the sink).

Say, we found our max flow to be *F*, we then place *F* number of barricades between the edges connecting any vertex in S to any vertex in T. Since the source is in S and the sink is in T, we have blocked all paths from s to t.
Note, that the directed nature of edges has been introduced by us to solve the problem, in reality just placing a single barricade on the railway line is sufficient to preventing the disease from spreading between two towns. Therefore, we need only *F* barricades.

A min-cut of the network in a sense limits the amount of flow through a network, as it prevents the flow from passing through the edges crossing the cut.
Thus, the capacity of the min cut can be thought of as a bottleneck flow for the network, as it limits the amount of flow that can be passed through the network. Hence, the capacity of the min-cut will determine the maximum flow of the network.

Explanation:

We run the Ford Fulkerson algorithm on our network to find the maximum flow subject to the capacity constraints.

Explanation of Ford Fulkerson Algorithm:

1) In this algorithm, each bidirectional edge will be treated as one forward edge which will carry the flow in the forward direction and one reverse edge which carry the flow in the reverse direction. We will also maintain a global max-flow counter variable for calculating the maximum flow and initialize it to 0.
2) The algorithm starts by initializing the initial flow to 0 and the initial capacity to 1 for all the edges in the network.
3) Augmenting path is a path from source to sink where each edge in the path has non-zero residual capacity (initial capacity - flow value).

Theory Assignment-3

Roll No: 2021055 **Section: A**

Name: Jyotirmaya Singh **Specialisation: CSE**

4) The augmenting paths can be found using any traversal technique. We are using the DFS traversal technique for finding the augmenting paths.
5) Once, an augmenting path is found, the minimum residual capacity is selected from the edges in the path and then the flow for all the edges in that path is increased by that value. The global max-flow counter variable is also increased by this value.
6) After increasing the flow along an edge, the residual capacities of that forward edge and reverse edge have to be updated. For example, if a flow f is being sent through the forward edge with residual capacity c_1, the residual capacity updates and becomes c_1-f and if for the reverse edge the residual capacity is c_2, then the residual capacity of the reverse edge updates and becomes c_2+f.
7) The algorithm repeats the process of finding an augmenting path until there are no more augmenting paths that exist from source to sink. At this point, the algorithm has completed and terminates.
8) We have the maximum flow value of the network stored in the max-flow global counter variable.
9) We return the max-flow global counter variable as the solution to our problem.

Pseudocode:

```
Ford-Fulkerson(G, s, t):
    // Initialize the max_flow global counter to 0
    max_flow = 0
   // Initialize flow to zero on all edges
   // Initialize capacity to one for all the edges
   for each edge e in G:
      e.flow = 0
      e.capacity = 1

   while there exists an augmenting path P in the residual graph:
      // Find the minimum residual capacity along the path
      min_capacity = infinity
      for each edge e in P:
         min_capacity = min(min_capacity, e.capacity)
      // Updating the max_flow
      max_flow = max_flow + min_capacity
```

Theory Assignment-3

Roll No: 2021055 **Section: A**

Name: Jyotirmaya Singh **Specialisation: CSE**

```
    // Update flow along the path
    for each edge e in P:
        e.flow = e.flow + min_capacity
        e.capacity = e.capacity - min_capacity
        e.reverse.capacity = e.reverse.capacity + min_capacity
        e.reverse.flow = e.reverse.flow - min_capacity // for bidirectional graphs

return max_flow
```

Running Time:

The runtime of Ford-Fulkerson is O(F|E|) where F is the maximum flow possible in the network. This is a pseudopolynomial time complexity.
However, since we have assumed all our edges to be of unit weight, the maximum flow possible between the source s and sink t will be |V-2| , i.e., when the source is connected to every vertex other than itself and the sink. Thus, the time complexity of our algorithm becomes O(|V-2||E|) = O(|V||E|). This is in the case when we have only a single railway line connecting any two towns/villages.

In case we have multiple edges (i.e., there are multiple railway lines between two towns/villages), then the maximum flow possible becomes |E/2| (as when we converted the given problem to a network problem we introduced two edges to represent a railway line between two towns). Note that the capacities of the edges have been defined to be 1, therefore, the max flow value becomes a function of the number of edges in the network. Thus, the time complexity of our algorithm becomes $O((|E|/2)|E|) = O(|E|^2)$.