Theory Assignment – 3
ADA Winter - 2023

Naman Aggarwal | 2021070
Ashutosh Gera | 2021026

Problem: The towns and villages of the Island of Sunland are connected by an extensive rail network. Doweltown is the capital of Sunland. Due to a deadly contagious disease, recently, few casualties have been reported in the village of Tinkmoth. To prevent the disease from spreading to Doweltown, the Ministry of Railway of the Sunland wants to completely cut down the rail communication between Tinkmoth and Doweltown. For this, they wanted to put traffic blocks between pairs of rail stations that are directly connected by railway track. It means if there are two stations x and y that are directly connected by railway line, then there is no station in between x and y in that particular line. If a traffic block is put in the track directly connecting x and y, then no train can move from x to y. To minimize expense (and public notice), the authority wants to put as few traffic blocks as as possible. Note that traffic blocks cannot be put in a station, it has to be put in a rail-track that directly connects two stations.

Formulate the above as a flow-network problem and design a polynomial-time algorithm to solve it.
Give a precise justification of the running time of your algorithm.

Solution:

Basic Idea / Intuition

We see that this is one of the main applications of the *Network Flow* problem. We can formulate our problem statement into a flow network and solve the classic *minimum cut problem* which asks us to find the minimum "damage" needed to separate source *s* from sink *t.*

Graph Formulation

To tackle the problem of minimizing rail blockages between Tinkmoth and Doweltown to prevent the spread of a deadly disease, we can model the situation as a network flow problem. We can construct a directed graph, where each town or village is represented by a node and each rail track connecting them is represented by a directed edge i.e. if there is railway track between station 'a' and 'b' then in the graph there will be a directed edge from 'a' to 'b' and from 'b' to 'a'. We set Tinkmoth as the source node and Doweltown as the sink node, and assign a unit capacity to each edge in the graph. Our objective is to minimize the number of edges we need to block, such that there is no path from Tinkmoth to Doweltown. We can formulate this as a minimum cut problem on the graph.

Algorithm

To solve our problem, we use maximum flow algorithm to find the maximum number of edge-disjoint paths from Tinkmoth to Doweltown in the graph. The minimum number of

edges we need to remove is the difference between the total number of paths and the maximum flow. This can be done in polynomial time using our maximum flow algorithm given by Ford-Fulkerson.

We apply Ford-Fulkerson's algorithm on our graph G with source as the vertex 's' which represents the station of city where disease has spread (*Tinkmoth*) and sink as the vertex 't' which represents station of city to which we want to prevent spread of disease(*Doweltown*).

Ford-Fulkerson will give us **maximum flow *'f'*** in the given graph and after finding '*f*' we can find the min cut in the graph as capacity of min cut is equal to max flow of graph (The min cut i.e. minimum number of edges we need to remove is the **difference between the total number of edge-disjoint paths from Tinkmoth to Doweltown and the maximum flow**).

To find the min cut once we have obtained the maximum flow, we can perform a depth-first search or a breadth-first search on the **residual graph** to find the nodes that can be reached from the source node, and these nodes will form one side of the minimum cut. The other side of the minimum cut will be the nodes that cannot be reached from the source node in the residual graph. **The edges that connect the two sides of the minimum cut will be the edges that need to be blocked in order to prevent flow from the source node to the sink node.**

Finding the min cut of any graph gives us the bottleneck of that graph as min cut is set of edges whose sum of capacities is equal to max flow of graph so if we remove the edges of min cut then the graph could be separated into two such parts so that there can be no flow from source to sink. So by finding min cut in our graph we can place a traffic block on min cut edges so that no flow of disease could be there from source to sink.

As we have shown there will be no flow from source to sink in similar fashion there will be no flow from sink to source as in the original problem there were undirected paths which we converted to directed paths (i.e. of there is a path between a and b then directed path from a to b and b to a exists. Blocking any of one directed paths will also block other directed path in reverse direction) so if there exists any path from sink to source then its complementary path would also exist and have been considered to find min cut and putting a block on any edge would block entire path. Hence, our problem has been solved!

Runtime analysis

Time complexity for original Ford-fulkerson is **O(f*E)** which is pseudo polynomial, where f is max flow and E is number of edges in our graph. To find the minimum cut from the Ford-Fulkerson algorithm, we performed an additional search on the residual graph which would take O(V+E) time. In our graph since all edges have same capacity so in worst case if we have 'V' vertices then source could be connected to V-1 vertices and max flow would be V-1 and in that case time complexity would be O((V-1)*e) = **O(V*e) i.e. polynomial time algorithm.**

If there are duplicate edges or railway tracks between two stations i.e. between vertices a and b there are multiple edges. In such a scenario max flow will be equal to 'e' (number of edges) ,we know capacity of each edge is unit only so in worst case if source is directly connected to sink via 'e' edges then max flow will be 'e' then time complexity will be **O(e*e)=O(e**2)**

In all cases, the running time of our algorithm would be polynomial time.