Assignment 7

Meet Desai – 202311031

1. **Let G = (V, E) be a directed graph where for each vertex v, the in-degree and out-degree of v are equal. Suppose G contains k edge-disjoint paths from some vertex u to another vertex v. Under these conditions, must G also contain k edge-disjoint paths from v to u? Give a proof or a counterexample with explanation.**

In the given scenario where the in-degree and out-degree of every vertex in graph G are equal, it is not always true that if there are k edge-disjoint paths from vertex u to v, there must also be k edge-disjoint paths from v to u.

Proof:

Consider a directed graph G=(V,E) where each vertex has an in-degree and out-degree of **2.** Now, if there are k edge-disjoint paths from u to v, it implies that there are k different routes from u to v that do not share any edges.

However, when trying to find k edge-disjoint paths from v to u, it is not always possible. This is because the paths that were disjoint while going from u to v might intersect when traveling from v to u. Due to this intersection, it might not be feasible to find k edge-disjoint paths from v to u.

2. Given an undirected graph G = (V, E), with three vertices u, v, and w, describe and analyze an algorithm to determine whether there is a path from u to w that passes through v.

To determine whether there is a path from vertex u to vertex w that passes through vertex v in an undirected graph G = (V, E), you can use a depth-first search (DFS) algorithm. Below is an algorithmic description along with an analysis:

Algorithm:

- Initialize a Boolean variable, found, to false. This variable will be used to track whether a path from u to w through v is found.
- Start a DFS traversal from vertex u. During the DFS traversal, maintain a visited set to mark vertices that have been visited to avoid revisiting them.
- During the DFS traversal, if you encounter vertex v, mark it as visited and continue the DFS recursively.
- If the DFS traversal reaches vertex w and w is marked as visited (indicating that w is reachable from u through v), set found to true.
- After completing the DFS traversal, check the value of found. If it is true, then there is a path from u to w that passes through v. Otherwise, there is no such path.

Analysis:

- Time Complexity: The time complexity of this algorithm depends on the depth-first search. In the worst case, it can visit all vertices and edges of the graph, leading to a time complexity of O(|V| + |E|), where |V| is the number of vertices, and |E| is the number of edges.
- Space Complexity: The space complexity is determined by the space required for the visited set and the recursive call stack. In the worst case, the visited set can contain all vertices, and the maximum depth of the call stack is the diameter of the graph, leading to a space complexity of O(|V|) for the visited set and O(|V|) for the call stack.

This algorithm efficiently determines whether there is a path from u to w through v in an undirected graph by exploring the graph using depth-first search.

3. **The Island of Sodor is home to a large number of towns and villages, connected by an extensive rail network. Recently, several cases of a deadly contagious disease (either swine flu or zombies; reports are unclear) have been reported in the village of Skarloey. The controller of the Sodor railway plans to close down certain railway stations to prevent the disease from spreading to Tidmouth, his home town. No trains can pass through a closed station. To minimize expense (and public notice), he wants to close down as few stations as possible. However, he cannot close the Skarloey station, because that would expose him to the disease, and he cannot close the Tidmouth station, because then he couldn't visit his favorite pub. Describe and analyze an algorithm to find the minimum number of stations that must be closed to block all rail travel from Skarloey to Tidmouth. The Sodor rail network is represented by an undirected graph, with a vertex for each station and an edge for each rail connection between two stations. Two special vertices s and t represent the stations in Skarloey and Tidmouth.**

Algorithm:

1. Convert Undirected Graph to Directed Graph:
   - Represent each undirected edge as two directed edges.
   - For every edge between stations A and B in the original graph, add one edge from A to B and another from B to A in the directed graph.

2. Find Max Flow:
    - Utilize algorithms such as Ford-Fulkerson or Edmonds-Karp to find the maximum flow in the network.
    - Start with an initial flow and augment it until further augmentation is not possible, obtaining the maximum flow.

3. Find Minimum Cut:
    - Use the Ford-Fulkerson algorithm to find the minimum cut.
    - Conduct a Depth-First Search (DFS) from the source ‘s’ in the residual graph.
    - Mark all reachable vertices. The set of edges where one endpoint is reachable and the other is not comprises the minimum cut.

4. Identify Stations to Close:
    - The number of stations to be closed equals the number of vertices in the minimum cut, excluding ‘s’ and ‘t’ (Skarloey and Tidmouth stations). These cannot be closed.

5. Special Consideration:
    - If there exists a direct edge between Skarloey and Tidmouth, it must be removed before calculating the minimum number of stations to close.

Analysis:

Time Complexity:

- For Edmonds-Karp algorithm: O(E^2 * V) where E is the number of edges and V is the number of vertices in the graph.
- For Ford-Fulkerson algorithm: O(max_flow * E) where E is the number of edges and max_flow is the maximum flow in the network.