Problem 1. (10 points)

1. Given a flow network \( G = (V, E) \), we define an edge \( e \in E \) to be *upward critical* if increasing the capacity of \( e \) increases the value of the maximum flow. Does every network have an upward-critical edge? Describe an algorithm for identifying all upward-critical edges in \( G \), and briefly justify its correctness. The worst-case complexity of your algorithm should be substantially better than that of solving \( m \) maximum-flow problems from scratch.

2. Given a flow network \( G = (V, E) \), we define an edge \( e \in E \) to be *downward critical* if decreasing the capacity of \( e \) (by any amount) decreases the value of the maximum flow. Is the set of upward-critical edges the same as the set of downward-critical edges? If not, describe an efficient algorithm for identifying all downward critical edges in \( G \), briefly justify its correctness, and analyze its worst-case time complexity.

Answer:

1. Not every network has an upward-critical edge. Consider the network \( G = (s, v, t, (s, v), (v, t)) \), with the capacity of \((s, v)\) and \((v, t)\) equal to 1. Increasing the capacity of either edge does not increase the maximum flow. Here is Algorithm A. Consider that if we increase the capacity of an edge \((u, v)\) by a constant integer value \( C \), the flow can be updated (by finding at most \( C \) augmenting paths, each of which increases the value of the flow by at least one unit) in \( O(m) \) time since \( C \) is a constant. We can use this simple update algorithm to check whether an edge in \( G \) is upward critical, for a total complexity of \( O(m^2) \). Alternatively, one can observe that an edge is upwards critical if and only if it is contained in every minimal cut. Here is algorithm B. 1) Compute a max flow \( f \) in the network \( G \). 2) Let \( S \) be the set of all vertices reachable from \( s \) in the residual graph \( G_f \). (We can find such vertices using DFS) 3) Let \( T \) be the set of all vertices that can reach \( t \) in \( G_f \). 4) Any edge \((u, v)\) such that \( u \in S \) and \( v \in T \) is an upward-critical edge. If there is only one min-cut, then these edges are the ones on the cut, if there is more than one min-cut, then \((u, v)\) must belong to every min-cut. This algorithm is correct because increasing the capacity of any edge \((u, v)\), s.t. \( u \in S, v \in T \) results in an augmenting path, so it is upward-critical. Furthermore, every upward-critical edge has this property, so the set of edges found by the algorithm is equal to the set of upward-critical edges. The running time of the algorithm is dominated by the one calculation of the maximum flow, which is \( O(m|f^*|) \) (FF) or \( O(nm^2) \) (EK2).

2. The set of downward-critical edges is not the same as the set of upward-critical edges. In the example in part (a), each of the edges is downward-critical, whereas none are upward-critical. To compute the set of downward-critical edges, consider the following lemma.
Lemma. Let \( f \) be a max-flow for \( G \), and let \((u, v)\) be any edge in \( G \). Consider the residual network \( G_f \). Then the set of edges \((u, v)\) such that there is no path from \( u \) to \( v \) in \( G_f \) is exactly the set of downward-critical edges.

Proof. If there is no path from \( u \) to \( v \), then the edge \((u, v)\) is saturated. Therefore, \( f((u,v)) = c((u,v)) \). Say we decrement the capacity by \( \delta \). Then if we are to keep the max flow at its current value, the \( \delta \) units of flow over capacity must be rerouted. But note that we cannot reroute the flow because there is no path from \( u \) to \( v \). Therefore, \((u,v)\) is a downward-critical edge. Now suppose there is a path from \( u \) to \( v \), with minimum capacity \( \delta \). If we increase the capacity of \((u,v)\) by \( \delta \), we can reroute the flow through the path from \( u \) to \( v \), so that the max-flow would remain the same. So \((u,v)\) is not a downward-critical edge.

The algorithm then is to compute all pairs \((u,v)\) such that \((u,v)\) is an edge in the graph, but there is no path from \( u \) to \( v \) in the residual graph \( G_f \). This can be done using the transitive closure of the graph. The running time of this algorithm is just the time to compute the max flow and then the running time transitive closure.

Problem 2. (10 points)
You are given a flow network \( G \) with source \( s \), sink \( t \) and non-negative edge capacities. Given polynomial-time algorithm to determine whether \( G \) has a unique minimum cut. Analyze the time complexity of your solution.

Answer: Let’s call \( E' \) the set of edges that belong to a minimum cut of \( G \). If the minimum cut is not unique, then there exists another minimum cut composed of edges in \( E'' \), where \( E'' \neq E' \). We provide two possible algorithms.

1. Compute the min cut of network \( G \) by any max-flow min-cut algorithm e.g. Ford-Fulkerson algorithm to get a min cut \( C = (A,B) \) with capacity \( c^* \). Let \( F \) be the set of directed edges from \( A \) to \( B \). For each edge \( e \in F \), increase the capacity of \( e \) by 1, recompute the min cut, and let capacity of the cut be \( c^{e}_{min} \). If for all edges \( e \in F \), \( c^{e}_{min} = c^* + 1 \), then \( (A,B) \) is the unique minimum cut. Since \( |F| = O(m) \), the total running time is \( O(m^2n) \).

2. Given a cut defined by \((A, B)\), denoted the capacity of the cut to be \( c_{A,B} \), and let \( E_{A,B} \) be the set of directed edges from \( A \) to \( B \). Given a max s-t flow \( f \) in the network \( G \) we can construct the residual graph \( G_f \) and perform BFS or DFS to determine the set \( A \) of all nodes we can reach from \( s \) and the set \( B \) of all the nodes that can reach \( t \). If \( A \cup B = V \) then \( G \) has a unique min cut. The total running time is \( O(mn) \) as DFS or BFS just takes \( O(m) \) time.

Problem 3. (10 points)
Suppose that you wish to find, among all minimum cuts in a flow network \( G \) with integral capacities, one that contains the smallest number of edges. Show how to modify the capacities
of $G$ to create a new flow network $G'$ in which any minimum cut in $G'$ is a minimum cut with the smallest number of edges in $G$.

**Answer:**

Suppose that the flow network contains $|E|$ edges. We modify all of the capacities by taking any edge that has a positive capacity and increasing its capacity by $\frac{1}{|E|+1}$. Doing this modification can’t get us a set of edges for a min cut that isn’t also a min cut for the unmodified graph because the difference between the value of the min cut and the next lowest cut value was at least one because all edge weights were integers. This means that the new min cut value is going to be at most the original plus $\frac{|E|}{|E|+1} < 1$. Since this value is more than the second smallest valued cut in the original flow network, we know that the choice of cuts we make in the new flow network $G'$ is also a minimum cut in $G$. Lastly, since we added a small constant amount to the value of each edge, our minimum cut would have the smallest possible number of edges, otherwise one with fewer would have a smaller value.