Graph Algorithms

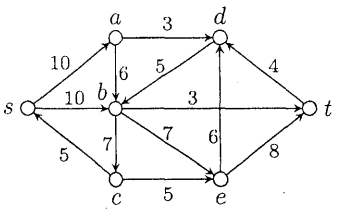
Mid-semester Examination

MTech (CS) II Year (2024-2025) Date: 25-02-2025
Time: 2 hours Marks: 50

Provide proofs for correctness and run-time complexity for all algorithms. Statements already proven in class can be assumed to be true without proof. The symbols n and m always denote the number of vertices and edges respectively of the graph under consideration.

1. A *directed acyclic graph (DAG)* is a directed graph that does not contain any directed cycles.
   (a) Construct an efficient algorithm that given a DAG having weights on its edges as input, computes a path having maximum weight in it. [5]
   (b) Construct an $O(nm)$ time algorithm that computes the all pairs shortest paths matrix for any input DAG having weights on its edges. [10]

2. Show how a maximum flow in the flow network given below will be computed by *any one* of the following algorithms: (a) the Edmonds-Karp algorithm, (b) the Dinitz algorithm, (c) the Malhotra-Kumar-Maheshwari algorithm, or (d) a push-relabel algorithm. Draw the residual graph at each stage of the algorithm. [10]



3. Let G be a directed graph with vertex set $V(G) = \{1, 2, \ldots, n\}$. Construct an $O(n + m)$ time algorithm which when given an adjacency list representation of G that stores a list of out-neighbours for each vertex, constructs another adjacency list representation of G of the same type in which the list of out-neighbours of each vertex is a list sorted in the ascending order. [10]

4. Prove that in any flow network, there is always a maximum value flow in which no directed cycle in the flow network has non-zero flow on every edge. [5]

5. Consider a directed graph in which there is a non-negative real number associated with each edge, called its "capacity", and a real number associated with each vertex, called its "demand/supply". A "flow" in this network is an assignment of non-negative real numbers to the edges such that the flow in each edge is at most the capacity of the edge, and the net out-flow of every vertex is equal to its demand/supply value. Construct an algorithm that given such a network, decides whether there is a valid flow in it. [10]