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.

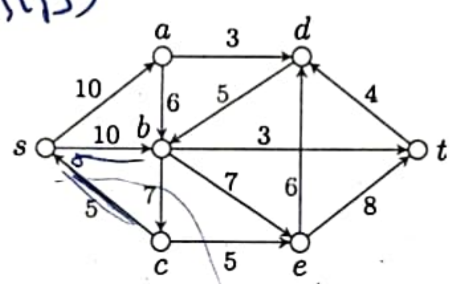
1. A *directed acyclic graph (DAG)* is a directed graph that does not contain any directed cycles.

   (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]

$i][j] = \min(A[i][j], A[i][k] + A[k][j])$



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]

Graph Algorithms

End-semester Examination

MTech (CS) II Year (2024-2025)
Time: 2 hours

Date: 29-04-2025
Marks: 50

Answer all questions. Provide proofs for correctness and run-time complexity for all algorithms.

1. Let G be an edge-weighted directed graph containing no negative weight cycles. For $u, v \in V(G)$, let $dist(u, v)$ denote the weight of a shortest path (a path having least weight) in G from u to v. Show that for vertices $x, y, z \in V(G)$, $dist(x, z) \leq dist(x, y) + dist(y, z)$. [5]

2. Consider the following problem:

Weighted Independent Set	
Input:	A graph G, and an assignment of weights $w : V(G) \to \mathbb{R}^{\geq 0}$ to the vertices of G.
Output:	An independent set of maximum possible weight in G. (Here, the weight $w(S)$ of an independent set S in G is defined as $w(S) = \sum_{u \in S} w(u)$.)

   Show that Weighted Independent Set is polynomial time solvable on bipartite graphs. [5]

3. Given two flows $f_1 : E(G) \to \mathbb{R}^{\geq 0}$ and $f_2 : E(G) \to \mathbb{R}^{\geq 0}$ in a flow network G, we define their sum, denoted as $f_1 + f_2$ to be the mapping $f : E(G) \to \mathbb{R}^{\geq 0}$ such that for each $e \in E(G)$, $f(e) = f_1(e) + f_2(e)$. Let G be a flow network, f_1, f_2 be two flows in G, and $f = f_1 + f_2$ as above. Then prove that:

   (a) f is a valid flow in G if and only if $f(e) \leq c(e)$ for every edge $e \in E(G)$ (here, $c(e)$ denotes the capacity of the edge e in the given flow network), [5]

   (b) if f is a valid flow in G, then $|f| = |f_1| + |f_2|$. [5]

4. Prove that the following problems are NP-hard:

   (a) [5]

Clique	
Input:	A graph G, and an integer k
Output:	"Yes", if G contains a set of vertices $S \subseteq V(G)$ such that $\lvert S \rvert \geq k$ and for every pair of distinct vertices $u, v \in S$, we have $uv \in E(G)$, "No", otherwise

A

[5]

(b)

BIPARTITE DOMINATING SET

Input: A bipartite graph having partite sets A and B, and an integer k

Output: A set $S \subseteq A$ of minimum possible size such that every vertex in B is adjacent to at least one vertex in S.

A matching M in a graph G is said to be *uniquely restricted* if no other matching in G has the same set of matched vertices as M. Prove that a matching M in a graph G is uniquely restricted if and only if there is no alternating cycle in G with respect to M. (An alternating cycle is a cycle in which for every pair of consecutive edges on the cycle, one belongs to M.) [10]

Consider the following optimization problem.

MAXIMUM ACYCLIC SUBGRAPH

Input: A directed graph G.

Output: A subgraph of G containing no directed cycles and having the maximum possible number of edges.

Construct a polynomial-time 2-approximation algorithm for MAXIMUM ACYCLIC SUBGRAPH. [10]