Algorithms

Assignment: Graphs II

Name: .................................................................
Id: .................................................................

Grade

Good Luck!

Diagram of a graph with nodes A, B, C, D, E, F, G, H, I, J connected by edges.
1. (i) Draw the BFS tree for the Petersen graph: Start with A and assume that the vertices are ordered: A, B, C, D, E, F, G, H, I, J. Classify all the 15 edges in the BFS-tree as tree, back, forward, or cross edges.

(ii) Draw another BFS tree for the Petersen graph: Start with J and assume that the vertices are ordered: J, I, H, G, F, E, D, C, B, A. Classify all the 15 edges in the BFS-tree as tree, back, forward, or cross edges.

(iii) The Petersen graph is very symmetric. Support this claim with your trees from the previous two parts.
2. (i) Draw the DFS tree for the Petersen graph: Start with $A$ and assume that the vertices are ordered: $A, B, C, D, E, F, G, H, I, J$. Classify all the 15 edges in the DFS-tree as tree, back, forward, or cross edges.

(ii) Draw another DFS tree for the Petersen graph: Start with $J$ and assume that the vertices are ordered: $J, I, H, G, F, E, D, C, B, A$. Classify all the 15 edges in the BFS-tree as tree, back, forward, or cross edges.

(iii) The Petersen graph is very symmetric. Support this claim with your trees from the previous two parts.
3. Consider the following weighted Petersen graph:

(a) Find a **minimum** weighted spanning tree for this weighted Petersen graph.

(b) Find a **maximum** weighted spanning tree for this weighted Petersen graph.
4. Let $G$ be a weighted graph with $n$ vertices and $m$ edges.

   (a) Design an efficient algorithm that finds a **maximum weighted spanning tree** in $G$.
   What is the complexity of your algorithm as a function of $n$ and $m$? Explain.

   (b) Design an efficient algorithm that finds a **minimum** weighted set of edges that intersect all the cycles $G$. That is, for any cycle of edges in $G$, at least one of the edges belongs to this set.
   What is the complexity of your algorithm as a function of $n$ and $m$? Explain.
5. The cycle graph $C_n$ is a graph that has one cycle containing all $n$ vertices. Describe an efficient algorithm that finds a minimum weighted spanning tree in $C_n$. The algorithm should be more efficient than the algorithms for general graphs. What is the complexity of your algorithm as a function of $n$? Explain.
6. Let \( G \) be a directed graph represented by the adjacency matrix \( A \).

- In directed graphs: \( A(u, v) = 1 \) if \( (u \rightarrow v) \) exists and \( A(u, v) = 0 \) otherwise.
- A tournament is a directed graph such that for any pair \( u \neq v \) exactly one of \( (u \rightarrow v) \) or \( (v \rightarrow u) \) exists.
- An acyclic tournament is a tournament with no cycles.
- A source is a vertex whose in-degree is zero. A destination is a vertex whose out-degree is zero.

(a) Describe an algorithm that checks if a directed graph \( G \) is a tournament. What is the complexity of your algorithm? Explain.
(b) Prove that in an acyclic tournament there is exactly one source and one destination.
Describe an efficient algorithm ($O(n)$-time) to find the source vertex of an acyclic tournament.
7. Let $G$ be a graph with $n$ vertices and $m$ edges. Describe a greedy algorithm that finds a set $S$ of vertices with the following two properties:

- $S$ is independent: There is no edge between any pair of vertices of $S$.
- $S$ is dominating: Every vertex that is not in $S$ has at least one neighbor in $S$.

What is the complexity of your algorithm if the graph is represented by adjacency lists and what is the complexity of your algorithm if the graph is represented by an adjacency matrix? Justify the correctness of your algorithm and justify your complexity claims.
8. A leaf in a graph $G$ (not necessarily a tree) is a vertex that has exactly one neighbor.

Let $G$ be a graph with $n$ vertices and $m$ edges. Describe an efficient algorithm that counts the number of leaves in $G$.

What is the complexity of your algorithm if the graph is represented by adjacency lists and what is the complexity of your algorithm if the graph is represented by an adjacency matrix?

Justify the correctness of your algorithm and justify your complexity claims.