1: The First Problem

SOLUTION:

Algorithm:
Because the graph is dense, use adjacency matrix to implement Dijkstras shortest path algorithm. Initialize the adjacency matrix with all "Infinite" except for the starting node itself, which equals to 0. Update with Dijkstras algorithm.

Correctness: This is just another way to implement Dijkstras shortest path algorithm. The algorithm itself is proven to be correct.

Running time: The runtime of this algorithm is $O(|V|^2)$.

2: The Second Problem

SOLUTION:
(a.)

Algorithm: Initialize all distances as infinite and distance to source as 0. Find a topological sorting of the graph.and process all vertices one by one in topological order. For every vertex being processed, we update distances of its adjacent using distance of current vertex with Dijkstras algorithm. If $d(s,v) > d(s,u) + w[u,v]$, update $d(s,v) = d(s,u) + w[u,v]$

(b.)

Algorithm:
Find a topological ordering of graph G. For each vertex v, iterate through its incoming neighbors and add the weight of the edge between v and the neighbor to the maximum length among the neighbors. The number is equal to zero if v does not have any incoming neighbor. Store this number at every vertex. When you are done with labelling, find the vertex with the highest value and trace it backwards. At each step, choose the outgoing neighbor with the highest number stored until you make it all the way back to s.

Correctness: You are basically recording the longest path to a certain set of vertices at every step. When you are done, the one with the highest number is naturally the one on the other end of the longest path.

Running time: This algorithm runs linear time.

3: The Third Problem

SOLUTION:

Algorithm: For any vertex v in the tree, check its neighbors. Let an arbitrary neighbor of v be u. The inequality $d(s,u) \geq d(s,v) + weightw(u,v)$. If a vertex in the tree fail to meet this requirement, the tree T is not the shortest path tree. You are essentially checking each edge once, so the runtime is $O(|E|)$.

Correctness: If v=s, d(s,s) is zero, good. Otherwise, the vertices in the tree has to satisfy the property of shortest path.

Running time: $O(|E|)$, because you are checking all edges once

4: The Fourth Problem

SOLUTION:
Because it is a directed graph, for any edge on any path from s to t (excluding the edge has a node being either s or t), w(u, v)' = w(u, v) - f(u) + f(v). The edge before edge (u,v) = w(a, u)' = w(a, u) - f(a) + f(u). As we can see, for all the edges in the middle of the path, namely not the ones that are directly connected to s or t, the f(u)s and f(v)s get canceled out because it is a directed graph and the end node of an edge is the start node the the edge connected to it and because the signs of the two terms are the exact opposite, they cancel out. So the sum of all the weights of a path from s to t is W(s,t) - f(s) +f(t). f(s) and f(t) are the same for all path going from s to t. The ranking of all the paths from s to t does not change because the constant terms. Therefore, the shortest path from s to t of G is still the shortest path of G'.