If the complexity of the required algorithm is not given, you shall find an algorithm with as good complexity as you can. If your algorithm would be far from optimal, the grade will be decreased.

1(3). A graph can have several shortest paths between some vertices. Construct a linear-time algorithm that finds the number of vertices that lie on at least one shortest path from s to t in an undirected graph with unit edge weights.

2(2+2+2). Consider the following modification of Dijkstra's algorithm. During initialization, only vertex s is in the priority queue. Vertex v is added to the priority queue if, as a result of $\mathrm{Relax}(u, v)$ relaxation, the distance to v has changed, and v was not in the queue at that moment. The remaining steps of the algorithm remained unchanged.

1. Prove the correctness of the modified algorithm.

2. Prove that the modified algorithm works correctly even if there are edges of negative weight, but there is no cycle of negative weight. Estimate the running time of the algorithm on graphs of this kind and compare it with the running time of the Bellman-Ford algorithm.

3. Modify the algorithm so that it verifies existence of cycles of negative weight (and terminates with an error if such a cycle exist).

3 (3). The directed graph has edges of negative weight, but no cycles with negative weight. Construct an algorithm that finds for a given vertex the vertex **from which** it is the maximum distance away. Prove the correctness and estimate the complexity.

4(4). The light weight of the path p is the sum of the weights of the edges of p except for the edge of the maximum weight (when traveling between cities on a network of toll roads, you do not have to pay to travel on one road). Construct an algorithm that, given a weighted directed graph with positive edge weights, finds paths of the minimum lightweight weight from s to all other vertices. Prove the correctness and estimate the complexity.

5 (3). A directed weighted graph has exactly one edge $(u \to v)$ of negative weight. Describe an efficient algorithm for finding the shortest path between a given pair of vertices (a, b) "— problem input: weight matrix and vertices a and b. Prove the correctness and estimate the complexity.

6 (3). The following statements may or may not be correct. In each case, either prove it (if it is correct) or give a counterexample (if it isn't correct). Always assume that the graph $G = (V, E)$ is undirected. Do not assume that edge weights are distinct unless this is specifically stated.

a) If the lightest edge in a graph is unique, then it must be part of every MST.

b) If e is part of some MST of G, then it must be a lightest edge across some cut of G.

c) The shortest path between two nodes is necessarily part of some MST.

7 (4). Let T be an MST of graph G. Given a connected subgraph H of G, show that $T \cap H$ is contained in some MST of H.

8 (4). The input of the problem is an undirected weighted graph $G(V, E)$ and a subset of vertices $U \subseteq V$. It is necessary to construct a spanning tree that is minimal (by weight) among trees in which all vertices of U are leaves (but there may be other leaves), or find that there are no such spanning trees. Construct an algorithm that solves the problem in $O(|E| \log |V|)$. Note that the tree you are looking for may not be a minimum spanning tree.