1 What is the minimum sum of edge costs of a spanning tree of the following graph? $\{(a, b, 1), (a, c, 4), (a, d, 3), (b, d, 2), (a, c, 4), (c, d, 5)\}$

1. 6
2. 7
3. 8
4. 9

(2) is the correct answer. To see this we can apply one of the MST algorithm described in the text. Or since the graph is small, enumerate all spanning trees.

2 Which of the following running times best describes a straightforward implementations of Prim's minimum spanning tree algorithm for graphs in adjaceny-list representation? As usual, n and m denote the number of vertices and edges, respectively, of the input graph.

1. $O(m + n)$
2. $O(m \cdot logn)$
3. $O(n^2)$
4. $O(m \cdot n)$

(4) is the correct answer using a brute force algorithm. As the book describes the algorithm goes through $n - 1$ iterations and then checks up to m edges to find the cheapest edge for every iteration. This is order of $m \cdot n$ and hence $O(m \cdot n)$.

3 In figure 15.5, suppose the vertex x is extracted and moved to the set X. What should be the new values of y and $z's$ keys, respectively?

1. 1 and 2
2. 2 and 1
3. 5 and $+\infty$
4. $+\infty$ and $+\infty$

4 Which of the following running times best describes a straightforward implementation of Kruskal's MST algorithm for graphs in adjacency-list representation? As usual, n and m denote the number of vertices and edges, respectively, of the input graph.

1. $O(m \cdot logn)$
2. $O(n^2)$
3. $O(m \cdot n)$
4. $O(m^2)$

(3) is the correct answer. We go through m edges to obtains all n vertices in the spanning tree. The total runtime is order of $m \cdot n$ hence $O(m \cdot n)$.

5 What's running time of the FIND operation, as a function of the number n of objects?

1. $O(1)$
2. $O(logn)$
3. $O(n)$
4. Not enough information to answer

(3) is the correct answer. In the worst case the depth could be $n-1$ and hence $O(n)$.

6 Suppose we arbitrarily chose which root to promote. What's the running time of the FIND operation as a function of the number n of objects?

1. $O(1)$
2. $O(logn)$
3. $O(n)$
4. Not enough information to answer

(3) is the correct answer.

7 With the implementation of UNION above, what's the running time of the FIND operation; as a function of the number n of objects?

1. $O(1)$
2. $O(logn)$
3. $O(n)$
4. Not enough information to answer

(2) is the correct answer.

8 Consider an undirected graph $G = (V, E)$ in which every edge $e \in E$ has a distinct and nonnegative cost. Let T be an MST and P a shortest path from some vertex s to some other vertex t. Now suppose the cost of every edge e of G is increased by 1 and becomes $c_e + 1$. Call this new graph G'. Which of the following is true about G'.

1. T must be an MST and P must be a shortest s-t path.
2. T must be an MST and P may not be a shortest s-t path.
3. T may not be an MST but P must be a shortest s-t path.
4. T may not be an MST and P may not be a shortest s-t path.

9 Consider the following algorithm that attempts to compute an MST of a connected undirected graph $G = (V, E)$ with distinct edge costs by running Kruskal's algorithm "in reverse". Which of the following statements is true?

1. The output of the algorithm will never have a cycle, but it may not be connected.
2. The output of the algorithm will always be connected,, but it might have cycles.
3. The algorithm always outputs a spanning tree, but it might not be a MST.
4. The algorithm always outputs an MST.

10 Which of the following problems reduce, in a straightforward way, to the minimimum spanning tree problem?

1. The maximum-cost spanning tree problem. That is, among all spannign trees T of a connected graph with edge costs, compute the one with the maximum-possible sum $\sum_{e \in T} c_e$ of edge costs.

2. The minimum-product spanning tree problem. That is among all spanning trees T of a connected graph with strictly positive edge costs, compute one with the minimum-possible product $\Pi_{e \in T} c_e$ of edge costs.

3. The single-shortest path problem. In this problem, the input comprises a connected undirected graph $G = (V, E)$, a nonnegative length l_e for each $e \in E$, and a designated starting vertex $s \in V$. The required output is, for every possible destination $v \in V$, the minimum total length of a path from s to v.

4. Given a connected undirected graph $G = (V, E)$ with positive edge costs, compute a minimum-cost set $F \subseteq E$ of edges such that the graph $(V, E - F)$ is acyclic.

11 Prove the converse of Theorem 15.6: If T is an MST of a graph with real-valued edge costs, every edge of T satisfies the minimum bottleneck property.

12 Prove the correctness of Prim's and Kruskal's algorithm in full generality, for graphs in which edge's cost need not be distinct

13 Prove that in a connected undirected graph with distinct edge costs, there is a unique MST.

14 An alternative approach to proving the correctness of Prim's and Kruskal's algorithms is to use what's call the *Cut Property* of MSTS. Assume throughout this problem that edges' costs are distinct. A cut of an undirected graph $G = (V, E)$ is a partition of its vertex set V into two non-empty sets, A and B. An edge of G that crossed the cut (A, B) if it has one endpoint in each of A and B. In other words, one way to justify an algorithm's inclusion of an edge e in its solution is to produce a cut of G for which e is the cheapest crossing edge.

1. Prove the Cut Property.

2. Use the Cut Property to prove that Prim's algorithm is correct.

3. Repeat for Kruskal's algorithm