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 $+\infty$

(2) is the correct answer.

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 Find the all the edges in (1) that satisfies the *Minimum Bottleneck Property*

It is all the edges that make up the MST from the graph. Every edge in the MST have the *Minimum Bottleneck* property.

6 Prove that For every connected graph $G = (V, E)$ and real-valued edge costs, every edge chosen by the Prim's algorithm satisfies the MBP.

Consider an edge $(v^\star, w^\star)$ chosen in an iteration of Prim's such that $v^\star \in X$ and $w^\star \in V - X$, in others $v^\star$ has already been processed by $w^\star$ has not. By the algorithm's greedy rule the cost of the edge denoted $c_{v^\star w^\star} \leq c_{xy}$ where $x \in X$ and $y \in V - X$. To show that the edge $c_{v^\star w^\star}$ satisfies the *MBP* property consider any $v^\star - w^\star$ path. Since $v^\star \in X$ and $w^\star \in V - X$ there must be some edge say $(x, y) \in E$ where $x \in X$ and $y \in V - X$ such that we cross from X to $V - X$. The bottleneck of the path P is at least c_{xy} but then by the earlier equation c_{xy} is at least $c_{v^\star w^\star}$. Since we assumed the the path P was arbitrary by definition $(v^\star, w^\star)$ satisfies the *MBP* property.

7 Let $G = (V, E)$ be an undirected graph and $v, w \in V$ two distinct vertices such that $(v, w) \in E$

1. If v and w are in the same connected components of G, adding the edge (v, w) to G creates at least one new cycle and does not change the number of connected components.
2. If v and w are in different connected components of G, adding the edge (v, w) to G does not create any new cycles and decreases the number of connected components by 1.

For part (1) If v and w are in the same connected component then there already exists a $v - w$ path P in G. After the edge addition, $P \cup \{v, w\}$ forms a new cycle. Since we did not connect two different vertices in two different connected components the number of connected components remains exactly the same.
For part (2) If v and w are not in the same connected component then there does not already exist a $v - w$ path P in G, we could not have formed a cycle. Let S_1 and S_2 denote the distinct connected components of G that contains v and w respectively. The edge addition forms a single connected components $S_1 \cup S_2$ and decreases the number of connected components by 1.

8 Prove that every spanning tree of an n-vertex graph has $n - 1$ edges.

Let T be a spanning tree of a graph $G = (V, E)$ with n vertices. Start from the empty graph with vertex set V and continously add the edges of T one by one. Because T has no cycles every edge addition does not create a cycle but instead decreases the number of connected components by one. Since the graph is connected the number of connected components is 1, this implies that $n - 1$ edges were added.

9 Let $G = (V, E)$ be a graph and $T \subseteq E$ a subset of $n - 1$ edges, where $n = |V|$. The graph (V, T) is connected if and only if it contains no cycles.

Reprising the edge addition process from the previous problem, if each of the $n-$ 1 edge addition are fused typed then that means that the number of connected components of the graph will be one. There could not be any cycles in this graph. Suppose there was a cycle, then that means that one of the $n - 1$ edge additions were were of the cycle type. This means that we could have at least 2 connected components in the graph, contradicting the graph being connected. This shows that (V, T) is connected iff (V, T) has no cycles.

10 For every connected input graph, the *Prim* algorithm outputs a spanning tree.

The vertices of X form one connected component and the vertices of $V - X$ all form their own separate connected component. Each of the $n - 1$ edge additions involves a connected a vertex $v \in X$ to a vertex $v \in V - X$ and hence is of the type fusion. The final result is a spanning tree.

11 Prove that if every edge in a spanning tree satisfies the MBP then that implies it must be a MST.

We proceed by contradiction. Let T be a spanning tree in which every edge satisfies the MBP, and suppose there exists a spanning tree $T^\star$ has a strictly smaller sum of edge costs. Since the total cost of T and $T^\star$ are different there must be an edge $e_1 = (v_1, w_1)$ such that is it in T and not in $T^\star$. Adding this edge into $T^\star$ creates a cycle C that contains e_1. Since e_1 satisfies the MBP property that is an edge $e_2 = (x, y)$ in the $v - w$ path $C - \{e_1\}$ with cost at least c_{vw}. But since all the edges are distinct $c_{vw} < c_{xy}$. We can thus derive a spanning tree $T^{\star\star}$ that is even better than $T^\star$ by removing the edge e_2 and adding the edge e_1 into $T^\star$. This contradicts the optimality of $T^\star$ and hence completes our proof.

12 Use all of your previous work to proof *Prim's* algorithm.

(10) proves that the output of *Prim's* algorithm is a spanning tree. (6) proves that every edge of *Prim's* algorithm output satifies the MBP. (11) proves that any spanning tree that satifies the MBP is also an MST. Thus we conclude that *Prim's* correctly produces an Minimum spanning tree.

13 Let T be the set of edges chosen by Kruskal up to and including the iteration that examines the edge $e = (v, w)$. Then, v and w are in the same connected component of the graph (V, T).

Adding the edge e to the set T is of either type cycle or type fuse. In the first case the vertices v and w are already in the same connected component prior to the examination of the edge e. In the second type we connect v directly to w and fuse their connected components into one.

14 Let P be a $v - w$ path in G, and T the set of edges chosen by *Kruskal* up to and including the laster iteration that examines an edge of P. Prove that v and w are in the same connected component of the graph (V, T).

Denote the edges of P as $(v, x_1), (x_1, x_2), \ldots (x_{p-1}, w)$. By the previous problem after iteration of some edge (x_{i-1}, x_i) the vertices x_{i-1} and x_i are line the same connected component. After all edges have been processed the vertices v and w will belong in the same connected component.

15 Prove that *Kruskal* algorithms outputs a spanning tree.

The algorithms as described in the textbook ensures that the final outputs is acyclic. To prove that the output is also connected, suppose to arbitrary vertices v and w in G. Since the graph is connected there is a $v - w$ path. By the previous problem after the graph has been processed v and w will reside in the same connected component

16 Prove that for every connected graph $G = (V, E)$ and real-valued edge costs, every edge chosen by the *Kruskal* algorithm satisfies the *MBP.*

We prove the contrapositive, that is we prove that *Kruskal* never outputs a edge that does not satifies the MBP. Let $e = (v, w)$ be such an edge, and P a $v - w$ path in G in which ever edge has cost less than c_e. But in the algorithm we search the edges in order of decreasing cost and hence the algorithm processes every edge of P before e. By (14) its endpoint v and w already belong to the same connected component, adding e would result in a cycle. Kruskal would not have included the edge in the output.

17 Use all the previous work to prove the correctness of *Kruskal's* algorithm

(15) proves that the output of *Kruskal's* algorithm is a spanning tree. (16) proves that the output of *Kruskal's* algorithm edges satisfies the MBP property.(11) proves that the MBP property implies the spanning tree is a minium spanning tree. Hence we conclude that *Kruskal's* algorithm outputs a minimum spanning tree.

18 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 $O(n)$.

19 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.

20 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.

21 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.

(2) is the correct answer. Increasing the edge cost of all edges by ones does not alter the MBP edge for any eligible edge in the graph. But P may not be shortest $s - t$ path as could shown by a counterexample.

22 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.

(4) is the correct answer. We can define the edges that *Kruskal* removed as type *cycle* edges. The end result will be a spanning tree. All of the edges that are contained in the spanning tree also satisfies the MBP. The reason being is similar to the (11) exchange argument, we could make a more optimal solution from the most optimal solution which is nonsense. Hence we have a spanning tree where every edge satisfies the MBP from (11) implies that we have a MST. Hence the algorithm is correct.

23 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 spanning 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.

(1) is correct. We can negate the edges of graph G, from this we compute the MST and negate the output of the result.(2) is correct, if we take the logarithm of all the edges we can reduce the problem down to finding the MST. (3) is incorrect, the output of a MST algorithm and a shortest path algorithm such as dijkstra is different. This is because they both have two different objectives functions, MST minimizing the total cost of edges and Dijkstra minimizing the total cost of a path. (4) is correct.

24 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.

Suppose not every edge in the spanning tree T satifies the MBP. Then there exists atleast one edge $e = (v, w)$ such that the path P which denotes a $v - w$ path in G has all edges with cost less than c_e. Removing e from T will result in two connected components S_1 and S_2. There must be an edge $e' = (x, y)$ such that $x \in S_1$ and $y \in S_2$. The edge set $T' = T - \{e\} \cup \{e'\}$ is spanning tree that has lower cost than T, this contradicts T being a MST.

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

Let T be the spanning tree obtained by Prim's algorithm and $T^\star$ be any minimum spanning tree of G. We will prove that the cost $C(T) = C(T^\star)$. If $T = T^\star$ we're done $C(T) = C(T^\star)$. Else we have that $T \neq T^\star$, then $T - T^\star \neq \emptyset$. Let $e = (u, v)$ be an edge from $T - T^\star$. When e was added to T it was the minimum cost edge crossing some cut $(S, V - S)$. Since $T^\star$ is an MST, there must be an edge from u to v in $T^\star$. This path begins in S and ends in $V - S$. So there must be an edge $f = (x, y)$ along the path where $x \in S$ and $y \in V - S$. Since e is a least crossing edge from S to $V - S$ we have that $C(e) \leq C(f)$. Let $T^{\star'} = T^\star - \{f\} \cup \{e\}$. The cost $C(T^{\star'}) = C(T^\star) - C(f) + C(e) \leq C(T^\star)$. But since $T^\star$ is an MST $C(T^{\star'}) \geq C(T)$ hence we conclude $C(T^{\star'}) = C(T^\star)$. Note that $|T - T^{\star'}| = |T - T^\star| - 1$. Therefore if we continued this process we will have converted $T^\star$ to T while preserving the cost $C(T^\star)$. Thus $C(T^\star) = C(T)$. We conclude then that *Prim's* algorithm correctly outputs a Minimum Spanning Tree in graphs in which edge's cost need not be distinct. The proof of Kruskal's is similar.

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

Suppose that there exists at least two *Minimum Spanning Tree's* T_1 and T_2 such that not all the edge weights are the same. Let $e = (x, y)$ be the smallest weight that is only in either in T_1 or T_2. Without loss of generality assume that the edge e is in T_1. Then there must be a path $x - y$ that does not only contain e in T_2. Adding e to T_2 creates a cycle. If all the edges in this cycle were in T_1 then T_1 would have a cycle which it could not hence there must be an edge f that is in T_2 but not in T_1. By definition of e being the smallest weight in T_1 and the fact that all edge costs are distinct f must be larger than e. We could derive an tree $T_3 = T_2 - \{f\} \cup \{e\}$ that has smaller cost than T_2. This contradicts that T_2 was a *Minimum Spanning Tree.* We conclude that in a connected undirected graph with distinct edge costs there is a unique MST.

27 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.

We prove that given any cut in an edge-weighted graph, the crossing edge of minimum weight is in the MST. Let e be the crossing edge of minimum weight and Let T be an MST. Suppose that T does not contain e. Then if we include e in T we would create a cycle of length at least one with another crossing edge say f. Since e is the smallest crossing edge we could create an spanning tree of strictly lower weight by removing f and adding e into T. This contradicts T be a minimum spanning tree.

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

   Prim's algorithm starts from an arbitrary vertices v find the minimum crossing edge e connects to a vertex w crosses it and continues until $V-1$ edges have been used to create the MST. By the cut property any crossing edge is in the MST. The $V-1$ edges then chosen by *Prim* produces a MST

3. Repeat for Kruskal's algorithm

   The algorithm continously chooses the minimum crossing edges that does not create a cycle. By the Cut Property this edge must be in the MST. Kruskal continues until $V-1$ edges are obtained and thus correctly produces a MST.