ANSWERS: CHAPTER 16

Section 16.3 (Spanning trees and the MST problem)

1. Note that both graphs possess Hamiltonian paths, which means we can choose spanning trees which are chains. In the notation of Figure 15.2.3 (see Facit for Chapter 15), an example of a Hamiltonian path in the cube graph is

$$(0,\,0,\,0) \to (1,\,0,\,0) \to (1,\,1,\,0) \to (0,\,1,\,0) \to (0,\,1,\,1) \to (0,\,0,\,1) \to (1,\,0,\,1) \to (1,\,1,\,1).$$

In the notation of Figure 15.2.2 (see Facit for Chapter 15), an example of a Hamiltonian path in Petersen's graph is

$$a \to b \to c \to d \to e \to j \to g \to i \to e \to h.$$

2. See Figure 16.3.2.

3. If we start at the vertex a, say, then Prim's algorithm will choose the following edges in turn - at each step, there is a unique edge of minimal weight available to the algorithm:

Step	Edge chosen	Weight
1	$\{a,\,d\}$	1
2	$\{a,\,b\}$	2
3	$\{b,\,e\}$	1
4	$\{e,\,h\}$	2
5	$\{e,\,f\}$	3
6	$\{f,\,c\}$	1
7	$\{c,\,g\}$	2
8	$\{g,\,j\}$	1
9	$\{j,\,i\}$	1
10	$\{i,\,k\}$	2
Total weight		16

It is the unique spanning tree of weight 16. For note that our spanning tree included every edge in the graph of weight 1 or 2, plus one of the two edges of weight 3. Hence, the only other possibility for a spanning tree of weight 16 would be to exchange the edge $\{e,\,f\}$ for the other edge of weight 3, namely $\{g,\,i\}$. But the latter would create a 3-cycle $g - i - j$.

4. The graph is illustrated in Figure 16.3.4. There are four minimal spanning trees, each of weight 20. We denote them as T_{11}, T_{12}, T_{21}, T_{22}. All four contain the four edges $\{d,\,e\}$, $\{b,\,c\}$, $\{c,\,x\}$, $\{a,\,f\}$. Then we choose either $\{c,\,d\}$ or $\{e,\,x\}$ for the fifth edge, and either $\{a,\,b\}$ or $\{a,\,x\}$ for the sixth edge.

To see that this exhausts all possibilities, divide the vertex set into the two subsets $V_1 = \{b,\, c,\, d,\, e\}$, $V_2 = \{a,\, f\}$. Note that three of the first four edges listed above are the only three edges inside V_1 of weight less than 3. The two possibilities for the fifth edge are those inside V_1 which don't create a cycle when added to the first three. Hence, these are the only two possibilities for a minimal spanning tree on V_1. When we cross to V_2, we'll need at least two more edges to connect up with vertices a and f, and the cheapest way to do this is to use two of the three edges of weight 6, one of which must be the edge $\{a,\, f\}$.

5. This is the same idea as in the proof of Theorem 16.3.3 in Biggs (Theorem 17.x in the lecture notes). The proof is by contradiction. Suppose $w(e) > w(e^*)$. Let U be the subgraph of K got by taking the tree T and replacing the edge e by e^*. Since $w(e) > w(e^*)$, the total weight of U is less than that of T. Since T is assumed to be a minimal spanning tree of K, and U has the same number of edges as T, we'll have a contradiction if we can show that U is also a tree.

So let $x,\, y$ be two vertices of K. It suffices to show there is a unique path in U between x and y. We can show existence and uniqueness separately.

Existence: We know there's a (unique) path in T between x and y. Call this path $\mathcal{P}$. If the path $\mathcal{P}$ doesn't include the edge e, then it's also present in U. If the path does contain e then, since e also belongs to the unique path $\mathcal{Q}$ in T joining the vertices of e^*, if we replace e in the path $\mathcal{P}$ by the rest of the path $\mathcal{Q}$ together with the edge e^* (see Figure 16.3.5), then we'll have a path in U between x and y.

Uniqueness: If there are two paths in U between som pair $x,\, y$ of vertices then concatenating these two paths yields a cycle. Since T is a tree, this cycle can't already have been present in T, in other words e^* must appear somewhere in the cycle. But the edge e is, by definition, an edge along the unique path in T between the endpoints of e^*. Hence e also appears along the cycle. But this is impossible, since e is not present in U.

Section 16.6 (The shortest path problem)

1. The algorithm will add edges and labels in the following sequence. Steps 5 and 6 are interchangeable.

Step	Edge added	Label added
1	$\{v, a\}$	$l(a) := 4$
2	$\{v, c\}$	$l(c) := 6$
3	$\{a, d\}$	$l(d) := 7$
4	$\{d, e\}$	$l(e) := 12$
5	$\{e, b\}$	$l(b) := 14$
6	$\{a, f\}$	$l(f) := 14$
7	$\{b, g\}$	$l(g) := 18$
8	$\{g, h\}$	$l(h) := 24$
9	$\{h, w\}$	$l(w) := 27$

The unique $v \leftrightarrow w$ path in this tree is found by reading backwards from w:

$$v \leftrightarrow a \leftrightarrow d \leftrightarrow e \leftrightarrow b \leftrightarrow g \leftrightarrow h \leftrightarrow w.$$

2. Dijkstra's algorithm will add edges and labels in the following sequence. Steps 2 and 3 are interchangeable.

Step	Edge added	Label added
1	$\{A, D\}$	$l(D) := 3$
2	$\{A, E\}$	$l(E) := 4$
3	$\{D, B\}$	$l(B) := 4$
4	$\{D, C\}$	$l(C) := 6$
5	$\{E, F\}$	$l(F) := 7$

The unique $A \leftrightarrow F$ path in this tree is found by reading backwards from F and is imply $A \leftrightarrow E \leftrightarrow F$.

Review Section 16.7

6. There are two minimal spanning trees, each of total weight 15. Both contain each of the six edges $\{A, B\}$, $\{D, E\}$, $\{A, F\}$, $\{F, G\}$, $\{E, H\}$, $\{C, E\}$. Then for the last edge we can choose either $\{D, F\}$ or $\{G, H\}$.

To see that these are the only two possibilities, first note that we must use at least one edge of weight 4 or more, as otherwise the vertex C would be isolated. If we use at least one edge of weight 4, then the total weight will exceed 15 unless we use all five edges of weight 1 or 2. The subgraph consisting of these five edges has three connected components: $\{A, B, F\}$, $\{D, E, H\}$ and $\{C\}$. We've elready used weight 8 and have to insert two more edges. Hence we must use the edge $\{C, E\}$ to connect up C and then one of the two edges of weight 3 to connect the first two components.

8. The algorithm will add edges and labels in the following sequence.

Step	Edge added	Label added
1	$\{a, d\}$	$l(d) := 1$
2	$\{a, b\}$	$l(b) := 2$
3	$\{b, e\}$	$l(e) := 3$
4	$\{e, h\}$	$l(h) := 5$
5	$\{e, f\}$	$l(f) := 6$
6	$\{f, c\}$	$l(c) := 7$
7	$\{c, g\}$	$l(g) := 9$
8	$\{g, j\}$	$l(j) := 10$
9	$\{j, i\}$	$l(i) := 11$
10	$\{i, k\}$	$l(k) := 13$

The unique $a \leftrightarrow k$ path in this tree is found by reading backwards from k:

$$a \leftrightarrow b \leftrightarrow e \leftrightarrow f \leftrightarrow c \leftrightarrow g \leftrightarrow j \leftrightarrow i \leftrightarrow k.$$

9. For the graph in Figure 16.7.9, Kruskal's algorithm would choose three edges in the order: $\{A, B\}$, $\{C, D\}$, $\{A, D\}$. After the second step we have a disconnected graph. Note that Prim's algorithm would instead choose $\{A, D\}$ before $\{C, D\}$.

10. First we prove that Kruskal's algorithm always produces *some* spanning tree. If the graph G has n vertices then a subgraph is a spanning tree if and only if it consists of $n - 1$ edges and has no cycles. Since Kruskal's algorithm proceeds by adding edges one by one and will always find *some* edge to add as long as there exists such an edge which creates no cycles, the only thing that could possibly go wrong is if at some point, having added say k edges where $1 \leq k < n - 1$, every remaining edge in G is such that its addition would create a cycle. Let's denote by T the subgraph consisting of these k edges. Then either

(i) T is a tree. Since T has less than $n - 1$ edges, it cannot yet span G. Since G is connected, there must therefore be at least one edge left in G which connects T to a vertex not yet covered. Adding this edge will not create a cycle and, in the worst case, it is available to Kruskal's algorithm.

(ii) T is a forest. Since T has less than $n-1$ edges and G is connected, there must be at least one edge in G which connects up two components of T. Adding this edge will not create any cycle and it is also available to Kruskal's algorithm.

This proves that Kruskal's procedure yields *some* spanning tree. The proof that this spanning tree is minimal is completely analogous to the proof for Prim's algorithm, given in both the book and the lecture notes. I leave it to the reader to fill in the details.

17. Applying (i) and (ii) over and over tells us how to construct the Prüfer symbol for a given spanning tree, i.e.: it describes explicitly the map

$$\phi : \{\text{Spanning trees of } K_n\} \to \{\text{Prüfer symbols } (p_1, p_2, \ldots, p_{n-2}),\ 1 \leq p_i \leq n\}.$$

To show that ϕ is a bijection it suffices to describe its inverse, i.e.: to describe how to reconstruct a spanning tree from a Prüfer symbol. Let $(p_1, p_2, \ldots, p_{n-2})$ be a Prüfer symbol. Set $p_{n-1} := n$. For $i = 1, 2, \ldots, n-1$, let b_i be the least vertex not in

$$\{a_i, a_{i+1}, \ldots, a_{n-1}\} \cup \{b_1, b_2, \ldots, b_{i-1}\}.$$

It's actually clear form the description of (i) and (ii) that this procedure is the reverse of ϕ. What we need to prove is that the set $\{\{b_i, a_i\} : i = 1, \ldots, n-1\}$ of edges it produces always forms a spanning tree of K_n. Since we produce $n-1$ edges it suffices to prove that we never get a cycle. The key points here are the following:

(a) By definition, $b_i \notin \{b_1, \ldots, b_{i-1}\}$. Hence, the vertices $b_1, b_2, \ldots, b_{n-1}$ are distinct.

(b) By definition, $b_i \notin \{a_i, a_{i+1}, \ldots, a_{n-1}\}$. Hence if, for some $i \neq j$, the edges $\{a_i, b_i\}$ and $\{a_j, b_j\}$ share a common vertex, then this vertex must be $a_{\min\{i,j\}} = b_{\max\{i,j\}}$.

From (b) it follows that, in any path,

$$\{a_{i_1}, b_{i_1}\} \to \{a_{i_2}, b_{i_2}\} \to \ldots$$

we must always have $b_{i_1} = a_{i_2}$, $b_{i_2} = a_{i_3}$ etc. and $i_1 > i_2 > \ldots$. But then there can't be a cycle of such edges.