

## Solutions to Assignment #6, updated 11/17/08

**2.2.1** The values in a Prüfer code correspond precisely to the degrees of the non-leaves, so (a) There is only one value (repeated  $n - 2$  times) iff there is one vertex of degree  $n$ , and the tree is a star. (b) There are exactly two values iff there are two vertices  $u, v$  that are not leaves, in which case they must be adjacent, and the graph is a “double-star”. (c) All  $n - 2$  values are distinct iff there are  $n - 2$  vertices of degree 2 and the other 2 vertices are leaves, which means that it’s a path graph.

**2.2.3** (This should be straightforward.)

**2.2.10**  $\tau(K_{2,m})$  using the Matrix Tree Theorem: Let  $\{v_1, v_2\}$  and  $\{v_3, \dots, v_{m+2}\}$  be the partite sets of  $K_{2,m}$ . Let  $Q$  be the matrix  $Q$  from the Matrix Tree Theorem. Delete the first row and column of  $Q$  to get  $Q^*$ . The first row and column of  $Q^*$  are each  $\langle m, -1, \dots, -1 \rangle$ , and the rest of it is  $2I_m$ . Expanding along the first row, the first term is  $m \det 2I_m = m2^m$ . Each other term has a row of the form  $\langle -1, 0, \dots, 0 \rangle$ ; expand along it to see that each of these terms contributes  $-2^{m-1}$  to  $\det Q^*$ . Altogether these terms contribute  $-m2^{m-1}$  since there are  $m$  of them. Thus,  $\det Q^* = m2^m - m2^{m-1} = m2^{m-1}$ .

*Alternatively, a direct argument:* Suppose that  $T$  is a spanning tree of  $K_{2,m}$ .  $N_T(v_1)$  must intersect  $N_T(v_2)$  since  $T$  is connected; since  $T$  is acyclic,  $v_1$  and  $v_2$  have exactly one common neighbor. Every other vertex in  $\{v_3, \dots, v_{m+2}\}$  is a leaf in  $T$ . There are  $m$  ways to pick the common neighbor of  $v_1$  and  $v_2$ , then choose whether each leaf is adjacent to  $v_1$  or  $v_2$ .

*The number of isomorphism classes of spanning trees:* Each spanning tree consists of a path of length 2, with  $m - 1$  leaves attached to the endpoints. The leaves are partitioned into two sets  $S, S'$ , and  $\{|S|, |S'|\}$  determines the isomorphism class. The smaller set has size between 0 and  $\lfloor (m - 1)/2 \rfloor$ , so the number of possibilities is  $1 + \lfloor (m - 1)/2 \rfloor = \lfloor (m + 1)/2 \rfloor = \lceil m/2 \rceil$ .

**2.3.2** Disproof: Consider a cycle of length at least 3, where every edge has the same weight. Then  $T$  is a spanning path; let  $u, v$  be its endpoints.

**2.3.3** Use Kruskal’s algorithm to get edges 12,23,45,25 in that order, for a total cost of 21.

**2.3.5** You might use Dijkstra’s algorithm five times.

**2.3.8** For a tree  $T$ , let  $W(T)$  be the edge-weights listed in nondecreasing order. We will show that  $W(T) = W(T')$  for any minimum-weight spanning trees  $T$  and  $T'$ . (This is stronger than what the problem asks, since it could be possible that there exists a minimum-weight spanning tree  $T$  which will never be produced by Kruskal’s algorithm, no matter how ties are broken. The fact that this is *not* possible can be proved using the result we’re about to prove.)

Suppose that  $W(T) \neq W(T')$ , and choose  $T, T'$  such that  $E(T) \cap E(T')$  is maximized. Choose  $e \in E(T) - E(T')$  and  $f \in E(T') - E(T)$  of smallest weight; we may assume that  $w(e) \leq w(f)$ . By Theorem 2.1.7, there exists  $e' \in E(T') - E(T)$  such that  $T' + e - e'$  is a spanning tree. Since  $T'$  is a min-weight spanning tree,  $w(e') \leq w(e)$ . By the choice of  $e$ ,  $w(e') \geq w(f) \geq w(e)$ . Then  $T' + e - e'$  is a minimum-weight spanning tree, and  $W(T' + e - e') = W(T')$ .  $E(T' + e - e') \cap E(T)$  is larger than  $E(T) \cap E(T')$ , a contradiction.

*Alternatively:* For any fixed weight, Kruskal’s considers the edges of that weight in some order (in any order). Fix a weight  $w$ . After Kruskal’s algorithm considers all the edges of weight at most  $w$ , it has constructed a spanning forest  $F$ . Let  $V_1, \dots, V_k$  be the vertex sets of the components of  $F$ , and let  $W_F = \{V_1, \dots, V_k\}$ . *Claim 1:* If  $F'$  is another forest obtained by Kruskal’s algorithm after checking all edges of weight at most  $w$ , then  $W_{F'} = W_F$ .

We prove Claim 1 inductively. **Basis Step:** For  $w$  less than the smallest weight,  $F$  must be the trivial forest  $(V(G), \emptyset)$ . **Inductive Step:** We may assume that  $W_F$  is the same for every forest  $F$  obtained by considering all edges of weight strictly less than  $w$ . Let  $S$  be the set of edges of weight  $w$ . Order them arbitrarily, and let Kruskal's algorithm continue by considering those edges in that order, and let  $F^*$  be a forest obtained. *Claim 2:* The vertex sets of the components of  $F^*$  are identical to the vertex sets of the components of  $F \cup S$ . *Proof:* Each component of  $F^*$  is contained in a component of  $F \cup S$  because  $F^* \subseteq F \cup S$ . Two components in  $F^*$  cannot be joined by an edge  $e \in S$ , or else  $e$  should have been added to  $F^*$  when it was considered in the algorithm. Thus Claim 2 is proved, and Claim 1 as well.

By Claim 1,  $|W_F|$  and  $|W_{F^*}|$  do not depend on the ordering of edges with the same weight. So  $|W_{F^*}| - |W_F|$  is a constant, and exactly that many edges of weight  $w$  are in any forest produced by Kruskal's algorithm. Thus, the list of edge-weights in a forest produced by Kruskal's algorithm is constant.

**2.3.9** Note that  $F \cup e$  is acyclic. Then there is a spanning tree  $T$  that contains  $F \cup e$ . (For example, you could start with  $G$  and repeatedly select an edge not in  $F \cup e$  that lies in a cycle, and remove that edge.) Suppose that  $T'$  is a spanning tree of  $G$  of minimum weight among those that containing  $F$ . Also, suppose that  $T'$  does not contain  $e$ .

$T' + e$  has a unique cycle  $C$ , which consists of  $e$  and a path  $P$  in  $T'$  of length at least two.  $P$  is not contained in  $F$ , since  $F \cup e$  is acyclic; so there exists an edge  $e' \in E(P) - E(F)$ .  $F \cup e'$  must be acyclic because it is contained in  $T'$ . Then  $w(e) \leq w(e')$  by the definition of  $e$ . Removing  $e'$  from  $T' + e$  breaks  $C$ , so  $T' + e - e'$  is acyclic. Then  $T' + e - e'$  is a spanning tree of  $G$ , and its weight is no more than  $w(T')$ , so it is a spanning tree of minimum weight among those that contain  $F$ .

**2.3.12** For all  $n \geq 4$  we can weight  $K_n$  so that it is a counterexample.

Let  $e_1, \dots, e_n$  be consecutive edges in a spanning cycle of  $K_n$ , and let  $v$  be the vertex common to  $e_1, e_2$ . Let  $w(e_i) = 0$  for all  $i \geq 3$ , let  $w(e_1) = w(e_2) = 3$ , and let all other edges of  $K_n$  have weight 1. The algorithm begins by taking all the weight 0 edges, and then is obligated to end by taking a weight 3 edge. However there is a spanning path that has two edges of weight 1 incident to  $v$ , and all other edges of weight 0. Thus the minimum weight spanning path has weight less than what is produced by the algorithm. (*Finding a minimum weight spanning path is actually an NP-hard problem.*)