Math 485, Graph Theory: Homework #3

Stephen G. Simpson

Due Monday, October 26, 2009

The assignment consists of Exercises 2.1.29, 2.1.35, 2.1.37, 2.2.18, 2.3.8, 2.3.10, 2.3.13, 2.3.14, 2.3.15 in the West textbook, plus the following, where L_n is the n-ladder.

1. Find a recursion formula for $\tau(L_n)$.
2. Use your recursion to calculate $\tau(L_n)$ for $n = 1, 2, \ldots, 10$.
3. Solve your recursion to get an explicit formula for $\tau(L_n)$.
4. Find $\lim\limits_{n\to\infty} \dfrac{\tau(L_{n+1})}{\tau(L_n)}$.

Each exercise counts for 10 points. Here are some (rather sketchy) solutions.

2.1.29. Let T be a tree. T is bipartite, so let X and Y be the partite sets. Assume that X contains at least half of the vertices of T. We are to show that X contains at least one leaf of T. (A *leaf* is a vertex of degree 1.)

Let $|X|$ denote the number of vertices in X, and similarly for Y. On the one hand, T is a tree, so the number of edges in T is equal to the number of vertices minus one, i.e., $|X| + |Y| - 1$. On the other hand, since X is one of the partite sets, the number of edges in T is equal to the sum of the degrees of the vertices in X. Thus we have

$$|X| + |Y| - 1 \;=\; \sum_{x\in X} \deg(x)\,.$$

If X contains no leaves, then for each $x \in X$ we have $\deg(x) \geq 2$, hence

$$|X| + |Y| - 1 \;\geq\; 2|X|\,,$$

hence $|X| \leq |Y| - 1$, i.e., X contains less than half of the vertices, Q.E.D.

Here is an alternative proof which does not use the degree sum formula. Let X and Y be as above and assume that X contains no leaves. Let r be one of the leaves, and designate r as the *root* of the tree. For all vertices v other than r, define $p(v)$ = the *predecessor* of v, i.e., the unique vertex which is adjacent to v and lies on the unique path from v to the root. Because X contains no leaves, it is easy to see that the function p maps $Y - r$ onto X. Hence $|Y| - 1 \geq |X|$, i.e., X contains less than half of the vertices, Q.E.D.

2.1.35. Let T be a tree. We are to prove:

> All vertices of T are of odd degree if and only if, for each edge $e \in T$, each component of $T - e$ has an odd number of vertices.

$\Rightarrow$: Assume that all vertices of T are of odd degree. Let T_1 be one of the components of $T - e$. Then T_1 has exactly one vertex of even degree. But, by the degree sum formula, T_1 has an even number of vertices of odd degree. Therefore, the total number of vertices of T_1 is odd.

$\Leftarrow$: Assume that for each edge e of T, each component of $T - e$ has an odd number of vertices. Let n be the total number of vertices in T. Since $T - e$ has exactly two components, n is even. Let v be any vertex of T. Let $T_1, \ldots, T_k$ be the components of $T - v$, and let $n_1, \ldots, n_k$ be the number of vertices in $T_1, \ldots, T_k$ respectively. Note that k is the degree of v. Our assumption implies that each n_i is odd. On the other hand, n is even, and $n = 1 + \sum_{i=1}^{k} n_i$. It follows that k is odd, Q.E.D.

2.1.37. Assume that T and T' are two spanning trees of a graph G. Let e be any edge of T which is not an edge of T'. Let u and v be the end vertices of e, let T_u and T_v be the components of $T - e$ containing u and v respectively, and let P be the unique uv-path in T'. Clearly P contains at least one edge e' with one end vertex in T_u and the other in T_v. It follows that $T - e + e' = T_u + T_v + e'$ is a spanning tree of G. Let u' and v' be the end vertices of e' lying in T_u and T_v respectively. Let $T'_{u'}$ and $T'_{v'}$ be the components of $T' - e'$ containing u' and v' respectively. Since u lies in $T'_{u'}$ while v lies in $T'_{v'}$, it follows that $T' - e' + e = T'_{u'} + T'_{v'} + e$ is a spanning tree of G.

2.2.18. The Matrix Tree Theorem tells us that $\tau(K_{r,s})$ is the determinant

$$\begin{vmatrix} r & \cdots & 0 & -1 & \cdots & -1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & r & -1 & \cdots & -1 \\ -1 & \cdots & -1 & s & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & \cdots & -1 & 0 & \cdots & s \end{vmatrix}$$

where there are s occurrences of r and $r - 1$ occurrences of s. Adding all of the other rows to the first row, we obtain

$$\begin{vmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & r & -1 & \cdots & -1 \\ -1 & \cdots & -1 & s & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & \cdots & -1 & 0 & \cdots & s \end{vmatrix}$$

where now there are only $s-1$ occurrences of r. Adding the first row to each of the last $r-1$ rows, we obtain

$$\begin{vmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & r & -1 & \cdots & -1 \\ 0 & \cdots & 0 & s & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & s \end{vmatrix}$$

and this is upper triangular, so the determinant is the product of the diagonal entries. Thus $\tau(K_{r,s}) = r^{s-1}s^{r-1}$.

2.3.8. We are to prove that, in a weighted graph, any two minimum-weight spanning trees have the same list of edge weights.

This follows from Exercise 2.3.13 below, where T and T' are spanning trees but only T is assumed to be a minimum-weight spanning tree. Now we are considering the special case where both T and T' are minimum-weight spanning trees. In this case, the solution of 2.3.13 shows that the edges e and e' have the same weight. Thus, replacing T' by $T'-e'+e$ does not change the list of edge weights.

2.3.10. The justification of Prim's Algorithm is similar to the justification of Kruskal's Algorithm.

2.3.13. In a weighted graph, let T be a minimum-weight spanning tree and let T' be a spanning tree other than T. Since T and T' have the same number of edges, let e' be an edge in T' which is not in T. By Exercise 2.1.37 above, let e be an edge of T which is not in T' and such that $T-e+e'$ and $T'-e'+e$ are spanning trees. Since T is a minimum-weight spanning tree, the weight of T is $\leq$ the weight of $T-e+e'$, hence the weight of e is $\leq$ the weight of e', hence the weight of $T'-e'+e$ is $\leq$ the weight of T'. Replacing T' by $T'-e'+e$ we move T' closer to T (in the sense that T and T' now have one more edge in common) and this move does not increase the weight of T'. A finite sequence of moves of this kind transforms T' into T without increasing the weight.

2.3.14. Let G be a weighted graph, let C be a cycle, and let e be an edge of maximum weight in C. The claim is that there exists a minimum-weight spanning tree of G which does not contain e.

Let T be a minimum-weight spanning tree of G. If T does not contain e, we are done. If T contains e, let u and v be the end-vertices of e, and let T_u and T_v be the components of T which contain u and v respectively. Since $C - e$ is a uv-path, let e' be an edge of $C - e$ which has one of its end-vertices in T_u and the other in T_v. Then $T - e + e'$ is a spanning tree of G. Since e is of maximum weight in C, the weight of e' is $\leq$ the weight of e. Hence, the weight of $T - e + e'$ is $\leq$ the weight of T. Since T is a minimum weight spanning tree of G, it follows that T' is also a minimum weight spanning tree of G. This proves the claim.

The above claim justifies the following "reverse Kruskal" algorithm. Let G be a connected weighted graph. Begin with $G_0 = G$. Given G_i, if G_i is a tree let $k = i$ and stop. Otherwise, let $G_{i+1} = G_i - e_i$ where e_i is a non-cut-edge of G_i of maximum weight. Since e_i belongs to a cycle of G_i, the above claim tells us that G_{i+1} contains a minimum-weight spanning tree of G_i. Hence by induction G_i contains a minimum-weight spanning tree of G. The algorithm stops after a finite number of steps, and then G_k is a minimum-weight spanning tree of G.

2.3.15. In a weighted graph, let T be a minimum-weight spanning tree, and let C be a cycle. The claim is that T omits at least one of the maximum-weight edges of C.

Suppose otherwise, i.e., T includes all of the maximum-weight edges of C. Let e be one of these maximum-weight edges, let u and v be the end-vertices of e, and let T_u and T_v be the components of $T - e$ which contain u and v respectively. Since $C - e$ is a uv-path, let e' be an edge in $C - e$ with one end-vertex in T_u and the other in T_v. Then $T - e + e'$ is a spanning tree. Since e' does not belong to T, it is not among the maximum-weight edges of C. Therefore, the weight of e' is $<$ the weight of e. It follows that the weight of $T - e + e'$ is $<$ the weight of T. Thus T is not a minimum-weight spanning tree, a contradiction.

For the last part of the assignment, let L_n be the n-ladder. Using the method of deleting and contracting edges, we obtain the recursion formula

$$\tau(L_n) = 4\tau(L_{n-1}) - \tau(L_{n-2})$$

for all $n \geq 3$. Since $\tau(L_1) = 1$ and $\tau(L_2) = 4$, it follows that $\tau(L_3) = 15$, $\tau(L_4) = 56$, $\tau(L_5) = 209$, $\tau(L_6) = 780$, $\tau(L_7) = 2911$, $\tau(L_8) = 10864$, $\tau(L_1) = 40545$, $\tau(L_{10}) = 151316$. Note also that our recursion formula holds for $n = 2$ if we define $\tau(L_0) = 0$. Using this, we can solve the recursion to get

$$\tau(L_n) = \frac{1}{2\sqrt{3}}\left(2+\sqrt{3}\right)^n - \frac{1}{2\sqrt{3}}\left(2-\sqrt{3}\right)^n.$$

Since $0 < 2 - \sqrt{3} < 1 < 2 + \sqrt{3}$, it is clear that

$$\lim_{n \to \infty} \frac{\tau(L_{n+1})}{\tau(L_n)} = 2 + \sqrt{3}$$

since $\left(2 - \sqrt{3}\right)^n$ is negligible for large n.