1. Weighted Enumeration of Spanning Trees of $Q_n$

Again, spanning trees of $Q_n$ were studied by Vic Reiner and myself in the paper [1], which is available from my MC’04 page. In these notes, I’ll restate our results somewhat more informally.

Recall that one can use the Matrix-Tree Theorem to prove that

\[
\tau(Q_n) = \prod_{S \subseteq [n], |S| \geq 2} 2|S| = \prod_{k=2}^{n}(2k)^{\binom{n}{k}}
\]  

(see [2]). Assign each spanning tree $T$ the weight monomial

\[
\text{wt}(T) = \prod_{e \in T} q_{\text{dir}(e)}
\]

where $\text{dir}(e) \in [n]$ is the direction of edge $e$—that is, the unique bit in which the endpoints of $e$ differ. With the Souped-Up Matrix-Tree Theorem, one can prove that

\[
\sum_{T \in T(Q_n)} \text{wt}(T) = q_1 \ldots q_n \prod_{S \subseteq [n], |S| \geq 2} \left(2 \sum_{i \in S} q_i\right) = \sum_{T \in T(Q_n)} \text{wt}(T) = 2^{2^n-n-1} q_1 \ldots q_n \prod_{S \subseteq [n], |S| \geq 2} \left(\sum_{i \in S} q_i\right),
\]

a stronger result than (1) (since one can recover (1) from (3) by setting all the $q_i$’s to 1).

Actually, we can do even better. Remember that the Prüfer code allows us to enumerate spanning trees of the complete graph $K_n$ and keep track of the valence of each vertex. We can’t quite do that for $Q_n$. I mean, we can certainly try and study the polynomial

\[
\sum_{T \in T(Q_n)} \prod_{v \in V(Q_n)} x_{v}^{\text{val}(T)}
\]

but this is not at all a nice expression. Yes, you can certainly factor out one copy of each variable, but what is left is an unsightly mess even for $n = 3$.

However, we can keep track of the following data. Let $e \in E(Q_n)$ be an edge in direction $i$, with endpoints

\[
(v_1, v_2, \ldots, v_i, \ldots, v_n), \quad (v_1, v_2, \ldots, \overline{v}_i, \ldots, v_n).
\]

(The bar, of course, means binary complement.) We want to keep track of which of the constant bits (that is, the $v_j$’s for $j \neq i$) are 1’s and which are 0’s. So associate to $e$ the following Laurent monomial in the variable set $\Psi = \{q_1, \ldots, q_n, x_1, \ldots, x_n\}$:

\[
\text{wt}(e) = q_i \prod_{j \neq i} x_j^{2v_j - 1}.
\]

Notice that

\[
2v_j - 1 = \begin{cases} 
1 & \text{if } v_j = 1, \\
-1 & \text{if } v_j = 0.
\end{cases}
\]

For example, here are the weights of all the edges in $Q_3$:
We now redefine the weight of a spanning tree $T \in \mathcal{T}(Q_n)$ as

$$\text{wt}(T) = \prod_{e \in T} \text{wt}(e),$$

which again is a Laurent monomial in the variables $\Psi$. Any theorem about these new weights can in principle be specialized to a statement about the old weights $\text{wt}(T)$ (see (2)) by setting all the $x_i$'s to 1.

Now, using the Souped-Up Matrix-Tree Theorem and the edge weights given by (4), we can prove the following result (see [1, Theorem 3]; the notation is a little bit different but it's really the exact same formula):

$$\sum_{T \in \mathcal{T}(Q_n)} \text{wt}(T) = q_1 \cdots q_n \prod_{S \subseteq [n], |S| \geq 2} \sum_{i \in S} q_i \left(x_i^{-1} + x_i\right).$$

Note that setting all $x_i$'s to 1 recovers (3), so this result really is more general. The proof of (6) involves linear algebra over arbitrary integral domains (such as the ring of Laurent polynomials in the variables $\Psi$); it’s not especially hard, but it’s sufficiently technical that I don’t want to talk about it here (see [1] or talk to Jeremy if you are really interested).

**References**
