Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics 155 (2007) 2594–2600

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

On encodings of spanning trees

Glenn H. Hurlbert[1]

Department of Mathematics and Statistics, Arizona State University, Tempe, AZ 85287-1804, USA

Received 29 September 2004; received in revised form 31 May 2007; accepted 2 July 2007
Available online 27 August 2007

Abstract

Deo and Micikevicius recently gave a new bijection for spanning trees of complete bipartite graphs. In this paper we devise a generalization of Deo and Micikevicius's method, which is also a modification of Olah's method for encoding the spanning trees of any complete multipartite graph $K(n_1, \ldots, n_r)$. We also give a bijection between the spanning trees of a planar graph and those of any of its planar duals. Finally we discuss the possibility of bijections for spanning trees of DeBriujn graphs, cubes, and regular graphs such as the Petersen graph that have integer eigenvalues.
© 2007 Elsevier B.V. All rights reserved.

PACS: 05C05; 05C85; 68R10

Keywords: Spanning tree; Complete multipartite graph; Bijection

1. Introduction

Given a graph $G = (V, E)$ with vertices $V = V(G)$ and edges $E = E(G)$, a *spanning tree* $T = (V, E')$ of G is a connected acyclic subgraph of G. One very natural and old problem is to determine the number $\tau(G)$ of labelled spanning trees for a fixed graph G, or better yet, a formula for each in a family $\mathscr{G} = \{G_n\}_{n=0}^{\infty}$.

The first and most famous result of this kind is due to Cayley [5], who proved that $\tau(K_n) = n^{n-2}$, where K_n is the complete graph on n vertices. A wonderful collection of many different proofs of this result is found in [16]. The first bijective method of this result was given by Prüfer [18]. It is possibly the simplest proof of Cayley's Theorem. A generalization of Prüfer's encoding was given by Olah [17] for complete multipartite graphs $K(n_1, \ldots, n_r)$. This graph has $n = \sum_{i=1}^{r} n_i$ vertices, partitioned into sets of sizes $n_1, \ldots, n_r$, and edges between every pair of vertices from different parts. The encoding proves.

Theorem 1. $\tau(K(n_1, \ldots, n_r)) = n^{r-2} \prod_{i=1}^{r} (n - n_i)^{n_i - 1}$.

Another wonderful bijection for these graphs is found in [10,11], where the authors give a full description of the q-analog properties of their bijections.

Recently, a new method has been found by Deo and Micikevicius [9] that allows one both to encode and decode a spanning tree of K_n in such a way that the diameter, center, and radius of the tree can be calculated directly from the

[1] Partially supported by National Security Agency Grant #MDA9040210095.
E-mail address: hurlbert@asu.edu.

0166-218X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2007.07.014

code without having to resort to other algorithms that operate on the tree itself. In this paper we devise a generalization of Deo and Micikevicius's method which is also a modification of Olah's method for encoding the spanning trees of any complete multipartite graph $K(n_1, \ldots, n_r)$.[2] Our method shares the benefits of finding the diameter, center, and radius directly. We describe this method in Section 2.

We discuss the possibility of bijections for spanning trees of integral graphs, such as the Petersen graph and d-dimensional cubes, and of DeBruijn Graphs in Section 3. We also give a bijection between the spanning trees of a planar graph and those of any of its planar duals. The section includes some open problems.

2. Complete multipartite graphs

2.1. Encoding a tree

Let T be a fixed spanning tree of $K = K(n_1, \ldots, n_r)$, and let $n = \sum_{j=1}^{r} n_j$ be the number of its vertices. Let the partition of the vertices V given by K be $V_1, \ldots, V_r$, where each $|V_j| = n_j$, and define $s_j = \sum_{i=1}^{j} n_i$. Then we may assume that $V_j = \{s_{j-1} + 1, \ldots, s_j\}$. Define $V(k) = j$, where $k \in V_j$. The code for T will be a set of sequences $C = \{C_0, C_1, \ldots, C_r\}$, where the length of C_0 is $r - 2$ and the length of each C_j $(1 \leqslant j \leqslant r)$ is $n_j - 1$. Thus the total length of the code is $(r-2) + \sum_{j=1}^{r}(n_j - 1) = n - 2$, as expected. Fig. 1 shows the algorithm **E** used to encode T by C. The sequence C_0 can contain any of the labels, while every other sequence C_j will contain the labels of the neighbors, at appropriate stages of the algorithm, of the vertices in V_j (in some specified order). The set of all possible such codes has cardinality matching the right-hand side of Theorem 1. When k is a leaf of T denote its unique neighbor in T by k^+.

Fig. 2 shows an example of **E** encoding a tree T.

Proposition 2. *Algorithm* **E** *is injective.*

Proof. For a given tree T and for $0 \leqslant t \leqslant \text{radius}(T)$ denote by $T^{(t)}$ the subgraph of T induced by the vertices $Q_t \bigcup_{j=0}^{t-1} (Q_j \cup \text{nbhd}(Q_j))$, where nbhd$(W)$ is the set of vertices adjacent to some vertex of W. Suppose that Algorithm **E** encodes the trees T_1 and T_2 by the same code C. According to **E** any vertex that does not appear in C is a leaf, and so T_1 and T_2 have the same set of leaves; i.e. $Q_0(T_1) = Q_0(T_2)$ and $T_1^{(0)} = T_2^{(0)}$. Using induction we will assume that $Q_{t-1}(T_1) = Q_{t-1}(T_2)$ and $T_1^{(t-1)} = T_2^{(t-1)}$ for some $t > 0$. Because C encodes both trees, the neighbors of Q_{t-1} are determined by C and so $T_1^{(t)} = T_2^{(t)}$. This also determines the leaves in the remaining tree at this stage, and so $Q_t(T_1) = Q_t(T_2)$. Hence $T_1 = T_2$. □

2.2. Decoding a sequence

Suppose C is a code satisfying the conditions described in Section 2.1. We describe Algorithm **D** (see Fig. 3), which constructs from C the tree T that Algorithm **E** encodes as C. Values of the n-tuple U are initialized in the loop at line 2 to one less than the degrees of the vertices so that the leaves become evident, forming the queue Q_0 (we set $n_0 = r - 1$) in the loop at line 3. The process of constructing T is facilitated by the iterative construction in the loop at line 5 of each queue $Q_1, Q_2, \ldots$, essentially reproducing them in the same manner done by **E**. The variables k and k' are redefined each time they are removed; that is, in line 5(a)i for example, k is defined to be the head of Q_j and then it is removed.

Proposition 3. *Algorithm* **D** *is injective.*

Proof. By the above discussion, Algorithm **D** is the realization of the argument given in the proof of Proposition 2. Hence Algorithm **D** is the inverse of Algorithm **E**. □

[2] It is worth correcting a small typographical error that appears in Deo and Micikevicius's paper: In each of lines 4 and 5 of Fig. 3 in [9] the variable *used*$[i]$ should be replaced by *used*$[C(i)]$.