Univerzita Mateja Bela v Banskej Bystrici
Fakulta prírodných vied

Katedra matematiky

Šachovnicové reprezentácie a počítačové programy pre graciózne ohodnotenia stromov

(ŠVOČ)

Autor práce: Michal Ivaška
Vedúci práce: Doc. RNDr. Miroslav Haviar, CSc.

2009

University of Matej Bel in Banská Bystrica
Faculty of Natural Sciences

Department of Mathematics

Chessboard Representations and Computer Programs for Graceful Labelings of Trees

(Student Competition)

Author: Michal Ivaška
Supervisor: Doc. RNDr. Miroslav Haviar, CSc.

2009

I would like to thank to Dr Miroslav Haviar for valuable ideas and his assistance with my work.

Contents

1 Introduction

The question whether every tree can be gracefully labeled (The Ringel-Kotzig Conjecture) is one of the most famous problems in discrete mathematics. Over the last fifty years more than a thousand papers have been devoted to this problem. Our work aims at contributing to the problem by giving a 'chessboard' reformulation of the problem and its applications, and by producing computer programs for finding graceful labelings of small trees.

The chessboard representation gives a nice visualisation of a gracefully labeled tree. As its applications, we give the Sheppard's result [14] on the number of gracefully labeled graphs with q edges, and the fact that finding a graceful labeling of a tree G means transfering its assigned G-chessboard to a graceful G-chessboard via so-called elementary chessboard operations (Section 3).

In Sections 4-6 we describe three computer programs that we developed for finding graceful labelings of trees. The program COUNT calculates the number of all nonisomorphic trees with n vertices for small n, and gives one graceful labeling for each of them via its chessboard representation. The program FIND calculates and lists all graceful labelings of a given tree. And the program TRY is a useful tool for finding a graceful labeling of a given tree manually. The programs are programmed in Borland Delphi 6 (which is based on Pascal) and are attached on DVD.

2 Preliminaries

The following basic concepts are taken from [3].

2.1 Graphs

Definition 2.1 *A graph G is a triple consisting of a vertex set $V(G)$, an edge set $E(G)$, and a function h_G that assigns to each edge $e \in E(G)$ an unordered pair of vertices. When $h_G(e) = \{u, v\}$, we say that u and v are the endpoints of e and that e is incident to u and v. A graph is simple if the function $h_G(e)$ is injective. In this case, we write $e = uv$ instead of $h_G(e) = \{u, v\}$.*

The terms 'vertex' and 'edge' come from geometry. We can visualize graphs by drawing them in the plane. To each vertex we assign a point; to each edge we assign a curve that joins the points assigned to its vertices.

Definition 2.2 *The degree $d(x)$ of a vertex $x \in V(G)$ is the number of edges in G incident to x.*

Definition 2.3 *A trail (of length k) in a graph G is a list $v_0, e_1, v_1, e_2, \dots, e_k, v_k$ that alternates between vertices and edges, such that $h_G(e_i) = v_{i-1}v_i$ for all i and $e_1, \dots, e_k$ are distinct elements of $E(G)$.*

2.2 Isomorphism of Graphs

Definition 2.4 *An isomorphism from a simple graph G to a simple graph H is a bijection $f : V(G) \to V(H)$ such that $uv \in E(G)$ iff $f(u)f(v) \in E(H)$.*

We say "G is isomorphic to H" if there is an isomorphism from G to H. The set of pairs G, H such that G is isomorphic to H is the *isomorphism relation*.

Definition 2.5 *Vertices u and v in a graph G are adjacent if they are the endpoints of an edge. The adjacency relation of G (defined on $V(G)$) is the set of ordered pairs (u, v) such that u and v are adjacent.*

The *adjacency relation* is symmetric. In the language of adjacency, simple graphs G and H are isomorphic iff there is a bijection $f : V(G) \to V(H)$ that preserves the adjacency relation. The *isomorphism relation* is an equivalence relation on the set of simple graphs.

2.3 Connection and Trees

Now we confine our attention to simple graphs, viewing the edge set of a graph as a set of unordered pairs of vertices, since a simple graph has (at most) one edge with specified endpoints v_{i-1} and v_i. We consider special types of trails.

Definition 2.6 *A path is a trail with no repeated vertex. A u, v-path is a path with endpoints u and v. A cycle is a closed trail in which "first = last" is the only vertex repetition.*

Definition 2.7 *A graph G is connected if for every pair $u, v \in V(G)$, there is a u, v-path in G.*

Definition 2.8 *A tree is a connected graph with no cycles. A leaf is a vertex of degree 1.*

Lemma 2.9 ([3]; Lemma 11.39) *Every tree with at least two vertices has a leaf, and deleting a leaf from a tree yields a tree with one less vertex.*

Theorem 2.10 ([3]; Theorem 11.40) *Every tree with n vertices has $n-1$ edges.*

2.4 Graceful Labeling

The following preliminary facts about graceful labeling of a graph are taken from [4].

A graph labeling is an assignment of integers to the vertices or edges, or both, subject to certain conditions. Graph labelings were first introduced in the late 1960s. Since then dozens of graph labeling techniques have been studied in over 1000 papers.

Rosa [13] called a function f a *β-valuation* of a graph G with q edges if f is an injection from the vertices of G to the set $\{0, 1, \ldots, q\}$ such that, when each edge xy is assigned the label $|f(x) - f(y)|$, the resulting edge labels are distinct. Golomb [5] subsequently called such labelings *graceful*. Sheppard [14] has shown that there are exactly $q!$ gracefully labeled graphs with q edges.

The Ringel-Kotzig Conjecture saying that *all trees are graceful* has been the focus of many papers. Among the trees known to be graceful are for example: caterpillars [13] (a *caterpillar* is a tree with the property that the removal of its endpoints leaves a path); trees with at most 4 end-vertices [8], [15] and [9]; trees with diameter at most four [15] and five [7]; trees with at most 27 vertices [1]. In 1979 Bermond [2] conjectured that lobsters are graceful (a *lobster* is a tree with the property that the removal of the endpoints leaves a caterpillar). Mishra and Panigrahi [11] and [12] found classes of graceful lobsters of diameter at least five.

Despite the efforts of many, the graceful tree conjecture remains open. It remains open even for trees with maximum degree 3. In a paper published in 2004 Krishnaa [10] claims to have proved that all trees have graceful labeling. However, her proof was flawed.

3 Chessboard Representation

The idea of representing gracefully labeled graphs via chessboards is due to M. Haviar [6].

3.1 Chessboard Representation of a Labeled Graph

In what follows, G will be a labeled simple graph without isolated vertices. Without lost of generality we can assume that its vertices are labeled with distinct numbers from the set $\{1, 2, \ldots, n\}$. As the graph is simple, its edge set is a set of unordered pairs of vertices. So let $E(G) = \{v_{a_1}v_{b_1}, v_{a_2}v_{b_2}, \ldots, v_{a_m}v_{b_m}\}$, where $v_{a_i}, v_{b_i} \in \{1, 2, \ldots, n\}$ for every $1 \leq i \leq m$. We shall only consider the labeling of edges given by $l(v_{a_i}v_{b_i}) = |v_{a_i} - v_{b_i}|$ for every edge $v_{a_i}v_{b_i}$ of G.

Now consider a table with n rows and n columns. Let the square with coordinate $[i, j]$ of the table means the square in the i-th column counting from the left and the j-th row counting from the top. Let *the r-th diagonal* be the set of all squares with coordinates $[i, j]$ where $|i - j| = r$ and $i \leq j$. The 0-th diagonal will sometimes be called *the main diagonal* and the other r-th diagonals with $r \neq 0$ will be called *associate diagonals*.

We assign a table B^G to each finite labeled graph G as follows: for every edge $v_{a_i}v_{b_i} \in E(G)$ we draw a dot in the pair of squares with the coordinates $[v_{a_i}, v_{b_i}]$ and $[v_{b_i}, v_{a_i}]$. Denote this dots $d_{v_{a_i},v_{b_i}}$ and $d_{v_{b_i},v_{a_i}}$, respectively. We obtain a table with $2m$ dots, where $m = |E(G)|$. We shall call this table a *G-chessboard* and denote it by B_n^G (see Figure 1) or simply B_n if the considered graph G is clear from the context.

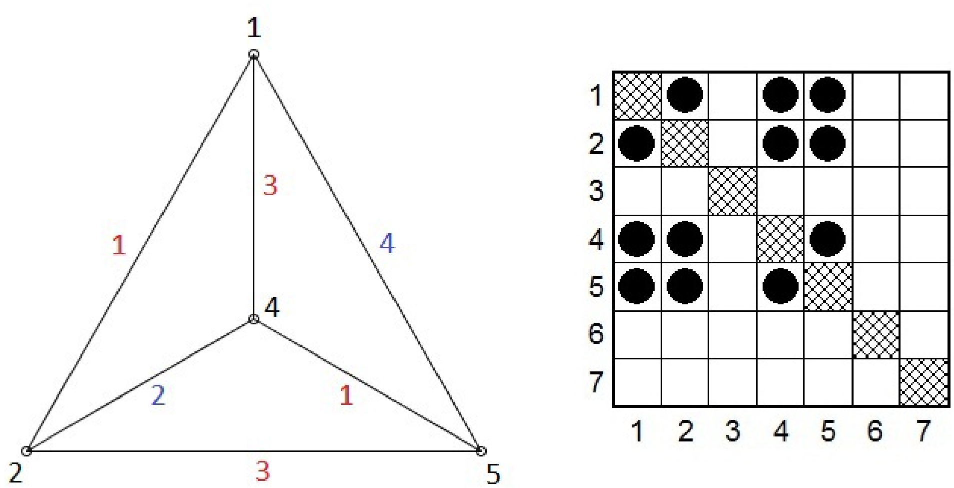

Figure 1: Graph G and its corresponding G-chessboard B_7^G

Clearly, the G-chessboards we consider are symmetric by the main diagonal.

3.2 Chessboard Representation of a Gracefully Labeled Graph and Sheppard's Result

Let G be a simple graph with vertices labeled with selected distinct numbers from the set $\{1, 2, \dots, n\}$, with $q = n - 1$ edges, and whenever two vertices x, y are endpoints of an edge e, then the label of the edge e is $l(e) = |x - y|$. Notice that if a graph G is graceful, then the labels of its edges are distinct numbers $1, \dots, q$, and thus there is exactly one dot on every associate diagonal of the assigned G-chessboard B_n^G. Let us call such G-chessboard *graceful*. For example, the G-chessboard assigned to the gracefully labeled complete graph $G = K_4$ in Figure 2 is graceful.

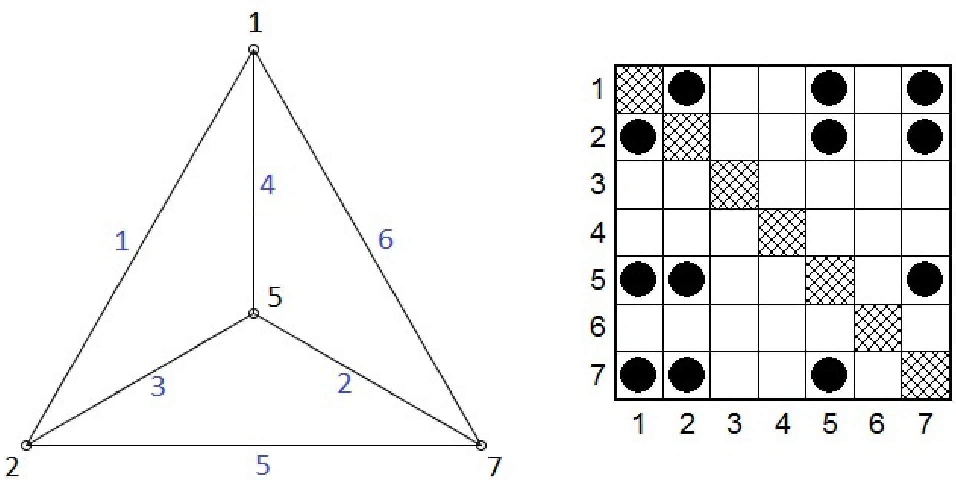

Figure 2: Graph K_4 and its corresponding graceful G-chessboard $B_7^{K_4}$

Let n be a natural number, $n > 0$. Let $\mathcal{L}_n$ be the set of all gracefully labeled simple graphs G with $q = n - 1$ edges and vertices labeled with selected numbers from the set $\{1, 2, \dots, n\}$. Let $\mathcal{B}_n$ be the set of all assigned graceful G-chessboards. We define a map $f : \mathcal{L}_n \to \mathcal{B}_n$ which maps each gracefully labeled graph to its assigned graceful chessboard. Clearly, for every $G \in \mathcal{L}_n$ there is a unique assigned chessboard $B_n^G \in \mathcal{B}_n$, as defined in Section 3.1. Also, for every graceful chessboard $B_n \in \mathcal{B}_n$, one can assign a unique graceful graph $G \in \mathcal{L}_n$ such that $B_n = B_n^G$. Consequently, the map $f : \mathcal{L}_n \to \mathcal{B}_n$ is a bijection.

Hence, to calculate $|\mathcal{L}_n|$, it suffices to calculate $|\mathcal{B}_n|$. For every $B_n \in \mathcal{B}_n$, there is exactly one dot on every associate diagonal. There are exactly i possibilities one can place the dot on the associate diagonal consisting of i squares, $1 \leq i \leq n-1$. Thus, there are exactly $1 \cdot 2 \cdots (n-1) = (n-1)!$ different graceful G-chessboards B_n with this property. So $|\mathcal{B}_n| = |\mathcal{L}_n| = (n-1)! = q!$. We have shown the following result due to Sheppard [14]:

Theorem 3.1 *There are exactly $q!$ gracefully labeled graphs with q edges.*

3.3 Chessboard Representation of a Gracefully Labeled Tree

We showed that all graceful G-chessboards correspond to gracefully labeled graphs. Now we will add other conditions to a G-chessboard, in order for the corresponding graph to be a tree.

By the definition, a tree is a connected graph with no cycles. It is possible to decide whether a graph is a tree or not when we see the diagram of the graph. To decide whether a G-chessboard represents a tree, we will need some extra conditions. In our further considerations, it will be more convenient to use another characterization of a tree, namely that a graph with n vertices is a tree iff it is connected and has $n-1$ edges.

We say that *two dots* $d_{i,j}$ and $d_{u,v}$ in a G-chessboard are *'connectable'* if there is a sequence $d_{i,j} = d_{i_0,j_0}, d_{i_1,j_1}, \ldots, d_{i_m,j_m} = d_{u,v}$ of dots such that for any two consecutive dots $d_{i_k,j_k}, d_{i_{k+1},j_{k+1}}$ either $i_k = i_{k+1}$ or $j_k = j_{k+1}$, i.e., one can 'jump' from one dot to the other within the same column or row. Further, we say that *two vertices* i and j are *'connectable'* if there are dots in the i-th and the j-th columns of the G-chessboard that are connectable (instead of the columns we can consider rows here as the G-chessboards are symmetric).

Let G be a labeled graph with n vertices $1, 2, \ldots, n$ and let B_n^G be its corresponding G-chessboard. Then G is a tree iff the following conditions hold:

(1) B_n^G has in total $(n-1)$ dots in the associate diagonals and

(2) the vertices i and $(i+1)$ are connectable for every $1 \leq i \leq (n-1)$.

The first condition corresponds to the fact that the graph G has $(n-1)$ edges. The second condition corresponds to the fact that there is a path between any two vertices in the graph, so the graph is connected; it explicitly says, that there must be a dot in every column (row) of the G-chessboard.

If we asssume there exists a graceful labeling for every n-element tree, then obviously we are able to find it from an arbitrary starting labeling of the vertices with the numbers $1, 2, \ldots, n$ by a finite sequence of permutations (ij). However, it is not clear how to do it efficiently.

By an *elementary chessboard operation* (ECO) we shall mean a mutual replacement of the i-th and j-th columns ($i \neq j$) of the G-chessboard followed by the mutual replacement of the i-th and j-th rows of the chessboard. We will then denote it as ECO(i, j). In the assigned graph it corresponds exactly to the permutation (ij) of the vertices. We shall say that two G-chessboards assigned to n-element trees are *equivalent* if there is a finite sequence of ECOs transferring one chessboard to the other. It is easy to see that the following statement is true:

Theorem 3.2 *Two labeled trees are isomorphic iff their assigned G-chessboards are equivalent.*

Hence, as the second application of our chessboard representation of gracefully labeled graphs we present the fact that to find a graceful labeling of a given tree G means to find a sequence of ECOs for transfering a G-chessboard to a graceful G-chessboard.

4 Description of Program COUNT

COUNT (Figure 3) is a computer program for counting nonisomorphic trees of a given number of vertices n. After the calculations its result is a number of all nonisomorphic trees with n vertices. In addition, the program creates a text file 'tables(n).txt' with chessboards of all nonisomorphic n-element trees using chessboard representation (Figure 4). Following every chessboard is a list of all edges of the tree. Although it can be easily read from the chessboard, it is there as an input to be easily used for other programs, for example for the program TRY.

For each of all nonisomorphic trees found, the program always saves in the text file the chessboard representation of the first graceful labeling found for the given

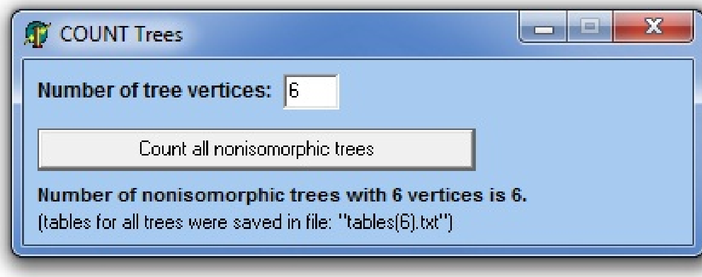

Figure 3: Program COUNT

```
TABLES OF ALL NONISOMORPHIC TREES WITH 6 VERTICES

TREE NO. 1:    EDGES  TREE NO. 2:    EDGES  TREE NO. 3:    EDGES
1|=|*|*|*|*|*| 1 2    1|=| |*|*|*|*| 1 3    1|=| | |*|*|*| 1 4
2|*|=| | | | | 1 3    2| |=|*| | | | 1 4    2| |=|*|*| | | 1 5
3|*| |=| | | | 1 4    3|*|*|=| | | | 1 5    3| |*|=| | | | 1 6
4|*| | |=| | | 1 5    4|*| | |=| | | 1 6    4|*|*| |=| | | 2 3
5|*| | | |=| | 1 6    5|*| | | |=| | 2 3    5|*| | | |=| | 2 4
6|*| | | | |=|        6|*| | | | |=|        6|*| | | | |=|
  1 2 3 4 5 6           1 2 3 4 5 6           1 2 3 4 5 6

TREE NO. 4:    EDGES  TREE NO. 5:    EDGES  TREE NO. 6:    EDGES
1|=| | |*|*|*| 1 4    1|=| |*| |*|*| 1 3    1|=| | | |*|*| 1 5
2| |=| |*| | | 1 5    2| |=| | |*| | 1 5    2| |=| |*|*| | 1 6
3| | |=|*| | | 1 6    3|*| |=|*| | | 1 6    3| | |=|*| | | 2 4
4|*|*|*|=| | | 2 4    4| | |*|=| | | 2 5    4| |*|*|=| | | 2 5
5|*| | | |=| | 3 4    5|*|*| | |=| | 3 4    5|*|*| | |=| | 3 4
6|*| | | | |=|        6|*| | | | |=|        6|*| | | | |=|
  1 2 3 4 5 6           1 2 3 4 5 6           1 2 3 4 5 6

Number of nonisomorphic trees with 6 vertices is 6.
```

Figure 4: Chessboards and edge lists from the output file 'tables(6).txt'

tree (see again Figure 4). So there is no need to check the existence of graceful labelings of trees with a given number of vertices by some other program.

The algorithm of program COUNT is very simple. It generates all possible trees with given number of vertices. After each new graph is generated, the algorithm checks all saved graphs so far, and if none of the saved graphs in the database is isomorphic to this new graph, then the new graph is added to database. The set of all trees with given number of vertices is of course finite, so the algorithm will get all nonisomorphic trees after finite number of steps. Even though it works with graphs with more than a dozen vertices, in such cases the time to compute can be (especially on slower computers) very long.

Generating of all trees with given number of vertices uses chessboard characterization. At first it generates gracefully labeled graphs. After that, it checks whether a gracefully labeled graph is a tree or not (using the two conditions from the Section 3.3). If it is a tree, then it has to be compared to the other trees in the database. By 2.4, two trees G and H are isomorphic if there is a bijection $f : V(G) \to V(H)$ such that $uv \in E(G)$ iff $f(u)f(v) \in E(H)$. This is a sufficient condition, but for computer program it is more convenient to employ also one condition for necessity, which says that if two trees are isomorphic, then the isomorphism maps each vertex to a vertex of the same degree. So if, for example, the vertices of a tree have degrees 1,1,1,3 and the vertices of another tree have degrees 1,1,2,2, the trees cannot be isomorphic. Using this condition in our program shortens the time needed for checking whether two trees are isomorphic.

5 Description of Program FIND

The program FIND (Figure 5) is a program for finding all graceful labelings of a given graph. As for graphs with more than a dozen vertices the calculation would take a long time, the program can be set to find only one labeling, which takes a reasonable time even for 'bigger' graphs (the user ticks the dialog box saying "Find only one labeling" - see Figure 5).

First we need to enter the graph of which graceful labelings we want to compute. We label the vertices of our graph with variables $x_1, x_2, \ldots, x_n$, where n is the number of vertices. We can label the vertices arbitrarily, but two different vertices must be labeled with different variables. The input to the program is the set of edges simply written like this: if there is an edge x_2x_7 in the graph, the edge in the input will be '2 7' (indexes separated with space). Each edge represented by a pair of vertices is given on separate line (see the left top part of Figure 5).

The found graceful labelings of trees are represented by sequences of labels of variables $x_1, x_2, \ldots, x_n$ in the form $(x_1)(x_2)\ldots(x_n)$, where (x_i) is the label of variable x_i, hence the sequences $(x_1)(x_2)\ldots(x_n)$ are certain permuations of the ordered n-tuple $[1, 2, \ldots, n]$. These special permutations tell us how to label our graph in

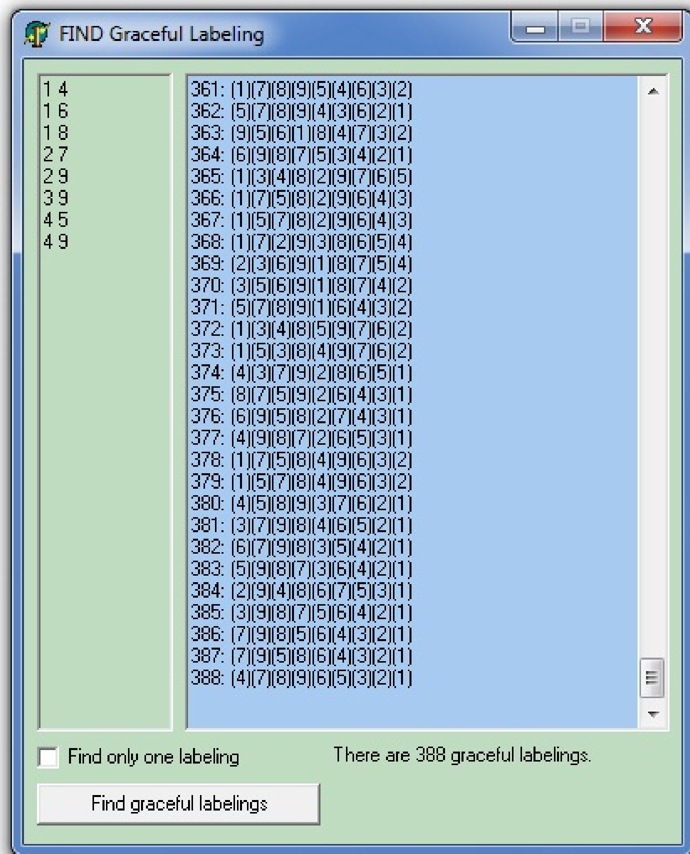

Figure 5: Program FIND

order to obtain the graceful labeling. For example, if the permutation in the output is (4)(7)(8)(9)(6)(5)(3)(2)(1) (see the last permutation in Figure 5), we label the vertex x_1 with 4, the vertex x_2 with 7, etc.

Even though the results written as permutations are listed directly in the program window, the program FIND also creates a text file 'tables.txt' with all the results. In addition, to each permutation representing a graceful labeling, it creates a table with its chessboard representation. Note that two different labelings can have the same chessboard (see Figure 6).

The program uses its own implemented algorithm for generating permutations. It generates all permutations of the set $\{1, 2, \ldots, n\}$, where n is the number of ver-

```
GRAPH EDGES:
1 2
1 3
1 4
```

```
LABELING 1: ((1)(2)(3)(4))
1|=|*|*|*|
2|*|=| | |
3|*| |=| |
4|*| | |=|

LABELING 2: ((1)(3)(2)(4))
1|=|*|*|*|
2|*|=| | |
3|*| |=| |
4|*| | |=|
```

Figure 6: Different labelings of a tree with identical chessboards

tices of a given graph. Then every permutation is checked whether it gives a graceful labeling or not. If the answer is positive, then the permutation is added to the program output and along with the chessboard, it is saved into the file 'tables.txt'.

6 Description of Program TRY

The program TRY (Figure 7) is more complex than the previous ones. It is a useful tool for finding a graceful labeling of some given graph manually. The input is the same as in program FIND, the user enters the edges of a graph in the same way.

After the graph input in entered, the program draws the graph (see Figure 7) and user is able to fill in numbers in the small squares representing the graph vertices. After pressing ENTER button, the program checks whether there are no conflicts (a conflict arises if two vertices have the same number or there is a filled in number greater than the number of vertices of the graph). Then the program labels with $l(uv) = |u - v|$ all edges uv which have the numbers u and v already filled in. The program displays the value $l(uv)$ of the edge uv in the graph diagram: in green if the value $l(uv)$ is unique in the diagram of the graph and in red otherwise. Of course we can fill in the vertex numbers step by step and see if the edge values are distinct.

Also, after pressing ENTER button, the program displays the edges in the chessboard. Since we know that there must be a dot (representing edge) on every associate diagonal for a labeling to be graceful, the chessboard presented in the right bottom part of the program window (see Figure 7) can help us with deciding how to fill in the vertex numbers. If we want to make a step back to the position after the previ-

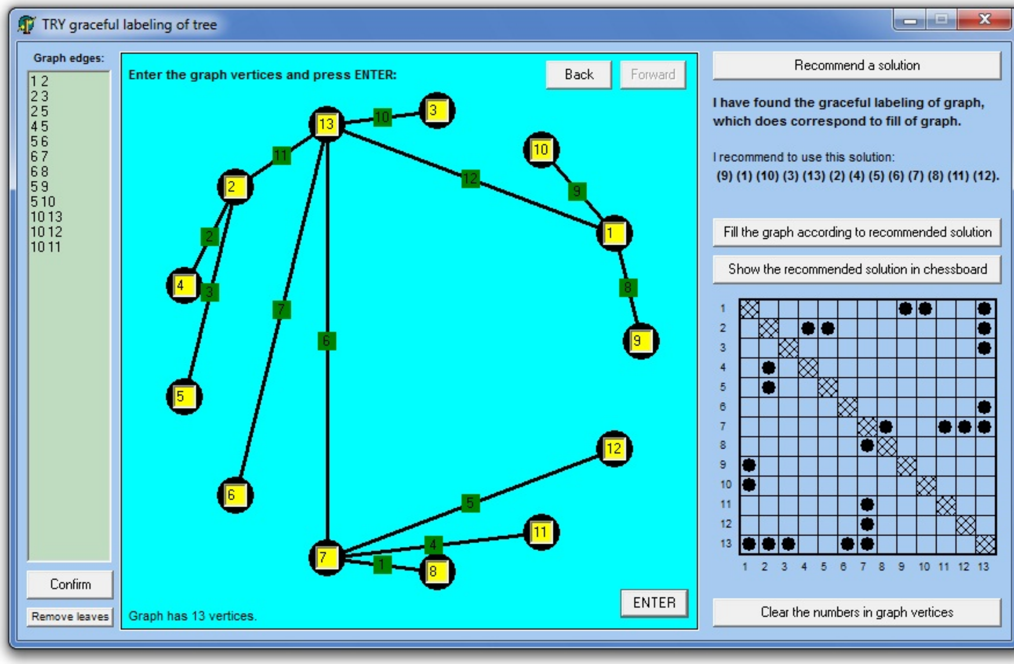

Figure 7: Program TRY

ous pressing of ENTER button, there is such a possibility, the program remembers its steps.

There is also a possibility to get a hint from the program. After pressing the button "Find and recommend a solution", program finds a solution using the same algorithm as in program FIND. In addition, if the number of graph vertices does not exceed ten, the program chooses the most convenient solution, particularly the one which is the 'closest' to the actual filling of the graph (so that we have to change as less vertex numbers as possible). For bigger graphs it only finds one solution, because for all solutions it would need long time. When the program recommends a solution, there is also a possibility to show the recommended solution in the chessboard (even before we actually fill in the vertex numbers in the graph diagram).

Another feature of the program is removing the leaves. The button to do this can be used repeatedly until there is only one vertex or one edge left. The algorithm of doing this is following. The program calculates the degrees of every vertex (as a number of edges which connect this vertex to other vertices). Whenever the degree of a vertex is 1 (i.e., it is a leaf), after pressing the button "Remove leaves" the program will not display this vertex anymore and neither it will display the edge which is connecting this leaf with the rest of the graph. In addition, it also displays the chessboard representation of such modified graph, so that we can see which dots are disappearing after disposal of leaves.

7 Conclusion

Our work is a small contribution to the famous problem of graceful labelings of trees. We introduced the chessboard representation of a labeled graph and presented some of its applications. More precisely, we gave the Sheppard's result [14] on the number of gracefully labeled graphs with q edges and we showed that finding a graceful labeling of a tree G means transfering its assigned G-chessboard to a graceful G-chessboard via elementary chessboard operations.

We described three computer programs that we developed as useful tools for finding graceful labelings of trees. The program COUNT calculates the number of all nonisomorphic trees with n vertices for small n, and gives one graceful labeling for each of them via its chessboard representation. The program FIND calculates and lists all graceful labelings of a given tree. The program TRY is a tool for finding a graceful labeling of a given tree manually. The programs were programmed in Borland Delphi 6 and are attached on DVD.

References

[1] R.E. L. Aldred and B. D. McKay, Graceful and harmonious labelings of trees, J. Gallian's personal communication.

[2] J.C. Bermond, Graceful graphs, radio antennae and French windmills, *Graph Theory and Combinatorics*, Pitman, London (1979) 18-37.

[3] John P. D'Angelo and Douglas B. West, *Mathematical Thinking* (Problem-Solving and Proofs), Prentice-Hall, (1998), 171-185.

[4] Joseph A. Gallian, *A Dynamic Survey of Graph labeling*, University of Minnesota Duluth, (twelveth edition published January 31, 2009), 1-9.

[5] S. W. Golomb, How to number a graph, in *Graph Theory and Computing*, R. C. Read, ed., Academic Press, New York (1972) 23-37.

[6] M. Haviar, Chessboard Representations of Labeled Graphs, personal communication.

[7] P. Hrnčiar and A. Haviar, All trees of diameter five are graceful, *Discrete Math.*, **233** (2001) 133-150.

[8] C. Huang, A. Kotzig, and A. Rosa, Further results on tree labelings, *Util. Math.*, **21c** (1982) 31-48.

[9] D. J. Jin, F. H. Meng, and J. G. Wang, The gracefulness of trees with diameter 4, *Acta Sci. Natur. Univ. Jilin.*, (1993) 17-22.

[10] A. Krishnaa, A study of the major graph labelings of trees, *Informatica (Vilnius)* **15** (2004) 515-524.

[11] D. Mishra and P. Panigrahi, Graceful lobsters obtained by component moving of diameter four trees, *Comput. Math. Appl.*, **50** (2005) 367-380.

[12] P. Panigrahi and D. Mishra, Graceful lobsters obtained from diameter four trees using partitioning technique, *Ars Combin.*, **87** (2008) 291-320.

[13] A. Rosa, On certain valuations of the vertices of a graph, *Theory of Graphs (Internat. Symposium, Rome, July 1966)*, Gordon and Breach, N. Y. and Dunod Paris (1967) 349-355.

[14] D. A. Sheppard, The factorial representation of major balanced labeled graphs, *Discrete Math.*, **15** (1976) 379-388.

[15] S. L. Zhao, All trees of diameter four are graceful, *Graph Theory and its Applications: East and West (Jinan, 1986)*, 700-706, Ann. New York Acad. Sci., 576, New York Acad. Sci., New York, 1989.