Tree Isomorphism Algorithms.

Oleg Eterevsky **VS.** Arist Kojevnikov

based on

Tree Isomorphism Algorithms:
Speed vs. Clarity

Douglas M. Campbell

Observation 1. *Since a tree isomorphism preserves root and edge incidence, the level number of a vertex (the number of edges between the root and the vertex) is a tree isomorphism invariant.*

Conjecture 1. *Two trees are isomorphic if and only if they have the same number of levels and the same number of vertices on each level.*

Observation 2. *Since a tree isomorphism preserves root and edge incidence, the number of paths from the root to the leaves is a tree isomorphism invariant.*

Conjecture 2. *Two trees are isomorphic if and only if they have the same degree spectrum.*

Observation 3. *Since a tree isomorphism preserves longest paths, the number of levels in a tree (the longest path) is a tree isomorphism invariant.*

Conjecture 3. *Two trees are isomorphic if and only if they have the same degree spectrum at each level.*

Observation 4. *The number of leaf descendants of a vertex and the level number of a vertex are both tree tree isomorphism invariants.*

AHU algorithm

Input: trees T_1 and T_2.

1. Assign to all leaves of T_1 and T_2 the integer 0.

2. Inductively, assume that all vertices of T_1 and T_2 at level $i-1$ have been assigned integers. Assume L_1 is a list of the vertices of T_1 at level $i-1$ sorted by non-decreasing value of the assigned integers. Assume L_2 is the corresponding list for T_2.

3. Assign to the non-leaves of T_1 at level i a tuple of integers by scanning the list L_1 from left to right and performing the following actions: For each vertex on list L_1 take the integer assigned to v to be the next component of the tuple associated with the father of v. On completion of this step, each non-leaf w of T_1 at level i will have a tuple $(i_1, i_2, \ldots, i_k)$ associated with it, where $i_1, i_2, \ldots, i_k$ are integers, in non-decreasing order, associated with the sons of w. Let S_1 be the sequence of tuples created for the vertices of T_1 on level i.

4. Repeat step 3 for T_2 and let S_2 be the sequence of tuples created for the vertices of T_2 on level i.

5. Sort S_1 and S_2 lexicographically. Let S_1' and S_2' respectively, be the sorted sequences of tuples.

6. If S_1' and S_2' are not identical then halt; the trees are not isomorphic. Otherwise, assign the integer 1 to those vertices of T_1 on level i represented by the first distinct tuple on S_1', assign the integer 2 to the vertices represented by the second distinct tuple, and so on. As these integers are assigned to the vertices of T_1 on level i, make a list L_1 of the vertices so

assigned. Append to the front of L_1 all leaves of T_1 on level i. Let L_2 be the corresponding list of vertices of T_2. These two list can now be used for the assignment of tuples to vertices of level $i+1$ by returning to step 3.

7. If the roots of T_1 and T_2 are assigned the same integer, T_1 and T_2 are isomorphic.

```
Post_Order_Version_One(v : vertex)
Begin

    if v is childless then

       Give v the tuple name (O)

    else

       begin

       For each child w of v do

         Post_Order_Version_One(w);

       Concatenate the names of all the children
       of v to temp;

       Give v the name (temp);

       end

end
```

```
Post_Order_Version_Two(v : vertex)
Begin

    if v is childless then

       Give v the tuple name 10


    else

       begin

       For each child w of v do

         Post_Order_Version_Two(w);

       Sort the names of the children of v;

       Set temp to the concatenation if v's sorted
       children's names;

       Give v the tuple name 1temp0;
```

```
        end

end
```

Observation 5. *Induction on the level number proves that a vertex's canonical name is a tree isomorphism invariant.*

Observation 6. *Two trees are isomorphic if and only if their roots have identical canonical names.*

Observation 7. *For all levels i, the canonical name of level i is a tree isomorphism invariant.*

Observation 8. *Two trees T_1 and T_2 are isomorphic if and only if for all levels i, the canonical level names of T_1 and T_2 are identical.*

```
Tree_Isomorphism(T1,T2 : trees)
Begin
  Assign all vertices of T1 and T2 to level numbers lists
  and let hi be the largest level number in Ti;

  If h1 <> h2 then
    write('trees are not isomorphic'); Halt;
  else
    set h to h1; {h1 = h2}

  { process from bottom to top level }
  for i := h downto 0
  begin
    { assign vertices their string name }
    For all vertices v of level i do
      If v is a leaf then
        assign v the string 10
      Else
        assign v the tuple 1i1i2...ik0, where i1i2...ik
        are the strings associated with the children
        of v in non-decreasing order;
      { assign vertices to temporary sorting lists }
      For all vertices v of level i do
        If v belongs to Tj then
          add v's string to Tj(i);
      Sort T1(i) and T2(i) lexicographically;
      If T1(i) <> T2(i) then
        write('trees are not isomorphic t level',i); Halt;

      { assign condensed canonical names }
      For all vertices v of level i do
        If v is the k-th element in Tj(i) then
          assign v the binary string for the integer k
  end;
  write('the trees are isomorphic');
end
```