Code Jam 2008 - APAC Semifinal

Analysis: Modern Art Plagiarism

This problem is a classical graph problem called *subtree isomorphism*. As an interesting note, the modern era in art history actually starts far earlier than the so called classical period in computer science.

In this problem, we are given two trees $\mathbf{T_1}$ and $\mathbf{T_2}$. We are going to decide if T_2 is isomorphic to any subtree of T_1. The general sub-graph isomorphism problem is notoriously hard. But as you perhaps have seen many times, when the things come to trees, it is very solvable. The problem is actually well studied, and is a standard exercise in algorithm design.

First, a bit of terminology. Just for convenience, in our discussion, for two rooted trees, we say one *fits into* the other if there us an isomorphism that maps the former to a subtree of the latter such that the root is mapped to the other's root.

We may fix T_2 and consider it as a tree rooted at vertex 0. We do not know which vertex of T_1 corresponds to 0 of T_2 in the isomorphism. But we may try each vertex in T_1, and see if T_2 fits into T_1.
For a concrete example, let's say we root T_1 at vertex **x**. Assume that there are 3 children of 0 in T_2 -- y_1, y_2, and y_3. Assume that there are 5 children of x in T_1 -- x_1. x_2, x_3. T_2 fits into T_1 if and only if we can find that the subtree at y_1 fits into the subtree at x_i for some i, the subtree at y_2 fits into the subtree at x_j for some other $j \neq i$, and the subtree at y_3 fits into the subtree at x_k for some $k \neq i$, $k \neq j$.

The solution for this problem as follows. Once we have fixed the root x, each vertex has a level in its tree. For each vertex **u** in T_1 and **v** in T_2, if they have the same level (just a bit of reasonable optimization, not necessary for this problem), we want to decide if the subtree at v fits into the subtree at u. We do this from bottom up, the deeper levels first.
For any such pair (u, v) with children $\{ u_i \mid i = 1,2,... \}$ and $\{ v_j \mid j = 1,2,... \}$, we know which v_j fits into which u_i since we are doing the computations bottom up. We find a fit if and only if we can find, for each v_j, a distinct u_i such that v_j fits into u_i. This is clearly a bipartite graph matching problem.

We leave it an exercise to prove that the algorithm, runs in $O(N^2M^2)$ time. There are algorithms with better complexities. For interested readers, we refer to the following paper
R. Shamir, D. Tsur, "*Faster Subtree Isomorphism*", Journal of Algorithm, 33, 267-280 (1999).
which contains a recent result as well as references to earlier works.

More information

Subtree Isomorphism - Graph Isomorphism - Bipartite Matching