Isomorphism of Trees and Planar Graphs

An instance of *Isomorphism* consists of two graphs G_1 and G_2. We are asked to determine if there is a bijection f from $V(G_1)$ to $V(G_2)$ such that xy is an edge of G_1 precisely if $f(x)f(y)$ is an edge of G_2.

An instance of *Labelled Isomorphism* consists of two graphs G_1 and G_2 and an integer label for each edge or vertex. We are asked to determine if there is a bijection f from $V(G_1)$ to $V(G_2)$ such that xy is an edge of G_1 precisely if $f(x)f(y)$ is an edge of G_2, and such that (i) for every vertex x of G_1, the label of x and $f(x)$ agree, and (ii) for every edge xy of G_1, the labels of xy and $f(x)f(y)$ agree. In an instance of *Rooted Isomorphism* there are two labels, one of which appears on exactly one vertex (and no edges) of each graph. Thus we have a root vertex in each graph and we want to find an isomorphism which maps the root of G_1 to the root of G_2.

To begin, we consider rooted isomorphism on trees. For each node t, in one of the two trees, we let T_t be the subtree consisting of t and its descendants. We can label each node by its distance from the root, by traversing the trees using a depth first search. We can build a list L_i of the set S_i of nodes at distance i from the root of the tree they are in, We actually determine for every two vertices u and v of L_i, whether or not there is a rooted isomorphism between T_u and T_v which maps u to v. More strongly, we partition S_i up into k non-empty equivalence classes, for some k, such that two vertices u and v are in the same equivalence class precisely if T_u and T_v are isomorphic. We then k-colour the elements of S_i so that elements in the same equivalence class get the same colour (which we also call a label).

Having done this for S_{i+1}, we proceed on S_i as follows. We can think of each vertex of S_i with l children as a multiset of l of the labels used on S_i where T_u and T_v have a rooted isomorphism precisely if these multisets are the same. We can use a bucketsort, to order the list, for every u in S_i, in nondecreasing order, in $O(|S_{i+1}|)$ time. We then partition L_i up into smallers list so that the elements in each sublist have the same number of children, in $O(|S_{i+1}|)$ time. We can partition each of these smaller lists further into sublists such that T_u and T_v are isomorphic if and only if they are in the same lists by using the ordered lists of length l corresponding to the labels of the children of a node as pointers into an l-dimensional array (without spending the time to zero the array) whose elements are pointers (possibly NIL) to lists containing all the nodes whose list is that indexing the element of the array,as discussed in class, in linear time.

Now, as discussed in class, every tree T either contains a unique edge xy such that the two components of T-xy each contain exactly half the vertices, or contains a unique vertex v such that every component of T-v has fewer than half the vertices. Furthermore, we can find such a vertex or edge in $O(|V|)$ time as follows . We root the tree and traverse it in post-order. We compute for each node u in turn, the number of nodes in T_u by adding 1 to the sum of the corresponding values for its children. We let x be the first node for which this sum is at least $|V|/2$. If the sum is exactly |V|/2 then we let y be the parent of x and return xy, otherwise we set v=x and return v.

Now if two trees are isomorphic, we will obtain either an edge xy for both trees, or a vertex v for both trees. Furthermore, any isomorphism maps the set of (1 or 2) vertices returned for the first graph to the set returned by the second. Thus we can test if the trees are isomorphic by rooting the first tree at one of the vertices returned, and testing for a rooted isomorphism, to the rooted trees obtained by rooting the second tree at one of the (at most 2) vertices returned. The trees will be isomorphic if and only if one of these two rooted isomorphisms exists. So we can test for isomorphism

We turn now to isomorphism on planar graphs. A graph is planar if it can be drawn in the plane so that none of its edges cross. In fact, if a graph has such an embedding It has one in which all the edges are straight lines. We remark that a planar graph has fewer than $3|V|$ edges.

Deleting the vertices and edges of a drawing of a planar graph G leaves a set of regions know as faces. If G is 2-connected then each face is bounded by a cycle of G. Our embedding also yields a cyclic ordering on the edges of G incident to a vertex *v*, which is the order they appear in when we traverse *v* in a clockwise direction. The set of these cyclic orders is an *orientation* scheme for G. If G is a subdivision of a 3-connected graph then the set of cycles bounding its faces are unique and it has two orientation schemes, one is obtained from the other by reversing the ordering at every vertex.

Thus, in order to test whether two planar graphs are isomorphic, we can actually test for two pairs of (graph, rotation scheme), whether there is an isomorphism between the two graphs which fixes the rotation scheme (i.e. so that for every v and pair of nighbours u and w of v, vw immediately follows vu in the circular order s around v precisely if f(v)f(w) immediately follows f(v)f(u) in the rotation scheme around f(v).

In order to do so, for each edge *xy* of G_1 and edge *vw* of G_2 we can, as discussed below, in O*(|V|)* time, determine whether there is an isomorphism *f* from G_1 to G_2 with *f(x)=v* and *f(y)=w* and if so find the unique such *f*. We can then test if such an *f* respects the labeling in O(*|V|*) time. By fixing *xy* and trying all choices for *vw*, we can solve isomorphism for pairs of 3-conncted planar graphs, for each of which we have an orientation scheme, in $O(|V|^2)$ time.

We note that if for some edge xy of G_1 and edge uv of G_2 , we are to have f(x)=v and f(y)=w, then w can follow the cyclic order around v and x to determine the image of every neighbour of v (and x). We can do the same for every neighbour of w. Since G_1 is connected we will eventually fix f entirely. Details were presented in class.