***Proposed Algorithm to Calculate the Automorphism Sets of a Graph.***

1. Maintain a data structure that places each of the vertices into an automorphism set.
2. Initialize this data structure to have each of the vertices in the same automorphism set.
3. Remove all tags from the graph. For each node that was tagged, create a new automorphism set for the number of tags of each kind that were attached to the node. For example, if the vertex had a tag of 3, two tags of 2 and a tag of 5, you would have a specific automorphism class which only contains the vertices which had (in the original graph) 2,2,3,5 as tags. Note that this is not recursive; do not do it multiple times, and don't count anything as a tag if it is not in the original, generic equivalency class. After this step, you should have a graph where every node is EITHER in the non-original equivalency set, OR, it has degree of 2 or greater.
4. Note that step 3 is not operative, and it is very possible that the algorithm will work without it, but it does us the favor of allowing us to tag the graph and be sure of that uniqueness, which may turn out to be theoretically important.
5. Using the existence of the automorphism sets as a starting point, differentiate them to the fullest degree possible using the Paths Invariant.
6. You now may have some non-singleton automorphism groups. For each of these groups, perform the following operations, once ought be enough.
7. For each pair of vertices within a non-empty automorphism group, create two paired graphs, augmentations of the original, with each of the members of the examined pair tagged in one of the two graphs with a singleton node.
8. For each of the newly created augmented graphs, calculate the paths invariant up to the guaranteed maximum differentiating power. Compare these values. If they are distinct, the two nodes are !!!not!!! automorphic. Separate the equivalence class accordingly, using the graph's paths invariant that is "greater" (as an orderable numerical object) to classify which vertex belongs in each set.
9. According to my hypothesis, this fully determines the automorphism classes of the graph, making isomorphism trivial.

This would still be in polynomial time. Why: We are doing Paths at most N^2 times, so total running time would be on the order of O(N^5).