Graph Isomorphism

The problem of graph isomorphism is one that has been discussed for many years.
Many people have worked on various aspects of it, but as for a bound on the time complexity of an algorithm to determine whether two graphs are isomorphic, the most recent breakthrough is due to the mathematician László Babai, who showed that the problem can done in "quasipolynomial" time ($\exp(\ln^{O(1)} n)$) in this paper.

I demonstrate my own much much much less elegant (but straightforward) algorithm that runs in worse than factorial time.

Setup

I define a Graph as an object which encapsulates a `std::vector<std::vector<bool>>`, representing the adjacency matrix of the graph.
I use booleans assuming that that maximum number of edges between any pair of vertices is one.
This Graph object is constructed from a string which represents the rows of the adjacency matrix, separated by commas.
I overload `operator<<` to display the matrix, and then `operator==` to check whether two graph are equal, asserting that any isomorphic graphs are also equal.

Implementation

Formally, two graphs G and H are isomorphic if
$\exists f: V(G) \to V(H), \forall uv \in E(G) \Leftrightarrow f(u) f(v) \in E(H)$
Which is kind of opaque to think about.
Computationally, it's more obvious that this describes the existence of a "vertex permutation" from G to H.
essentially, if we label the vertices of G from 1 to n (assuming they both have n vertices), we need to find a way to permute this labelling such that matrix of G becomes the matrix of H.

Upon realizing this we can use an implementation of Heap's Algorithm (NB: this only generates permutations of an array, it's not related to graphs directly in any way) to generate the permutations and check them each manually.
It's useful to realize at this point that parts of our matrix representation are redundant, and we can ignore the main diagonal because we are assuming that no "loops" are present, and we only need to check one side of this diagonal, because we assume an un-directed graph, so we have symmetry along this diagonal.

Emmanuel Katwikirize

Program Output