

Re: Is graph isomorphism in P?

Source: <http://sci.tech-archive.net/Archive/sci.math/2007-07/msg03382.html>

- *From:* Proginoskes <CHeckman@xxxxxxxx>
 - *Date:* Sat, 21 Jul 2007 03:18:40 -0000
-

On Jul 20, 4:11 am, Fra <fra.cristi...@xxxxxxxx> wrote:

On 20 Lug, 06:30, Proginoskes <CHeck...@xxxxxxxx> wrote:

Fra states: "In this site you will not find the algorithm or any explanation of it, I wish to use this forum in order to place in it hard instances of the MI problem that otherwise would require a prohibitively computation-time to solve it." This makes it hard for people to (1) find two graphs which are isomorphic, for which the algorithm fails to find the isomorphism, or (2) verify that the algorithm runs in polynomial time.

(1) in this page: <http://isoproblem.freeforums.org/viewtopic.php?t=5>

I've suggested how to generate random adjacency matrices

This is a bad way of checking whether a problem is in P or not. For instance, if you choose a random matrix, then the vast majority of the time you can find its clique size in polynomial time. (This has been proven formally, btw.) This is because there are a small number of graphs which are "pathological", so most of the time, you can solve NP-complete problems in polynomial time.

This is also why the average running time is not a good measure; over 99% of the time, the running time is polynomial, which leads to a polynomial average running time.

of bipartite graph (I means undirected graph). It requires only bit of seconds with octave; the only difficulty is to obtain matrices with no duplicate rows.
This problem can be avoided increasing the maximum numbers of '1' in each row in this octave instruction:
from:
M=round(randerr(300,300,3));
to

Re: Is graph isomorphism in P?

$M = \text{round}(\text{randerr}(300, 300, 5));$

for the (2)'s answer see below.

Fra invites people to post graphs to test his isomorphism program, adding: "please not post matrices with more than 350/400 rows, this because I've limited computing resources (only my laptop with 1,2 Gb RAM on AMD Athlon 64) to run the algorithm." Well, Fra, the isomorphism problem, where you only look at graphs with at most 400 vertices, can be solved in CONSTANT TIME.

Oh yes?! Please let me know how to find an isomorphism for this MI instance: <http://isoproblem.freeforums.org/viewtopic.php?t=19> in CONSTANT TIME.

Simple. Try all $400!$ permutations of the vertices of the first graph and see whether you get the second graph (which also takes a constant number of operations: $m \cdot O(n) = O(n^3) = O(400^3) = O(1)$, which is constant time). If any work, return YES; otherwise, return NO. Since $400!$ is a constant, this is a constant-time algorithm.

Theoretical computer science, when looking at running time, only concerns itself when the size of the problem is arbitrarily large.

I never meant that my algorithm can be solve GI problem with at most 400 vertices; I mean that: having only 1,2Gb of Ram, because algorithm is based on exploration of mathematical objects whose demanded of memory increases with vertices increasing, if I try to solve an MI instance for a 2000×2000 I'll get "Could not reserve enough space for object heap" by the java virtual machine.

Anyway, I've no problem to solve 600×600 or 1000×1000 MI instances, let me some hours and I will post here: <http://isoproblem.freeforums.org/viewforum.php?f=1> also a 600×600 MI problem on adjacency matrix of bipartite graphs.

Once again, I'm not interested in "random" graphs, since results concerning them are already known and established.

If Fra is afraid that he's found a polynomial-time algorithm for testing graph isomorphism and someone will steal it, he should realize that posting it to Usenet (where the post will be dated) and/or

Re: Is graph isomorphism in P?

sending himself a copy of that algorithm as a letter in the mail ("poor man's copyright") are both ways to later prove that he did come up with the algorithm first.

Sorry, but this is your personal interpretation.
The idea is that in this phase I wish to test the algorithm for "hard instances of the MI problem that otherwise would require a prohibitively computation-time to solve it".

It's a shame that Graph Isomorphism isn't NP-complete, because what I would do then would be to ask you to look at the graphs which arise from transforming instances of 3-SAT into Graph-Isomorphism, which are the graphs that everyone (concerned with P vs. NP, that is) is really interested in. Random graphs won't do it.

(it is written here: <http://isoproblem.freeforums.org/viewtopic.php?t=5>)
This indirectly can prove the polynomial-time nature of the algorithm.
(I.E. if for a 600x600 bipartite graph the world's fastest isomorphism testing program, that is Nauty, will need of thousand of years and I will give you an isomorphism in 6/7 hours, I think that this is really significant!)

Not as much as you'd expect. Physical time is a bad measure of how fast an algorithm runs, since computers run on different platforms with different operating systems with different compilers. I would rather see a $O(n^k)$ estimate of the running time.

Lastly, he says: "During my research I've found that is possible to give an algebraic characterization for every matrix involved in the MI problem and I've used it to solve the problem with a P-time algorithm (implemented in java)." This Java code should definitely be made public.

At the end of test phase, if the algorithm never fails, I'll publish all.

Once again, a "fast" algorithm, in terms of physical running time, will never prove that your algorithm is in P, because you can only test a finite number of cases. (You can find a polynomial upper bound for the running times of a finite number of problem instances, even for provably exponential-time algorithms.)

To show that your algorithm is polynomial-time, you need to find

Re: Is graph isomorphism in P?

constants k , C , and n , so that your algorithm (running on two graphs with N vertices) requires at most $C \cdot N^k$ steps, where $N \geq n$, and determines correctly whether there is an isomorphism. That requires a specific description of the algorithm and a running time analysis and a proof of correctness.

Nothing less will show that Graph-Isomorphism is in P. (These aren't my rules; they're everyone's rules.)

--- Christopher Heckman

.