

# Re: New algorithm for the isomorphism of graphs

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math/2007-07/msg04957.html>

---

- *From:* Proginoskes <CHeckman@xxxxxxxx>
  - *Date:* Sun, 29 Jul 2007 04:59:43 -0000
- 

On Jul 28, 12:48 pm, mimouni <mimouni.moha...@xxxxxxxx> wrote:

On 28 juil, 06:12, Proginoskes <CHeck...@xxxxxxxx> wrote:

On Jul 27, 7:08 pm, i.am.a.w...@xxxxxxxxxxxxxxxxxxxx wrote:

mimouni gave another trivial example:

$G=3D(12 ;24 ;34 ;36 ;45 ;56)$  and  $G'=3D(ab ;ae ;bc ;cf ;df ;ef)$

In  $G$ , 1 is the only node of lowest order (1), so we are forced to pick it first. Then 2 is the only node adjacent to 1, so it must be picked next. Then 4 is the only node adjacent to what was picked before, so it must be picked next. Next, 6 is unique as the only remaining node not adjacent to any already picked, so it must be picked next. Finally there's a symmetry between 3 and 5, so we can pick either arbitrarily.

In  $G'$ , d is the only node of lowest order (1), so we are forced to pick it first. [Hence 1 must map to d.] Then f is the only node

## Re: New algorithm for the isomorphism of graphs

adjacent to d, so it must be picked next. [Hence 2 must map to f.]

But then there are two nodes (c,e) each adjacent to f, and two nodes (a,b) neither adjacent to f, so the two graphs aren't isomorphic, because there's nothing like 4 that it can map to.

Nothing as complicated as your algorithm is needed to decide this example.

Please try to find a pair of graphs whereby my simpleminded algorithm isn't sufficient (to avoid backtracking) and your more complicated algorithm works better (avoids backtracking entirely, or reduces the amount of backtracking compared to my algorithm).

The graphs posted were intended to show an example of how the algorithm worked, seeing as the OP's native language is not English. If you want a challenge, try:

1 : 2 3 4 5  
2 : 1 6 7 8  
3 : 1 9 10 11  
4 : 1 12 13 14  
5 : 1 15 16 17  
6 : 2 18 19 20  
7 : 2 21 22 23  
8 : 2 24 25 26  
9 : 3 18 21 24  
10 : 3 19 22 25  
11 : 3 20 23 27  
12 : 4 18 22 28  
13 : 4 19 23 26  
14 : 4 21 25 27  
15 : 5 18 26 27  
16 : 5 20 22 24  
17 : 5 23 25 28  
18 : 6 9 12 15  
19 : 6 10 13 29  
20 : 6 11 16 30  
21 : 7 9 14 30  
22 : 7 10 12 16

Re: New algorithm for the isomorphism of graphs

23 : 7 11 13 17  
24 : 8 9 16 29  
25 : 8 10 14 17  
26 : 8 13 15 30  
27 : 11 14 15 29  
28 : 12 17 29 30  
29 : 19 24 27 28  
30 : 20 21 26 28

and

1 : 2 3 4 5  
2 : 1 6 7 8  
3 : 1 9 10 11  
4 : 1 12 13 14  
5 : 1 15 16 17  
6 : 2 18 19 20  
7 : 2 21 22 23  
8 : 2 24 25 26  
9 : 3 18 21 24  
10 : 3 19 22 25  
11 : 3 20 23 27  
12 : 4 18 23 26  
13 : 4 19 24 27  
14 : 4 20 25 28  
15 : 5 20 22 26  
16 : 5 21 25 27  
17 : 5 23 24 28  
18 : 6 9 12 29  
19 : 6 10 13 30  
20 : 6 11 14 15  
21 : 7 9 16 30  
22 : 7 10 15 29  
23 : 7 11 12 17  
24 : 8 9 13 17  
25 : 8 10 14 16  
26 : 8 12 15 30  
27 : 11 13 16 29  
28 : 14 17 29 30  
29 : 18 22 27 28  
30 : 19 21 26 28

The two graphs are not isomorphism (except error of manual processing data).  
Proves is in a Word file with 7 pages.

## Re: New algorithm for the isomorphism of graphs

Okay; now you need to count how many operations were needed to deduce this. (An introduction to analysis of running time of algorithms and "big O notation" is given from here to the end of the post. If you have the background, you can ignore the rest of the post.)

An operation is an arithmetic operation  $+$ ,  $*$ ,  $-$ ,  $/$ ,  $^$ , accessing an indexed variable, assigning a value to a variable, or comparing two variables. (I think that list should have everything on it that you are doing.) When doing this calculation, you should use the symbol  $N$  for the number of vertices; that is, give an answer involving  $N$ , not just an integer.

You can use "big O" notation here, even partway through the calculation, which means you can drop the lower-degree terms of a polynomial, and you can replace constants (in front of powers of  $N$ ) with 1. So for instance,

$$3N^4 + 4N^3 - 8N + 27$$

becomes  $N^4$ . (Remove the  $4N^3$ ,  $-8N$ , and  $27$  terms to get  $3N^4$ ; then replace the 3 with a 1.)

Also, you can "big O" your polynomials partway through.

EXAMPLE: if you are comparing the first element  $a[1]$  of an array with the next  $N+5$  elements of that array, you will end up with the number of operations being  $N$ . This is because the task will look like the following when written out:

```
for i from 2 to N+6 do
if a[1] = a[i] then return YES
return NO
```

(The only real difference is what programming language you're using, which tells you what you need to put in place of "for", "if", "then", "return", etc.)

Starting on the inside, the if statement runs with a constant number of operations; that constant will depend on how "fussy" you get. The important thing, though, is that this answer does not depend on  $N$ , and so, once you "big O" it, you end up with a running time of 1.

Then you are doing this running time of 1 for at most  $N+6$  iterations, so the total number of operations is  $1*(N+6)$ , which reduces to  $N$ . (Or, once again, you can insist that you keep track of what happens to the value of  $i$ , and you might end up with an expression like  $(N+6)*3+1$ , but in the end, you'll end up with  $N$ .)

Finally, when you do the for loop, then the return statement, the total number of operations is  $N+1$ , which reduces to  $N$ .

## Re: New algorithm for the isomorphism of graphs

Note that the final answer is  $N$  raised to a constant ( $N^1$ ), so the code shown above runs in polynomial time, with respect to  $N$ . (If your final function turns out to be  $N$ , the code is sometimes said to run in "linear time", which is considered to be optimal for any problem.) If I had gotten a final answer of  $2^N$ , then the algorithm would not run in polynomial time (with respect to  $N$ ).

--- Christopher Heckman

.