Re: Challenae question for mathematician

Source: http://sci.tech−archive.net/Archive/sci.math/2006−02/msg03108.html

* *From*: mareg@xxxxxxxxxxxxxxxxxxxxxxxxx ()
* *Date*: Sat, 18 Feb 2006 12:56:43 +0000 (UTC)

In article <1140206726.824817.293700@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx>,
"Arturo Magidin" <magidin@xxxxxxxxxxxxxxxxx> writes:

> C6L1V@xxxxxxx wrote:
>
>> Arturo Magidin wrote:
>
> [.snip.]
>
>>> If we let S = { (1,2), (2,3), (3,4), ..., (n−1,n) }.
>>>
>>> We can express (a)(b)^{−1} = (3,4) as a product of elements of S by
>>>
>>> (3,4) = (3,4).
>>>
>>> You can express (c)(b)^{−1} = (3,5) as a product of elements of S by
>>>
>>> (3,5) = (3,5)(4,5)(3,4) [composing right to left]
>>>
>>> For an element x of S_n, define its S−length to be the least number of
>>> factors needed to express x as a product of elements of S.
>>
>> Just as a matter of interest, how does one determine the S−length,
>> since there are many distinct product representations of the same S?
>
> I confess that I am not particularly well−versed in algorithmic
> problems relating to permutation groups. I know that's a weak point in
> my proposal, if no good algorithmic way exists to do so.
>
> Certainly you can do it in exponential time, since there is an easy way
> to express any cycle as products of elements of S, which gives you an

easy upper bound for length_S(x) for any x (add the lengths of the cycles).

Define the "distance" from x to y, d(x,y) to be the length of xy^{−1}.

Is this the same as the minimum number of factors needed to get from x to y?

It would be: say you can get from x to y by applying s_1, s_2, s_3,.... s_k. That means that

s_1*s_2*...*s_k x = y

from which you get

xy^{−1} = s_k^{−1} * ... * s_1^{−1} = s_k * ... * s_1

so the number of transpositions needed to convert x into y is greater than or equal to d(x,y). And if d(x,y) = k, then xy^{−1} = s_1 * ... * s_k for some s_i in S, from which you get s_k*...*s_1*x = y, hence you need at most d(x,y) transpositions to get from x to y.

Actually, it's not difficult to calculate a decomposition of minimal length for a given permutation. Roughly speaking, first use (1,2), (2,3), etc to map whatever is supposed to map to 1 to 1, then use (2,3), (3,4) to map whatever should map to 2 to 2, and so on. As an example, let's take a random permutation (1,4,3,5)(2,6) of S_6. This equals

(5,6) (3,4)(4,5)(5,6) (2,3)(3,4)(4,5)(5,6) (1,2)(2,3)(3,4)(4,5)
*** maps 5 to 1 ***
************* maps 6 to 2 ************
******************** maps 4 to 3 ***********************
**************** also maps 1 to 4 ***********************
************************** maps 3 to 5 **********************
********************* also maps 2 to 6 **********************

The longest element in S_n has length n(n−1)/2 and is the order reversing permuation (1,n)(2,n−1)....

This is probably what you would do in practice if you were asked to get a row of books into order, and you were only allowed to interchange pairs of adjacent books. You might decide first to move the book that should be at the left end into place, then the next one, and so on. (I think this is called "bubble sort" − it is a quadratic time algorithm, which is not very efficient as a sorting method − efficient sorting is O(n log n).)

I can't think of a completely elementary proof that this method yields a decomposition of minimal length. I can prove it using a result from the theory of Coxeter groups, which says that if a word in the generators does not have minimal length, then you can always remove two of the generators in the word to give another word for the same group element. It is not hard to see that you cannot do that with words of the form above, so they must have minimal length.

Derek Holt.
.