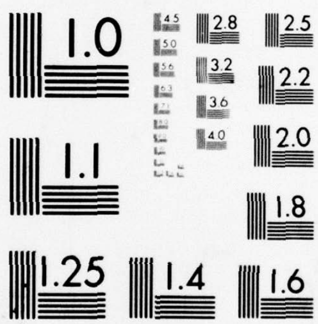
AD-A053 246 YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE F/G 12/1
THE COMPLEXITY OF WORD AND ISOMORPHISM PROBLEMS FOR FINITE GROU--ETC(U)
MAR 77 R J LIPTON, L SNYDER, Y ZALCSTEIN N00014-75-C-0752
UNCLASSIFIED RR-91 NL

1 OF 1
AD
A053246

END
DATE
FILMED
6-78
DDC



12

LUX ET VERITAS

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

DISCLAIMER NOTICE

The Complexity of Word and Isomorphism Problems for Finite Groups

(Preliminary Report)

R. J. Lipton,[†] L. Snyder,[†] and Y. Zalcstein[††]

Research Report #91

[†] Department of Computer Science, Yale University, New Haven, Connecticut 06520. Supported in part by ONR Grant N00014-75-C-0752.

[††] Computer Science Department, State University of New York, Stony Brook, New York 11794. Supported in part by NSF Grant DCR-75-01998.

The Complexity of Word and Isomorphism Problems for Finite Groups
(Preliminary Report)

R. J. LIPTON,[†] L. SNYDER,[†] and Y. ZALCSTEIN[††]

1. INTRODUCTION

In this paper we begin a study of the complexity of the word and ismorphism problems for finite groups. There are several specific reasons for studying these questions:

(i) Both word and ismorphism problems have practical interest. Such diverse areas as chemistry and the theory of simple groups require the solution of these problems.
(2) Since word problems are closely related to questions of language recognition, insight into them should aid in understanding recognition problems.
(3) Isomorphism problems for groups are interesting in that they are related to the well known question of graph isomorphism [5].

Thus, there is sufficient motivation for studying the complexity of finite groups. The rest of this paper contains an outline of our main results.

Our model of computation is the well known model of multitape deterministic Turing machines [1]. We will be interested in both the time and the space requirements of our algorithms.

A comment about our choice of model is in order. Indeed a reasonable question appears to be: why not use a random access computer rather than Turing machines? The main reason is that all the word problems considered here could then be done in linear time (on a random access computer). However this is a misleading result. For very large groups -- the kind currently being handled in a number of applications -- it is misleading to allow random access to the very large group multiplication tables. On the other hand, Turing machines charge a proper amount for each random access. Consequently, our result provide a more accurate accounting of costs.

2. WORD PROBLEMS

Let us consider the more general problem of evaluation of words in some groupoid [4]. More exactly we assume that we are given an input tape in the form

$$R_1 \ldots R_n \quad W_1 \ldots W_k$$

where $R_1 \ldots R_n$ represents* the $n \times n$ multiplication table for the groupoids binary operation $\circ$ and $W_1 \ldots W_k$ are k elements from the groupoid to be multiplied from left to right. Note each element of the groupoid uses $\log n$ space; the entire input tape takes

$$T = n^2 \log n + k \log n$$

space. We wish to study the time required to compute $W_1 \circ \ldots \circ W_k$. Our main result is:

Theorem: The evaluation of $W_1 \circ \ldots \circ W_k$ can be done in

(1) $O(T^2)$ in an arbitary groupoid;

(2) $O(T \log^2 T)$ in an arbitrary semigroup;

(3) $O(T \log T)$ in an arbitrary abelian group.

Essentially this theorem demonstrates how algebraic structure can be used to decrease the complexity of the word problem. In order to evaluate $W_1 \circ \ldots \circ W_k$ the multiplication table must be repeatedly accessed. Thus the above theorem demonstrates that we can organize our accesses to this table in a more efficient manner as more structure is placed on the table. In this regard note that $O(T^2)$ for an arbitrary groupoid corresponds to k scans across the table, i.e. no accesses are avoided.

We now will sketch the proofs of (2) and (3) in some detail.

We will now show how to get $O(T \log^2 T)$ in an arbitrary semigroup. The presence of the

† Department of Computer Science, Yale University, New Haven, Connecticut 06520. Supported in part by ONR grant N0014-75-C-0752.

†† Computer Science Department, State University of New York, Stony Brook, New York 11794.
Supported in part by NSF grant DCR-75-01998.

* R_i is the i^{th} row of this table.

associative law allows us to perform many products in "parallel", i.e. we can avoid costly repeated scans of the $n\times n$ multiplication table. The algorithm procedes as follows: (we assume that k is a power of 2 with at most a cost of 2)

(a) Form the pairs $(W_1,W_2)\dots(W_{k-1},W_k)$.

(b) Sort the pairs into $(x_1,x_2)\dots(x_{k-1},x_k)$ such that (x_{i_1},x_{i_2}) precedes (x_{j_1},x_{j_2}) iff $i_1<j_1$ or ($i_1=j_1$ and $i_2<j_2$).

(c) In one scan through the $n\times n$ table perform all these k/2 products to form $z_1\dots z_{k/2}$.

(d) Now "unsort" $z_1\dots z_{k/2}$ so that we obtain $W_1 \circ W_2 \dots W_{k-1} \circ W_k$. We can do this just by keeping a tag along with the pairs $(W_1,W_2)\dots(W_{k-1},W_k)$ <u>and</u> using a <u>stable</u> sort [3].

(e) If $k/2>1$, then recursively call (a); otherwise halt.

The time for this algorithm is:

(a) and (b) can be done in $O(k \log k \log n)$

(c) is $O(n^2 \log n)$

(d) is $O(k \log k \log n)$

(e) we recursively call log k times.

Thus the algorithm runs in at most $k \log^2 k \log n$ time; it is therefore bounded by $O(T \log^2 T)$.

We next will show how to get $O(T \log T)$ for an arbitrary abelian group. The algorithm depends on some nontrivial but elementary group theory, and is considerably more involved than the semigroup case so only a sketch of the construction is presented. Part (a) forms the generators using Lagrange's theorem to bound the iterations. In (b) the group elements are represented using the generators and (c) computes the product.

a. First we construct the group generators $x_{i_1},\dots,x_{i_m}$ from the $R_1\dots R_n$ table. We use Lagrange's theorem to guarentee that $m \le \log n$. The procedure will be iterated and at stage j, M_j will be a table containing those elements not yet included in generated, (initially, M_1 contains all group elements in group order.)

Stage j:

(i) Select x_j the first non-identity element of M_j.

(ii) Find the x_j row in the $R_1\dots R_j$ table and call it R_j.

(iii) Construct the *jth* coset table. The structure of this table is as follows.

The first entries are $1,x_j,x_j^2,\dots,x_j^{r-1}$ which may be easily computed using only the row R_j. Each of these is marked in M_j. The next unmarked element, y of M_j is the coset leader in the next sequence of entries $y,yx,yx^2,\dots,yx^{r-1}$ which can be computed wholly in R_j. These elements are also marked. This continues until all elements of M_j are marked. The table M_{j+1} is formed from the coset leaders of this stage and the procedure continues to stage j+1 with M_{j+1} in group order. The result of all stages is shown in figure 1.

$$
\begin{array}{llll}
1\; x_1\; x_1^2 \dots x_1^r & \left.\begin{array}{l} y_1\; y_1x_1\; y_1x_1^2 \dots y_1x_1^r \\ y_2\; y_2x_1\; y_2x_1^2 \dots y_2x_1^r \\ \quad\dots \\ y_u\; y_ux_1\; y_ux_1^2 \dots y_ux_1^r \end{array}\right\} & \text{stage 1} \\
1\; y_1\; y_1^2 \dots y_1^s & \left.\begin{array}{l} w_1\; w_1y_1\; w_1y_1^2 \dots w_1y_1^s \\ w_2\; w_2y_1\; w_2y_1^2 \dots w_2y_1^s \\ \quad\dots \\ w_v\; w_vy_1\; w_vy_1^2 \dots w_vy_1^s \end{array}\right\} & \text{stage 2} \\
\quad\dots & & \\
1\; z_1\; z_1^2 \dots z_1^t & \qquad\qquad\qquad\qquad \} & \text{stage m}
\end{array}
$$

Figure 1. Data structure produced by stage a(iii).

b. In this step we construct the generator representation for the group elements. The data structure of Figure 1 is somewhat over simplified in that we need to save more information than simply the yx^t entries. We suppose that the actual entry is a triple $<yx^t,y,t>$ called a descriptor with the first field called the element field and the second field called the leader field. We keep an auxiliary tape to contain the generator representation. Clearly n records of log n fields each of at most log n bits are required to hold the exponents for each generator of each group element.

Iteration 1: The stage 2 portion of figure 1 is sorted into group order by the element field of their description. This results in the representation of the coset leaders of stage 1 being ordered in the order that the coset leaders are given in stage 1. The elements of stage 1 are now transferred to the auxiliary tape with the coset leaders of each entry replaced by their stage 2 representation. This can be done in one scan of stage 1 by sequencing through the stage 1 and sorted stage 2 tables in unison. The result is that all group elements are given in terms of 2 generators. To complete this stage the auxiliary tape is resorted into group order on the leader field of the description.

Iteration ℓ: The ℓ+1 entries of Figure 1 are sorted into group order by the element field of the descriptor. The representation of the coset leaders of each entry on the auxiliary tape are changed in a single scan to reflect their representation given by stage ℓ+1. The auxiliary tape is resorted on the leader field of the descriptor.

c. The result of part b yields the generator representation of the group elements. In this part we produce the product. First we reduce $W_1 \ldots W_k$ to a product of group elements raised to powers, i.e.

$$x_1^{\varepsilon_1} x_2^{\varepsilon_2} \ldots x_n^{\varepsilon_n} .$$

Now we use the generator representation of the x_i's to produce the product. An m field workspace is used with the *jth* field containing the present power of the *jth* generator. In a sequential scan of the auxiliary tape the representation of x_i is found, its generator exponents multiplied by ε_i and the results added to the workspace. The size of each workspace position is bounded by the order of the element. Finally, one last scan through the auxiliary tape will locate the desired result.

For timing we recall that the number of generators is $m \le \log n$. The dominant term in the computation is a sort required in (c) to collect the $W_1 \ldots W_k$ into powers which counts k log k log n.

3. ISOMORPHISM PROBLEMS

Second, we will consider the isomorphism of finite groups. Our first result is

Theorem: The isomorphism problem for groups can be solved in polylogspace, i.e. it can be solved in $c \log^2 T$ (c is a constant) space where T is the length of the input tape that encodes the multiplication tables of the two groups.

This result (also observed independently by Gary Miller and M. O. Rabin) shows that if this isomorphism problem was NP-complete [2], then all of NP would be in polyspace. This is therefore one piece of evidence that it may not be NP-complete.

Our second result is

Theorem: The isomorphism for finite abelian groups can be solved in polynomial time.

This result relies heavily, of course, on the fundamental theorem of abelian groups [4].

Before stating our final result we need one definition. Let G_k be the class of all groups that can be generated by sets with cardinality at most k. For an interesting class of groups in G_2 we note that a deep conjecture of group theory states that all simple groups are in G_2.

Theorem: The isomorphism problem for groups in G_k (k fixed) is in nondeterministic and hence polynomial time. Moreover, it is in deterministic logspace provided deterministic logspace equals nondeterministic logspace.

REFERENCES

[1] J. Hopcroft and J. Ullman. *Formal Languages and Their Relation to Automata.* Addison-Wesley, 1969.
[2] R. Karp. Reducibility among combinatorial problems in R. Miller and J. Thatcher editors, *Complexity of Computer Computations.* Plenum Press, 1972.
[3] D. Knuth. *The Art of Computer Programming, Vol. III.* Addison-Wesley, 1973.
[4] J. Rotman. *The theory of Groups: An introduction.* Allyn and Bacon, second edition, 1973.
[5] G. Miller. Private communication.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 14 RR-91

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER: 9

4. TITLE (and Subtitle): 6 The complexity of word and isomorphism problems for finite groups.

5. TYPE OF REPORT & PERIOD COVERED: Technical rept.

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): 10 Richard J. Lipton, Lawrence Snyder, Y. Zalcstein

8. CONTRACT OR GRANT NUMBER(s): 15 N00014-75-C-0752, NSF-DCR-75-12998

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Yale University
Department of Computer Science
10 Hillhouse Ave, New Haven, CT 06520

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Office of Naval Research
Information Systems Program
Arlington, Virginia 22217

12. REPORT DATE: 11 MAR 1977

13. NUMBER OF PAGES

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 12 6p.

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this report is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

finite groups
isomorphism
word problem
complexity
algebraic structure

k-squared

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The uniform word problem for finite groups presented by their multiplication tables is considered. Upper bounds of $O(k^2)$ for arbitrary group and $O(n \log^2 n)$ for arbitrary semigroup and $O(n \log n)$ for abelian groups are shown where n is the length of the presentation.

log-squared n

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

407051