J. Korean Math. Soc. **38** (2001), No. 4, pp. 763–780

COMPUTERS IN ALGEBRA: NEW ANSWERS, NEW QUESTIONS

CHERYL E. PRAEGER

ABSTRACT. The use and development of computer technology by algebraists over the last forty years has revolutionised the way in which algebraists think about algebra, and the way they teach it and conduct their research. This paper is a personal reflection on these changes by a somewhat unwilling computer user.

1. Introduction

This paper is a personal reflection by a somewhat reluctant computer user on the crucial role played by the computer in algebra research over the past thirty years, and its likely future importance. Thirty years is the extent of my personal knowledge, but computers have been used innovatively in algebra for more than forty years, and accounts of the very early period can be found in [43, 57]. First came answers to several mathematical questions which demonstrated the power of computers. This served to raise new questions thereby inspiring established researchers, young mathematicians and computer scientists to channel their energies into designing new algorithms and new computer algebra systems. As a result the number-crunching devices of the 1970's were transformed into sophisticated oracles which seem almost to understand the way an algebraist thinks.

This metamorphosis in technology occurred in parallel with an equally dramatic change in the way algebraists think about their subject and the way they conduct their research. On the one hand computers facilitate mathematical discovery, and on the other hand they can be integral to proving theorems. It is impossible to overemphasise the impact of computer systems such as MAGMA [12] and GAP [54] on the professional lives of algebraists internationally, both on their teaching and on their

Received October 19, 2000.
2000 Mathematics Subject Classification: 20B40, 20C40.
Key words and phrases: computational algebra, computational group theory.

research. All of my research students use these computer systems as essential learning tools to explore new concepts. They demand illustrative examples to examine by computer to aid their understanding.

Answers to mathematical questions raised new questions which in turn inspired new conceptual breakthroughs. Computers have become an experimental tool for exploring new concepts and structures in algebra, for spotting patterns, and for suggesting new conjectures and theorems. Demand for higher performance and greater mathematical capabilities for investigating larger and more complex mathematical structures was the impetus for new algorithm development. This in turn highlighted the importance of complexity analysis and statistical analysis for understanding the performance of algorithms. Moreover the use of computers in algebra, and the mathematics developed to support it, have led to new areas of algebraic research which integrate and build on diverse areas of mathematics. Computers have become an indispensable tool at the forefront of cutting-edge research in algebra.

Because of my knowledge and experience the paper will focus on group theoretic illustrations rather than examples from other areas of algebra. Also, because I will concentrate on areas with which I have had some personal involvement I will make very little mention of several important lines of development related to computational group theory, such as computation with finitely presented groups [10] and crystallographic groups [13], computation with characters and representations of groups [23, 44], and computational complexity [1, 6, 37].

2. Computers as labour saving devices

My initial undergraduate education did not include any computer courses. There were none available. My first introduction to computers was a short course on the then new computer language Fortran IV at the Australian National University in December 1968. I was spending my summer at the Australian National University on a Vacation Scholarship at the end of my third year of undergraduate study. I found the Fortran language interesting, but the computer interface tedious and irritating. The only mathematically significant applications suggested as exercises during the course were two procedures to generate prime numbers. Our programs were presented to the computer in a stack of punched cards, and we could have our cards run through the computer once each day. I decided that an application which needed the use of computers would

have to be *very important to me* to warrant the tedium of daily debugging of a stack of punched cards.

As a research student in Oxford in the early 1970's I recognised the value of computers as labour saving devices for performing tasks that I could easily see how to do by hand but which could be done faster when automated. Examples of this were procedures for finding all solutions for certain divisibility conditions or linear inequalities over some restricted range of parameter values. My D. Phil. supervisor Peter Neumann had an ongoing 'spare-time' research program to classify the primitive permutation groups of prime degree less than 100. One purpose of this exercise was to test the power of the existing theory of permutation groups, and another purpose no doubt was to sharpen the understanding and expertise of his research students by involving them in some of the cases. It was good fun working together with Peter on this programme. I remember a group of us going to the University's computer centre to submit our program on punched cards. We would wait until our allotted 10 seconds of CPU time was used up, and re-cycle the cards for another run with the values of the parameters adjusted slightly. It was an enjoyable social experience, but I did not regard it as serious mathematical work.

3. Permutation groups and simple groups

The first time I experienced a sense of awe at what could be achieved in algebra using computers was in 1973. I was attending a course of lectures by Charles Sims in Oxford. The highlight of these lectures was Sims' description of his construction [56] of the Lyons-Sims sporadic simple group of order

$$51,765,179,004,000,000 = 2^8 \cdot 3^7 \cdot 5^6 \cdot 7 \cdot 11 \cdot 31 \cdot 37 \cdot 67$$

as a group of permutations of a set of size 8,835,156. Work of Richard Lyons [38] predicted that such a group might exist, but although a great deal could be worked out theoretically about its structure, without an explicit construction, neither its existence nor its uniqueness could be proved.

Richard Lyons' investigation was part of the programme to classify finite simple groups according to the structure of their involution centralisers. An involution t is an element of order 2, that is, $t \neq 1$ and $t^2 = 1$. According to a pivotal result of Feit and Thompson [21] all finite groups of odd order are soluble, and hence each finite non-abelian simple

group G contains at least one involution t. The centraliser $C_G(t)$ is the collection of all the elements $g \in G$ such that $gt = tg$. An important family of possibilities to be studied were those where $C_G(t)$ is a perfect extension of a group of order 2 by an alternating group A_n. The cases where $n \neq 11$ had been dealt with in work of Brauer and Suzuki, Janko and Thompson, Janko and Wong, and Lyons himself (see [38] for an account of this) showing that of these cases only $n = 8$ gave rise to an involution centraliser in a simple group, namely in McLaughlin's simple group (see [30]). Lyons used character theory and group representation theory in the remaining case $n = 11$ to calculate the order of G and its character table. He was also able to determine two permutation representations of G, of degrees $9,606,125$ and $8,835,156$, the respective point stabilisers being a non-split extension of a group of order 3 by the automorphism group of McLaughlin's group, and the Dickson-Chevalley group $G_2(5)$.

Sims used the latter permutation representation for his construction. The mathematical ideas he used formed the basis for future computer computations with permutation groups. A naive way to represent a permutation π of the set $\{1, \ldots, n\}$ on a computer is as a sequence $(x_1, \ldots, x_n)$, where $x_i = \pi(i)$. With the computers available at the time it was not possible even to store a single permutation of $8,835,156$ points in this way, much less multiply two such permutations. To represent and compute with such permutations Sims used a *base* of G, that is a small subset of the $8,835,156$ points such that two different permutations in the group G produced different images of the base. In this way elements of the group G could be represented on the computer by their 'base images'. Sims knew from the work of Lyons that G would have a small base, thereby making his computations feasible and efficient.

This striking success led Sims to develop further and disseminate his method, now known as the *Schreier/Sims* algorithm, for finding and using a base and corresponding strong generating set for computations with permutation groups. At the first international conference on 'Computational problems in abstract algebra', held in Oxford in 1967, Sims [55, Section 4] spoke about his algorithm and made available an implemented version of it. In 1973 he made available to attendees of his lectures in Oxford a full-scale implementation. Another implementation was made by Jeff Leon in 1975 and described in [35], and a complexity analysis of this algorithm was given by Furst, Hopcroft and Luks [22] in

1980, proving that it would run in polynomial time $O(n^6)$ for permutation groups of degree n. A good description can be found in the book of Greg Butler [15, Chapters 13 and 14].

4. Computers and the Burnside Problem

According to Burnside [14] in 1902, "a still undecided point in the theory of discontinuous groups is whether the order of a group may be not finite, while the order of every operation it contains is finite". He went on to state a special form of the question which is now known as the Burnside Problem, and about which several hundred research papers have been written, see [47]. The largest group generated by r elements such that $g^n = 1$ for every element g is now called the *Burnside group of exponent n with r generators* and is denoted $B(r, n)$.

The Burnside Problem: Which of the groups $B(r, n)$ are finite?

Hall [24, Chapter 18] reported on the state of knowledge at the time of writing his book: $B(r, n)$ was known to be finite for $n = 2, 3, 4, 6$ and all r, but the exact order of $B(r, 4)$ was not known for $r \geq 3$. Over a number of years upper bounds for the order $|B(3, 4)|$ of $B(3, 4)$ were derived by hand. The lowest of these 2^{69} obtained in 1973 by Narain Gupta and Mike Newman, was proved by Bayes, Kautsky and Wamsley [11], using a computer, to be equal to $|B(3, 4)|$.

I. D. Macdonald [39] was the first to use a computer to get information about the orders of Burnside groups. He developed appropriate descriptions of groups of prime power order to compute with them efficiently. The origins of these descriptions, now called power-commutator presentations, go back to work of Sylow in the 19th century. It was Wamsley's improvements to Macdonald's procedures that enabled the determination of $|B(3, 4)|$. Havas and Newman then took up the challenge of determining $|B(4, 4)|$. Making the significant theoretical and implementational improvements needed to achieve this took a couple of years. As a young postdoctoral research fellow at the Australian National University I listened to weekly working sessions as this theory was being developed in 1973-74. It led to the determination in 1975 that $|B(4, 4)| = 2^{422}$ and eventually to the proof (see [59]) in 1988 that $|B(5, 4)| = 2^{2728}$. From the spectacular success of these methods new group theoretic and algorithmic questions and challenges emerged.

Havas and Newman [25] developed a suite of programs for computing with power-commutator presentations of groups which is known now

as the ANU p-quotient program. A theory for computing with graded Lie rings was developed with Vaughan-Lee [26, 59] which led to theoretical bounds on the nilpotency class of the largest finite r-generator groups of exponent 5. Laue, Neubüser and Schoenwaelder [32] built on Newman's ideas for describing subgroups of a soluble group in their development of the SOGOS system (soluble groups' operating system) for interactive computing with soluble groups, complete with its own command language SOGOL (soluble groups' operating language). Newman's ideas also contributed to the development by Holt and Plesken [27] of algorithms for enumerating finite perfect groups. Yet another strand was a program due to Newman [46] effectively solving the isomorphism problem for groups of prime power order and leading to the systematic enumeration [48, 49, 50] of finite p-groups.

5. Comprehensive computer systems

During the 1970's those at the forefront of group computations, especially Joachim Neubüser in Aachen and John Cannon in Sydney, saw the potential benefits of making available to the mathematical community some general purpose computer systems which included the most up-to-date algorithms for computing with groups and which had a 'user-friendly interface'. These systems would be written in a computer language which would make sense to a professional group theorist. The first general systems developed were the SOGOS system in Aachen mentioned already for computing with soluble groups, the CAS system [44], also in Aachen, for computing group characters, and the system CAYLEY developed by Cannon [16] which included in particular many procedures for computing with permutation groups. They were followed in the 1980's and 1990's by the systems GAP [43, 54] developed by Neubüser, Schönert and others in Aachen and MAGMA [12] developed by Cannon in Sydney.

These systems were developed because their authors had a vision of how computers could enhance progress and understanding about groups, and not because of an overwhelming demand by the mathematical community. However these systems were the most important agents for cultural change in group theory in the twentieth century. For the first time novice computer users had the chance to experiment with computer calculations without the enormous time and energy investment needed to attain strong programming skills. The systems made it easy to make computations with moderate-sized groups, for example, in the

mid-1980's it was feasible using CAYLEY (see [16]) to compute the order of a permutation group of degree 10,000. It was possible to develop and implement an algorithm using the high level computer languages in these systems designed especially for ease of use by group theorists. Thus more mathematicians developed algorithms in order to solve their mathematical problems, and contributed their algorithms for inclusion in the systems.

The systems remain at the cutting-edge of new developments in algebra because leading researchers permit their most up-to-date algorithms to be included in them. In fact it is now a matter of pride to announce that one's algorithms are available in the systems GAP or MAGMA.

My interaction and knowledge of these developments began in a peripheral way. I remember meeting John Cannon in the 1970's while I was working at the Australian National University. He took copies of my notes of Charles Sims' Oxford lectures, and asked numerous questions about my work on elements of prime order in primitive permutation groups. The questions continued sporadically for a number of years, and I became aware that John was carefully monitoring theoretical results in this area for their applicability to algorithm development.

In 1983 I was appointed as a full professor, and I felt strongly that mathematics courses should make appropriate use of new technology. I also had enough self-knowledge to understand that I would need to develop a research interest in computational aspects of mathematics to ensure sufficient energy was devoted to achieving this.

John Cannon generously gave two weeks of his time in 1986 to run workshops in Perth on CAYLEY. He explained many of the algorithms and gave us some insights into what procedures were time-costly and which ones were fast. I responded to a challenge from John to find an efficient algorithm for computing the kernel of a homomorphism between two finite abelian groups. Thus commenced my first research involvement in computational group theory. The end-point of this first foray was a general study of computing with group homomorphisms in a joint paper [34] in 1991 with Charles Leedham-Green and Leonard Soicher. I felt a terrible novice entering this new research area, and am grateful to John Cannon and Mike Newman as well as my co-authors for their advice and help in levering me into the field.

John also discussed with me the occasionally erratic behaviour of a randomised algorithm for recognising whether the group generated by a given set of permutations was the full alternating or symmetric group. The alternating and symmetric groups A_n and S_n have no short bases

and so the deterministic methods of Sims for permutation group computations did not work well for these groups G if n was large. Efficient procedures existed for checking that such groups were transitive, and even that they were primitive. The basic idea of the recognition algorithm was to make several independent random selections of elements from G. If G really was the full alternating or symmetric group, then with high probability elements would be found which could not lie in any proper primitive subgroup, typically elements of prime order with few non-trivial cycles and lots of fixed points. Theory suggested that the algorithm should work extremely well. However the practical tests, as John showed me in several computer print-outs, indicated problems with finding satisfactory practical methods for making approximately random selections from a permutation group, given only a set of generators. The issue of random selection continues to be a crucial one in practice as for an increasing number of problems there is no practical deterministic algorithm to solve them, only a randomised one which depends on making random selections of elements from a group.

6. Matrix group algorithms

In 1988, at the suggestion I believe of John Cannon, I received an invitation to the Computational Group Theory week at the Mathematics Research Institute in Oberwolfach, Germany. It was exciting to meet the principal researchers in the area internationally, and to hear the latest reports on their research. However the single most important event to influence the direction of my future research in computational group theory was a discussion with Joachim Neubüser after dinner one evening. He bemoaned the fact that, whereas efficient algorithms existed for performing most of the important computational tasks for groups of permutations and finite soluble groups, this was not the case for matrix groups over a finite field. This was a serious defect in the current computer systems since finite groups were often represented as matrix groups. There was available a procedure called the MEATAXE developed by Parker [52] to find subspaces invariant under a given matrix group, and if both the dimension and the field size were small enough, then permutation group algorithms could be used to study matrix groups. However in general it was not even possible, he said, to decide whether the group generated by a given set of nonsingular $n \times n$ matrices over a finite field $\mathbb{F}$ contained the special linear group $\mathrm{SL}_n(\mathbb{F})$ (the group of all $n \times n$ determinant 1 matrices over $\mathbb{F}$).

Not knowing a great deal about computational group theory, I found this surprising and, influenced by my discussions with John Cannon about the randomised recognition algorithm for alternating and symmetric groups, I wondered whether a similar approach might be possible to recognise matrix groups containing $\mathrm{SL}_n(\mathbb{F})$. I explained the approach to Peter Neumann who was also present when Neubüser made his comments. What was needed was some subset of matrices in $\mathrm{GL}_n(\mathbb{F})$ with two important properties. First the subset needed to be sufficiently large that there was a good likelihood that matrices from the subset would be obtained after several random selections from any matrix group containing $\mathrm{SL}_n(\mathbb{F})$; and we would need an efficient procedure for deciding whether a matrix belonged to the subset. Secondly it was desirable that relatively few matrix groups, other than groups containing $\mathrm{SL}_n(\mathbb{F})$, would contain any matrices in the subset; and we would need efficient procedures for recognising these exceptional matrix groups. We rather quickly identified candidate subsets. We proposed to use two subsets, each large enough to have the first property, and we believed that very few matrix groups contained matrices from both subsets.

It took more than a year to verify that two slightly modified classes had all the required properties, and it took another year to write down all the details [45]. The algorithm was surprisingly simple in form and was implemented successfully by Holt and Rees [28]. Justifying the correctness of the algorithm required some deep group theoretic arguments relying on the finite simple group classification. Its simple form allowed us to give a complete complexity analysis. The SL-recognition algorithm, as it was called, may be regarded as a prototype for subsequent matrix group algorithms, as it exhibits a wide range of features common to later algorithms.

In particular, it was a 'one-sided Monte Carlo algorithm'. This means that a positive answer '*The group contains* $\mathrm{SL}_n(\mathbb{F})$' was guaranteed to be correct, but if a group containing $\mathrm{SL}_n(\mathbb{F})$ was submitted to the algorithm there would be a small chance that appropriate matrices would not be found by the random selection and hence the algorithm would not report that the group contained $\mathrm{SL}_n(\mathbb{F})$. The design of the algorithm was such that the probability of the latter occurring could be made as small as desired by the user.

The comments above assume that it is possible to make independent uniform random selections of elements from a group. However in practice the information available about the group is just a set of generating matrices. Finding an optimal method for making approximately

random, and approximately independent, selections from a group is a matter of on-going research and debate. A general method for doing this for an arbitrary finite group was given in a theorem of Babai [2], but in the context of the SL-recognition algorithm (see [53, p. 190]) the cost of doing this was $O(n^{10})$ field operations compared with an estimated cost of $O(n^5)$ (or in a typical case $O(n^4)$) field operations for the remainder of the algorithm. An algorithm developed in [17] in response especially to the requirements of computation with matrix groups works well in practice and has been analysed asymptotically. This latter algorithm, and more generally the question of random selection from groups has been critiqued and investigated by computer scientists and statisticians as well as algebraists [3, 8, 9, 18, 19, 20, 51].

A concerted research effort by a number of mathematicians world-wide has resulted in a coherent collection of computer algorithms for matrix group computations implemented both as a share package [33] within GAP and also in MAGMA. However more breakthroughs are needed before matrix group computation has similar capabilities to computation with permutation groups.

7. Use of GAP and MAGMA for problem solving

The computer systems GAP and MAGMA are now used widely as tools for exploring algebraic and combinatorial structures. The success of these systems was in part due to the growing acceptance by algebraists of the new 'algebraic' computer languages in which they were written. As their use became more wide-spread, many users expressed the wish for equally easy access to other specific purpose programs, for example programs developed for combinatorial, or number theoretic, or character theoretic applications. The benefits for research were becoming obvious. Various strategies were used to achieve this in both systems, and I will give an example of a development in which I was involved as a beneficiary.

A group theory system X which the user had learnt to use might send a command to another computer program Y which in turn would complete a computation and return the result to the user through the system X. This might make available to the user the program Y with which she was unfamiliar. She would only need to learn a few new commands in the language of system X rather than become proficient in using program Y.

This idea was put into practice by Leonard Soicher in the early 1990's to facilitate computations with automorphism groups of graphs. He used as system X the group theory system GAP and as programs Y his own coset enumeration program and the graph isomorphism program nauty of Brendan McKay [41]. His aim was to enable the user to construct a graph admitting a given group of automorphisms, to use nauty to compute its full automorphism group, and then to compute further with the graph and automorphism group in the system GAP. The outcome was Soicher's share package GRAPE [58] for the GAP system.

During the development of GRAPE Leonard Soicher visited Perth. At that time Alice Miller and I were struggling with a problem about vertex-transitive graphs which were not Cayley graphs. The *Cayley graph* for a group G relative to a self-inverse subset S of G is the graph with elements of G as vertices, and edges the pairs $\{x, sx\}$, for $x \in G$ and $s \in S$. Such a graph admits G acting by right multiplication as a vertex-transitive subgroup of automorphisms, so all Cayley graphs are vertex-transitive. However there exist vertex-transitive graphs which are not Cayley graphs, the smallest being the Petersen graph on 10 vertices which has automorphism group S_5. Marušič [40] had asked: for which positive integers n does there exist a vertex-transitive graph on n vertices, which is not a Cayley graph. The subset of such positive integers n is closed under multiplication, and so attempts to answer this question naturally focussed on integers with few prime divisors.

At the time of our investigation the simplest case for which the answer was not known was the case where $n = 2pq$ with p and q distinct primes congruent to 3 modulo 4. If a minimal vertex-transitive subgroup G of automorphisms was regular (that is, had trivial stabilisers) then the graph would be a Cayley graph for G. Thus our strategy was to examine all minimal transitive permutation groups of degree $2pq$, and investigate the vertex-transitive graphs admitting them to decide whether or not their full automorphism groups contained a regular subgroup. This was 'easier said than done'. Our analysis led to several explicit families of minimal transitive permutation groups, but we had great difficulties completing the task of examining the corresponding graphs. Leonard's prototype program GRAPE turned out to be the ideal tool for exploring these groups and their graphs. Our detailed computer 'interrogation' of small members of these families led in one instance to the insights necessary to construct a new family of non-Cayley vertex-transitive graphs and, for another family of minimal transitive groups, to a proof that all graphs admitting groups from the family were Cayley graphs (see [42]).

We were enormously grateful to Leonard for making his system available to us. Moreover our frequent discussions may have helped shape the design of GRAPE.

8. And so we reach the 21st Century

Early triumphs involving computer computations with groups, such as those discussed in Sections 3 and 4, focussed on getting answers to computationally difficult mathematical problems. For these pioneering researchers the challenge was:

'Can I get an answer to this specific mathematical question?'

Their spectacular success generated new questions and new challenges. With the development of general purpose computer systems an additional question was added.

'Can this general class of questions be answered efficiently in practice?'

New capabilities were added to the systems GAP and MAGMA which extended their ranges to include parts of combinatorics, number theory, or representation theory. Other new systems [10, 29] introduced graphical interfaces to help the user.

Researchers in theoretical computer science, studying problems such as graph-isomorphism testing, realised that it was essential to understand the computational complexity of group-theoretic problems, see [36]. For them the questions were different.

'How much time does this algorithm require?'

'What is the optimal algorithm to use for this computation?'

In particular, they wanted to know what group computations could be completed in a time which is a polynomial function of the size of the input. Their work was conducted independently of the developments which led to the computer systems CAYLEY, MAGMA and GAP. Striking contributions were made to a theoretical understanding of the complexity of algorithms contained in these systems and new polynomial time algorithms were developed [6, 22, 31, 37]. Moreover some of the new theoretical algorithms [5, 4, 7] suggested that practical gains in performance might be achievable, for example, for computations on permutation groups with a small base, or by integrating a Monte Carlo algorithm as an option in implemented algorithms.

Communication and cooperation between researchers interested principally in complexity issues and those concerned mainly with practically efficient computation were facilitated by international meetings during

the 1990's. Each side threw down its gauntlet with a challenge for the other. The demand was that in any algorithm development, attention should be given to all the questions above. Many responded positively. For example, Seress implemented in GAP many of the fast algorithms for small base permutation groups, and others of us learnt how to express the complexity of our algorithms in the accepted language for complexity analyses.

There is now a general expectation that new practical algorithms should be accompanied by an analysis of their complexity as well as a proof of their correctness. There is also an expectation that new theoretical algorithms should be accompanied by a critique of their practicality, and where appropriate a plan for their implementation. Thus in an ideal world all new algorithms will come with a proof of their correctness, a complexity analysis, and if appropriate a full implementation. Moreover their practical performance will be in line with that predicted by their complexity analysis.

Will the 21st century bring this ideal world? What might be our vision for the next century? I will end by sketching part of mine.

The 21st century will bring further integration in mathematics. Already we have seen the benefits of complexity analysis and statistical theory in critiquing existing algorithms. Deep results from group theory and representation theory have allowed algorithms to become simpler by turning some of them into essentially expert oracles. Probability distributions on groups have been exploited to design randomised algorithms. Developing the next generation of group theoretic algorithms will bring together many areas of mathematics.

I do not have any clear idea of what the group theoretic algorithms or computer systems of the 21st century will be like. Perhaps in the next century someone will succeed in building a quantum computer. If this happens then algebraists will be ready to design and implement highly parallel, non-deterministic algorithms which will revolutionise computer algebra.

Already group theorists recognise that non-trivial use of a system such as GAP or MAGMA in their research means that they are exploiting high level intellectual property developed by their colleagues. They acknowledge such use by referencing these systems in their research publications. However we are still working towards a satisfactory way to acknowledge and reward research for which the output is a widely-used and acclaimed software package rather than a book or journal article. Indeed the systems GAP and MAGMA on which so much current research

in group theory depends have no long-term guarantee of continued funding. Their future is not secure. My vision for the 21st century is that computer systems like these will be essential research tools for algebraists. Researchers will continue to donate freely the algorithms they design and implement, and this together with financial support from many national research councils will ensure that these systems always contain the most powerful algorithms known and are affordable by all researchers and students.

We have not yet found the best way to have a completely successful use of computers in the teaching of algebra although there have been many successful experiments. However our future students, who will have grown up accustomed to the excitement of computer games, will expect a similar ease of use and visual attractiveness from mathematical software packages. Moreover, some of them will have the computing skills and sophistication needed to develop a new generation of computer algebra systems with user interfaces which will make experimenting and exploring with groups as easy and as enjoyable as playing the computer game 'Quake'. We can but dream! The extraordinary developments of the 20th century in the use of computers in algebra will be surpassed by even greater achievements in the coming century.

ACKNOWLEDGEMENTS. I am grateful to Derek Holt, Bill Kantor, Gene Luks, Joachim Neubüser, Alice Niemeyer and Mike Newman for their generosity in responding to my request for comments on early drafts of this paper. Their timely responses led to a much enriched, improved and corrected final form of the paper.

References

[1] L. Babai, *Computational complexity in finite groups*, In Proceedings of the International Congress of Mathematicians **I, II** (Kyoto, 1990), 1479–1489, Tokyo, 1991. Math. Soc. Japan.

[2] ———, *Local expansion of vertex-transitive graphs and random generation in finite groups*, In Theory of Computing, pages 164–174, New York, 1991. (Los Angeles, 1991), Association for Computing Machinery.

[3] ———, *Randomization in group algorithms: conceptual questions*, In *Groups and computation, II* (New Brunswick, NJ, 1995), 1–17. Amer. Math. Soc., Providence, RI, 1997.

[4] L. Babai, G. Cooperman, L. Finkelstein, E. M. Luks, and Á. Seress, *Fast Monte Carlo algorithms for permutation groups*, J. Comput. System Sci. **50** (1995), no. 2, 296–308, 23rd Symposium on the Theory of Computing (New Orleans, LA, 1991).

[5] L. Babai, G. Cooperman, L. Finkelstein, and Á. Seress, *Nearly linear time algorithms for permutation groups with a small base*, In Proc. of International Symposium on Symbolic and Algebraic Computation ISSAC '91, pp. 200–209. (Bonn), ACM Press, 1991.

[6] L. Babai, W. M. Kantor, and E. M. Luks, *Computational complexity and the classification of finite simple groups*, Proc. 24th FOCS, 1983, pp. 162–171.

[7] L. Babai, E. M. Luks, and Á. Seress, *Fast management of permutation groups I*, SIAM J. Comput. **26** (1997).

[8] L. Babai and I. Pak, *Strong bias of group generators: an obstacle to the "product replacement algorithm"*, In Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 2000), pages 627–635, New York, 2000. ACM.

[9] A. Baddeley, C. R. Leedham-Green, A. C. Niemeyer, and M. Firth, *Measuring the performance of random element generators in large algebraic structures*, in preparation, 2000.

[10] G. Baumslag and C. F. Miller, III, *Experimenting and computing with infinite groups*, In *Groups and computation, II* (New Brunswick, NJ, 1995), Amer. Math. Soc., Providence, RI, 1997, pp. 19–30.

[11] A. J. Bayes, J. Kautsky, and J. W. Wamsley, *Computation in nilpotent groups (application)*, In Proc. Second Internat. Conf. Theory of Groups of Lecture Notes in Math., Berlin, Heidelberg, New York, Springer-Verlag **372** (1974)(Canberra, 1973), 82–89.

[12] W. Bosma and J. Cannon, *Handbook of* MAGMA *functions*, Department of Pure Mathematics, University of Sydney, 1994.

[13] H. Brown, R. Bülow, J. Neubüser, H. Wondratschek, and H. Zassenhaus, *Crystallographic groups of four-dimensional space* Wiley-Interscience [John Wiley & Sons], New York, Wiley Monographs in Crystallography, 1978.

[14] W. Burnside, *On an unsettled question in the theory of discontinuous groups*, Quart. J. Pure Appl. Math **33** (1902), 230–238.

[15] G. Butler, *Fundamental Algorithms for Permutation Groups*, Lecture Notes in Comput. Sci. Springer-Verlag, Berlin, Heidelberg, New York **559** (1991).

[16] J. Cannon, *A computational toolkit for finite permutation groups*, In Proc. Rutgers Group Theory Year, 1983–1984, pages 1–18, Cambridge, 1984. (Rutgers, 1983-1984), Cambridge University Press.

[17] F. Celler, C. R. Leedham-Green, S. H. Murray, A. C. Niemeyer, and E. A. O'Brien, *Generating random elements of a finite group*, Comm. Algebra **23** (1995), 4931–4948.

[18] P. Diaconis and L. Saloff-Coste, *Comparison techniques for random walk on finite groups*, Ann. Probab. **21** (1993), 2131–2156.

[19] ———, *Moderate growth and random walk on finite groups*, Geom. Funct. Anal. **4** (1994), 1–36.

[20] ———, *Walks on generating sets of abelian groups*, Probab. Theory Related Fields **105** (1996), 393–421.

[21] W. Feit and J. G. Thompson, *Solvability of groups of odd order*, Pacific J. Math. **13** (1963), 775–1029.

[22] M. Furst, J. Hopcroft, and E. Luks, *Polynomial-time algorithms for permutation groups*, In 21st Annual Symposium on Foundations of Computer Science (Syracuse, N.Y., 1980), pages 36–1. IEEE, New York, 1980.

[23] M. Geck, G. Hiss, F. Lübeck, G. Malle, and G. Pfeiffer, *CHEVIE—a system for computing and processing generic character tables*, Appl. Algebra Engrg. Comm. Comput. **7** (1996), no. 3, 175–210, Computational methods in Lie theory (Essen, 1994).

[24] M. Hall, Jr., *The Theory of Groups*, Macmillan Co., New York, 1959.

[25] G. Havas and M. F. Newman, *Application of computers to questions like those of Burnside*, In *Burnside Groups*, volume 806 of Lecture Notes in Math., Berlin, Heidelberg, New York, (Bielefeld, 1977), Springer-Verlag **806** (1980), 211–230.

[26] G. Havas, M. F. Newman, and M. R. Vaughan-Lee, *A nilpotent quotient algorithm for graded lie rings*, J. Symbolic Comput. **9** (1990), 653–664.

[27] D. F. Holt and W. Plesken, *Perfect Groups*, Clarendon Press, Oxford, 1989.

[28] D. F. Holt and S. Rees, *An implementation of the Neumann–Praeger algorithm for the recognition of special linear groups*, J. Experimental Math. **1** (1992), 237–242.

[29] ———, *A graphics system for displaying finite quotients of finitely presented groups*, In *Groups and Computation*, Amer. Math. Soc. DIMACS Series **11** (1993), 113–126 (DIMACS, 1991).

[30] Z. Janko and S. K. Wong, *A characterization of the McLaughlin's simple group*, J. Algebra **20** (1972), 203–225.

[31] W. M. Kantor, *Sylow's theorem in polynomial time*, J. Comput. System. Sci. **30** (1985), 359–394.

[32] R. Laue, J. Neubüser, and U. Schoenwaelder, *Algorithms for finite soluble groups and the SOGOS system*, In *Computational Group Theory*, London, New York (Durham, 1982), Academic Press, 1984, pp. 105–135.

[33] C. R. Leedham-Green, A. C. Niemeyer, E. A. O'Brien, and C. E. Praeger, *Recognising matrix groups over finite fields*, In V. Weispfenning J. Grabmeier, E. Kaltofen, editor, *Computer Algebra, Foundations, Applications, Systems* (to appear).

[34] C. R. Leedham-Green, C. E. Praeger, and L. H. Soicher, *Computing with group homomorphisms*, J. Symbolic Comput. **12** (1991), 527–532.

[35] J. S. Leon, *On an algorithm for finding a base and strong generating set for a group given by generating permutations*, Math. Comp. **20** (1980), 941–974.

[36] E. M. Luks, *Isomorphism of graphs of bounded valence can be tested in polynomial time*, J. Comput. System Sci. **25** (1982), 42–65.

[37] ———, *Permutation groups and polynomial-time computation*, In *Groups and Computation*, Amer. Math. Soc. DIMACS Series **11** (1993), 139–175 (DIMACS, 1991).

[38] R. Lyons, *Evidence for a new finite simple group*, J. Algebra **20** (1972), 540–569.

[39] I. D. Macdonald, *A computer application to finite p-groups*, J. Austral. Math. Soc. Ser. A **17** (1974), 102–112.

[40] D. Marušič, *Cayley properties of vertex symmetric graphs*, Ars Combin. **16B** (1983), 297–302.

[41] B. McKay, *Nauty users guide (version 1.5)*, Technical report, Dept. Comp. Sci., Australian National University, 1990.

[42] A. A. Miller and C. E. Praeger, *Non-Cayley vertex-transitive graphs of order twice the product of two odd primes*, J. Algebraic Combin. **3** (1994), no. 1, 77–111.

[43] J. Neubüser, *An invitation to computational group theory*, In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, *Groups'93 – Galway/St. Andrews* of London Math. Soc. Lecture Note Ser., Cambridge University Press **212** (1995), 457–475.

[44] J. Neubüser, H. Pahlings, and W. Plesken, *CAS; design and use of a system for the handling of characters of finite groups*, In *Computational Group Theory*, London, New York, Academic Press, (1984) (Durham, 1982), 145–183.

[45] P. M. Neumann and C. E. Praeger, *A recognition algorithm for special linear groups*, Proc. London Math. Soc. **65** (1992), no. 3, 555–603.

[46] M. F. Newman, *Determination of groups of prime-power order*, In *Group Theory* of Lecture Notes in Math., Berlin, Heidelberg, New York, Springer-Verlag **573** (1977) (Canberra, 1975), 73–84.

[47] ———, *A still unsettled question*, Mathematics Research Reports, 2000.007, Australian National University, 2000.

[48] M. F. Newman and E. A. O'Brien, *A CAYLEY library for the groups of order dividing 128*, In *Group Theory*, Berlin, New York, Walter de Gruyter, (1989) (Singapore, 1987), pp. 437–442.

[49] E. A. O'Brien, *The p-group generation algorithm*, J. Symbolic Comput. **9** (1990), 677–698.

[50] ———, *The groups of order 256*, J. Algebra **143** (1991), no. 1, 219–235.

[51] I. Pak, *Random walks on finite groups with few random generators*, Electron. J. Probab. **4** (1999), no. 1, 11.

[52] R. A. Parker, *The computer calculation of modular characters (the Meat-Axe)*, In M.D. Atkinson, editor, *Computational Group Theory*, London, New York, Academic Press, (1984) (Durham, 1982), 267–274.

[53] C. E. Praeger, *Computation with matrix groups over finite fields*, In *Groups and Computation*, Amer. Math. Soc. DIMACS Series, DIMACS **11** (1991, 1993), 189–196.

[54] M. Schönert *et al.*, GAP – *Groups, Algorithms and Programming*, Lehrstuhl D für Mathematik, RWTH Aachen, fifth edition, 1995, (http://www.gap.dcs.st-and.ac.uk/gap).

[55] C. C. Sims, *Computational methods in the study of permutation groups*, In *Computational problems in abstract algebra*, Oxford, Pergamon Press, 1970 (Oxford, 1967), pp. 169–183.

[56] ———, *The existence and uniqueness of Lyons' group*, In T. Hagen, M.P. Hale, and E. E. Shult, editors, *Finite Groups '72*, Amsterdam, North Holland, 1973, pp. 138–141.

[57] ———, *Group theoretic algorithms, a survey*, In Proceedings International Conference of Mathematicians, Helsinki, Acad. Sci. Fennica, 1980 (Helsinki, 1978), pp. 979–985.

[58] L. H. Soicher, *GRAPE: a system for computing with graphs and groups*, In *Groups and computation* (New Brunswick, NJ, 1991), Amer. Math. Soc., Providence, RI, 1993, pp. 287–291.

[59] M. R. Vaughan-Lee, *The Restricted Burnside Problem*, London Math. Soc. Monogr. (N.S.), Oxford University Press, Oxford **5** (1990).

Department of Mathematics and Statistics
University of Western Australia
35 Stirling Highway
Crawley WA 6009, Australia