JUL 28 1965

LA-ABC-7160 (Full Paper)
CONF-650505-5

Applications of a Language for Computing in Combinatorics*

MASTER

Mark B. Wells

University of California, Los Alamos Scientific Laboratory,
Los Alamos, New Mexico

An important method widely used in combinatorial work these days is the experimental brute-force approach. Given a problem, we let a computer grind out results any way we know how. Quite often either the initial programming or the first results suggest ways of improving the algorithm. As the experimentation continues, we are constantly looking for patterns in the answers (or in the programming) which might suggest either improvements in our algorithm or, if we are lucky, a non-computational solution to the problem.

Such an approach is feasible only when the programming of the many experiments is not burdensome. If too much attention and time is required in programming details, it is hard to maintain enthusiasm for the project as a whole. This is where language enters.

The Maniac group at Los Alamos now has a language for computing in combinatorics which opens the door for more extensive application of the experimental method. The language is described in detail in Wells [3]. Briefly, the language allows for manipulation of sets (subsets of $\{0,1,2,\dots\}$) and for backtracking -- the basic tool of computational combinatorics -- to be easily expressed.

APPROVED FOR PUBLIC RELEASE

*This work was performed under the auspices of the United States Atomic Energy Commission.

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission, makes any warranty or representation, expressed or implied, with respect to the use of any information, apparatus, method, or process disclosed in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights. In the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor is acting on behalf of the Commission, or employee of such contractor, in the performance of any duty or in the discharge of any responsibility, or in the use of any information, apparatus, method, or process disclosed in this report, or in the use of any information, apparatus, method, or process disclosed in this report, or in the use of any information, apparatus, method, or process disclosed in this report, or in the use of any information, apparatus, method, or process disclosed in this report.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

To illustrate the application of the language, I will describe to you one combinatorial study underway at Los Alamos.

The problem, mentioned by Steinhaus [2], is to find the minimum number of (binary) comparisons necessary to rank n different weights. Let this minimum number be given by $W(n)$. It is not difficult to show that $W(n) \geq [\log_2 n!] + 1 = L(n)$. On the other hand, to date, the best algorithm for ranking the n weights in as few steps as possible is due to Ford and Johnson [1]. Let $F(n)$ be the number of comparisons required by their algorithm. Now, $F(n) = L(n) = W(n)$ for $n = 1, 2, \dots, 11$ and for $n = 20, 21$. Since $F(12) = 30$ and $L(12) = 29$, the first interesting open question is which of the two values is the proper one for $W(12)$. This particular problem was the starting point for an experimental study of the Steinhaus problem.

Given 12 objects of distinct and unknown weight, there are, of course $12!$ possible ordered arrangements of the objects. Each possible comparison, say a with b , divides these arrangements into two sets of arrangements, one set

containing arrangements wherein a is always greater than b and the other containing arrangements with a less than b . Each further comparison divides a set of "permissible" arrangements into two smaller sets and so forth. At any stage of this process, the comparisons so far made yield a partial ordering of the objects, the set of permissible arrangements being identical to the set of permutations of the objects which are consistent with the partial ordering.

A partial ordering may be represented by a (consistent) oriented graph. Let g_i be such a graph arising from i comparisons and let $P(g_i)$ be the number of consistent permutations. The program then calculated, for $i = 0, 1, \dots, 29$, those g_i for which $P(g_i) \leq 2^{29-i}$, that is, those partial orderings which possibly could be completed with $29-i$ comparisons. If no g_{29} 's exist, then we know that $W(12) = 30$, since no strategy could exist to pinpoint an arbitrary arrangement of the objects in 29 steps. On the other hand if a g_{29} exists, we cannot say that $W(12) = 29$, only that a strategy still may exist. Of course, different sequences of comparisons can yield isomorphic graphs, thus, an important part of the program is the detection and deletion of isomorphic partial orderings. The heart of the program, however, is

the calculation of $P(g)$, part of which is given in fig. 1, and which illustrates many of the features of our combinatorial language.

I will not describe this program in detail, but will merely point out pertinent language features. Note, in particular:

1. the backtrack search expressed as an indefinite nest of loops and given notationally by the "with" statement and diagonally slanted dots;
2. the use of the set-theoretic notions "element of", "subtraction", "singleton", "cardinality" and "intersection" in statement #9040;
3. the binomial coefficient (and set cardinality) notation used in #9080.

It turns out that $W(12) = 30$. After 60 hours of running the machine finished testing the unique graph for $i = 24$ and announced that there were no g_{25} 's. That makes the case $n = 13$ the first open question. Unfortunately, though, the exhaustive technique applied here for $n = 12$ is impractical for $n = 13$ since there is so much more "freedom", that is, 2^{33} ($L(13) = 33$) is considerably

larger than 13 factorial. However, experience gained so far in this study makes me feel that it is extremely unlikely that $W(n) = F(n)$, for all n ; in fact, I would not be surprised if $W(13) = 33$.

The positive approach of searching for a strategy which yields a ranking of 13 objects in at most 33 steps is much more difficult than the negative approach taken for $n = 12$, and indeed, a success would be more exciting due to the possibility of generalization. I have initiated some investigations in this area, but have no results to report.

In conclusion, I would like to stress the important role of the (programming) language in such a combinatorial study. I cannot say that without our language such a study could not be done, but I can say that without the language this study would not have been done (at least by me).

References

- [1] L. R. Ford, Jr. and S. M. Johnson, "A tournament problem," American Mathematics Monthly, vol. 66, no. 5, May 1949, p. 387 ff.
- [2] H. Steinhaus, Mathematical Snapshots, Oxford University Press, New York, 1950, p. 36 ff.
- [3] M. B. Wells, "Aspects of a language for combinatorial computing," IEEE Transactions on Electronic Computers, vol. EC-13, no. 4, August, 1964, pp. 431-438.

$S_0 = 0$; (set) $L_0 = n$
 # 9000 with $i = 1, 2, \dots$
 $P_i = 1$
 # 9020 for each component $(L; K)$
 (set) $z = |K_i|$
 # 9040 for $j \in K_i : z \sim \{|C_j \cap K_i|\} \rightarrow z$
 if $z \equiv 0$: $S_i = 1$ go to # 9080 " K_i is a line"
 $S_i = 0$
 # 9060 for each subgraph $(K; L)$ "pair"
 $Q_i = 1$
 $L_i - \{24\} \rightarrow L_i$
 if $L_i = 0$:
 $S_i + P_{i+1} Q_i \rightarrow S_i$
 otherwise:
 $Q_i = P_{i+1}$, nest deeper
 # 9080 $P_i S_i \left(\frac{|A_i| + |K_i|}{|A_i|} \right) \rightarrow P_i$
 parperno () = P_i ; go to exit

FIGURE 1