МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2003
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2003
Proceedings of the Thirty Second Spring Conference of
the Union of Bulgarian Mathematicians
Sunny Beach, April 5–8, 2003

ON SOME APPLICATIONS OF THE CONCEPT OF SET IN COMPUTER SCIENCE COURSE

Krassimir Ia. Iordjev, Stefan M. Stefanov

In this work, the interrelation between mathematics and computer science is discussed, and especially the advantage of having thorough knowledge in mathematics in writing efficient computer algorithms during the computer science course. An example is presented how the concept of a set and operations with sets can be used in computer programming. The problem of obtaining all $n \times n$ matrices of zeroes and ones with exactly k ones in each row and column is considered. It is shown that using sets and operations with sets, an efficient algorithm for solving this problem can be constructed.

The purpose of this work is to show by examples from computer programming the necessity of students' thorough knowledge in mathematics in order to write more efficient computer programs.

In connection with this we think that during the computer programming course, knowledge of estimation of algorithms is especially useful, taking into account the estimation of running time. Using particular examples, the running time of some polynomial and exponential algorithms is given in abstract measures for an increasing parameter (see, e.g. [4] or [9]). A specific question students should answer when discussing a computer program is: how many times a particular operation is executed, where "operation" stands not only for arithmetic and logical operation but also for any other operation which is run during the algorithm performance, for example, call of a subroutine, insertion or deletion of an element of a file, test whether a condition is satisfied, etc. The problem under consideration becomes more complicated and more interesting when we have several embedded cycles. In such a case, mathematical knowledge of operations with finite sums is very useful.

Another – not less important aim we have posed – is to give a meaningful example for using variables of type "set". A classical example is the problem of finding prime numbers via Eratosthenes method [1, 3, 5, 6, 7]. Note that in [7] in the implementation of this algorithm using the Turbo-Pascal algorithmic language, the gross error of considering 1 as a prime number is done. Such an error is not allowable in a problem book because it would lead to confusion and misunderstanding when used by students.

A large class of computer programming problems is reduced to the following: A description of a finite set M is given. Find all elements of M having a particular property. In many specific cases it is not difficult for students to write a program for finding all elements of M and verifying whether the corresponding element possesses this property.

This method for solving such problems is called *explicit enumeration method.* An algorithm implemented via the explicit enumeration method will take time $O(|M|)$, more precisely time $t_1t_2|M|$ where t_1 is the time for obtaining each element of M, t_2 is the time for verifying whether an element possesses the corresponding property, and $|M|$ is the cardinality of M.

Problem 1. Find all square $n \times n$ matrices of zeroes and ones (Boolean matrices) with exactly k $(0 \le k \le n)$ ones in each row and column.

The experiment shows that solving Problem 1 by the explicit enumeration method is not very difficult for students. However, this method is quite slow when n increases.

Denote by B_n the set of all square Boolean matrices of order n and by Λ_n^k the set of all elements of B_n with exactly k ones in each row and each column. Let

$$\beta_n = |B_n|, \quad \lambda_n^k = |\Lambda_n^k|. \tag{1}$$

It is not difficult to show that

$$\beta_n = 2^{n^2}. \tag{2}$$

Therefore an algorithm for solving Problem 1 by the explicit enumeration method will take time $O\left(2^{n^2}\right)$.

Following formulas for obtaining λ_n^k are known:

$$\lambda_n^1 = \lambda_n^{n-1} = n! \tag{3}$$

$$\lambda_n^2 = \lambda_n^{n-2} = \sum_{2k_2+3k_3+\cdots+nk_n=n} \frac{(n!)^2}{\prod\limits_{j\ge 2} k_j!\,(2j)^{k_j}} \tag{4}$$

$$\lambda_n^3 = \lambda_n^{n-3} = \frac{n!^2}{6^n} \sum \frac{(-1)^\beta\,(\beta+3\gamma)!2^\alpha 3^\beta}{\alpha!\beta!\gamma!^2 6^\gamma}, \tag{5}$$

Where the sum is over all $\dfrac{(n+2)(n+1)}{2}$ non-negative solutions of the equation $\alpha+\beta+\gamma = n$. Formula (4) is published in [8] and (5) is published in [2]. Up to now a closed form formula for calculating λ_n^k with $k = 4, 5, \dots$ is not known.

In this paper we propose a considerably quicker algorithm for solving Problem 1. For this purpose, students should have basic knowledge of set theory and combinatorics. We would like here to once again underline the interdisciplinary relation between mathematics and computer science. The next problem is a useful exercise for the simultaneous assimilation of concepts like combinations (in mathematics) and operations with variables and constants of type "set" (in computer science classes).

Problem 2. Write a computer program for obtaining all subsets of k elements of the set $Z_n = \{1, 2, \dots, n\}$, $(0 \le k \le n)$.

As it is known, each subset of a n-element set M could be coded with the use of a n-dimensional Boolean vector (array), where the i-th element of the array is 1 or 0 depending on whether i-th element of M belongs or does not belong to this subset. After such hints, a useful exercise would be the individual task each student to make his/her own package for operating with sets and operations with sets regardless the fact that some programming languages like Turbo-Pascal have built-in resources for this purpose. Thus, students would better perceive the basic principles for constructing such resources.

As a test whether Problem 2 is solved correctly we use the following formula known from mathematics classes:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} = \frac{n\,(n-1)\dots(n-k+1)}{k!}. \tag{6}$$

From (6) it follows that Problem 2 can be solved by an algorithm with running time $O\left(n^k\right)$ and a student, perceived the basic principles of computer programming, would not be handicapped by this problem.

Denote by C_n^k the set of all k-element subsets of the set Z_n (which can be obtained by the computer program of Problem 2) and by D_n^k the Cartesian product

$$D_n^k = \underbrace{C_n^k \times C_n^k \times \dots \times C_n^k}_{n} = \{(A_1, A_2, \cdots, A_n) \mid A_i \in C_n^k, i = 1, 2, \dots, n\} \tag{7}$$

Consider the subset $\mathrm{M}_n^k \subset D_n^k$ of all elements $(A_1, A_2, \dots, A_n)$ of D_n^k such that for each $i = 1, 2, \cdots, n$, the number i is contained in exactly k sets among $A_1, A_2, \dots, A_n$.

The following problem is connected with the topic "data of type set" in the tutorials of computer programming. As it will be seen below, the solution of this problem will give us more "elegant" solution of Problem 1.

Problem 3. Write a program for obtaining all elements of the set M_n^k.

Problem 3 can be solved by using Problem 2 for obtaining all elements of C_n^k. Elements of C_n^k could be written in a file or in a list, from which they could be derived until obtaining all elements of D_n^k and after excluding the unnecessary elements to obtain all elements of the set M_n^k (explicit enumeration method). Of course, operations with files and dynamical structures are studied at a later stage of computer programming education. Fortunately, for educational purposes for comparatively small values of n and k (how much small values depends on the computer available) we can use array of sets. Since we must declare dimension of the array in advance, we need again formula (6).

Essentially, Problem 3 solves Problem 1 because the following proposition holds true.

Theorem 1. *There exists a one-to-one correspondence between element of Λ_n^k and M_n^k.*

Proof. Let $\alpha = (\alpha_{ij}) \in \Lambda_n^k$ and for each $i = 1, 2, \dots, n$, $(\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{in})$ be the i-th row of α in which there are exactly k ones by definition. Consider the set $A_i \subseteq Z_n$ such that $j \in A_i$ if and only if $\alpha_{ij} = 1$. Because in each row of α there are exactly k ones then $A_i \in C_n^k$ and since in each column of α there are exactly k ones then for every $j = 1, 2, \dots, n$, the number j is met exactly in k sets among the obtained sets $A_1, A_2, \dots, A_n$. Therefore there exists an injective mapping from Λ_n^k into M_n^k.

Conversely, let $\mu = (A_1, A_2, \dots, A_n) \in \mathrm{M}_n^k$. For each $i = 1, 2, \dots, n$ construct the n-dimensional Boolean vector $a_i = (\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{in})$ with $\alpha_{ij} = 1$ if and only if $j \in A_i$. Since $A_i \in C_n^k$ then in a_i there are exactly k ones and since for each $j = 1, 2, \dots, n$ the number j is met in exactly k sets among $A_1, A_2, \dots, A_n$, then in each row and column of matrix $\alpha = (\alpha_{ij})$ there are exactly k ones, that is, $\alpha = (\alpha_{ij}) \in \Lambda_n^k$. Therefore there exists injective mapping from M_n^k into Λ_n^k. The proof of Theorem 1 is complete. □

Theorem 1 and its proof give us a way for coding elements of Λ_n^k.

Using the facts that $|D_n^k| = |C_n^k|^n$, $|C_n^k| = \binom{n}{k}$ and formula (6) we conclude that an algorithm based on the explicit enumeration method will take time $O\left(n^{kn}\right)$ for solving Problem 3. Although this algorithm is not polynomial, it is considerably quicker than every algorithm for solving Problem 1 by the explicit enumeration method. The proof of this assertion follows from the following equality:

$$\lim_{n\to\infty}\frac{n^{kn}}{2^{n^2}} = \lim_{n\to\infty}\left(\frac{n^k}{2^n}\right)^n = 0. \tag{8}$$

REFERENCES

[1] D. J. Dahl, E. W. Dijkstra, C. A. R. Hoare. Structured Programming. Academic Press Inc., 1972.

[2] I. Good, J. Grook. The enumeration of arrays and generalization related to contingency tables. *Discrete Math.*, **19** (1977), 23–45.

[3] В. Г. Абрамов, Н. П. Трифонов, Г. Н. Трифонова. Введение в язык Паскаль. Москва, Наука, 1988.

[4] А. Ахо, Дж. Хопкрофт, Дж. Ульман. Построение и анализ вычислительных алгоритмов. Москва, Мир, 1979.

[5] К. Йенсен, Н. Вирт. Паскаль – руководство для пользователя и описание языка. Москва, Финансы и статистика, 1982.

[6] Д. Прайс. Программирование на языке Паскаль: Практическое руководство. Москва, Мир, 1987.

[7] Р. Стоянова, Г. Гочев. Програмиране на Pascal с 99 примерни програми. Учебно помагало (издателство и година не отбелязани).

[8] В. Е. Тараканов. Комбинаторные задачи на бинарных матрицах. Комбинаторный анализ, Москва, изд-во МГУ, 1980, вып. 5, 4-15.

[9] Н. Уирт. Алгоритми + структури от данни = програми. София, Техника, 1980.

Krassimir Iordjev
Stefan M. Stefanov
Department of Informatics
Faculty of Mathematics and Natural Sciences
Neofit Rilski South-West University
Blagoevgrad, Bulgaria
`e-mail: iordjev@yahoo.com`
`e-mail: stefm@aix.swu.bg`

ЗА НЯКОИ ПРИЛОЖЕНИЯ НА ПОНЯТИЕТО МНОЖЕСТВО В КУРСА ПО ПРОГРАМИРАНЕ

Красимир Я. Йорджев, Стефан М. Стефанов

В работата се разисква междупредметната връзка математика – информатика и по-конкретно ползата от задълбочени знания по математика при съставянето на ефективни алгоритми в часовете по информатика. Показан е пример за използването на понятието множество и операции над множества в програмирането. Разгледана е задачата за получаване на всички $n \times n$ матрици, съставени от 0 и 1 и имащи точно k единици на всеки ред и всеки стълб. Показано е, че с помощта на езика на множествата и операциите над тях може да бъде конструиран значително по-бърз алгоритъм, решаващ тази задача.