Informatics in Education, 2010, Vol. 9, No. 2, 159–170
© 2010 *Institute of Mathematics and Informatics, Vilnius*

The Recurrence Relations in Teaching Students of Informatics

Valentin P. BAKOEV
Department of Mathematics and Informatics, Veliko Tarnovo University
2 Theodosi Tarnovski Str., 5000 Veliko Tarnovo, Bulgaria
e-mail: v.bakoev@uni-vt.bg

Received: December 2009

Abstract. The topic "Recurrence relations" and its place in teaching students of Informatics is discussed in this paper. We represent many arguments about the importance, the necessity and the benefit of studying this subject by Informatics students. They are based on investigation of some fundamental books and textbooks on Discrete Mathematics, Algorithms and Data Structures, Combinatorics, etc. Some methodological aspects of training to solve problems with applying recurrence relations are also given. We hope that the considered topics concern also the school teachers in Mathematics and Informatics and the paper will be useful to them.

Keywords: recurrence relations, solving recurrences, teaching recurrences, studying recurrences, applications of recurrences

1. Introduction

The topic of recurrence relations (RR) and their solving has not commonly taken place in teaching students of Informatics in many universities. The same is valid for its presence in textbooks on Discrete Mathematics and Algorithms and Data Structures. If we look at the Discrete Mathematics textbooks which are more than 25–30 years old – for example (Denev *et al.*, 1984; Kuznetsov and Adelson-Velskii, 1980; Yablonski, 1979) – we shall see that they do not include RR. Almost the same is true for the classic textbooks on Algorithms and Data Structures by Knuth (1969), Aho, Hopcroft and Ullman (1974), Sedgewick (1983, with Pascal; 1998, with C; 2002, with Java programming). Knuth (1969, Chapter 1 – Basic Concepts, Part 1.2. Mathematical preliminaries) considers generating functions only, he does not use the term "recurrence relation" even when he talks about the Fibonacci numbers. In other cited textbooks RR are used in analysis of the time-complexity of algorithms (mostly of the type "divide-and-conquer"), but their solutions are not derived, they are simply given. During the last 15–20 years we can see a trend to the opposite direction – probably because of the increasing importance of Combinatorics, owing to its links with Computer Science, Statistic and Algebra (Cameron, 1994). The newer textbooks on Discrete Mathematics, such as Anderson I. (2001), Anderson J. (2001), Grimaldi (1999), Koshy (2004), Manev (2007), Rosen (1998), etc. consider RR and their solving. The modern textbooks on algorithms and especially for algorithms and complexity include this topic (Cormen *et al.*, 1990; Nakov and

Dobrikov, 2005; Wilf, 1994). Nevertheless there are enough counter examples (Akimov, 2001; Erusalimski, 2001; Garnier and Taylor, 2002). In the last editions of the textbooks by Sedgewick (1998, 2002) the recurrences, used in analysis of time-complexities of algorithms are not solved, their solutions are simply given again. Maybe this is one of the reasons for some professors to underrate the topic of RR and to omit it in their lectures. Other reasons known to us are:

- the politic and the corresponding curriculum in some universities are oriented to practice. Their students obtain mostly practical knowledge and skills (for almost anything that the job market needs), instead of “unnecessary” theoretical knowledge. Such politic is attractive for many students, especially when they choose university or subjects in the first terms of their study;
- the coverage time for Discrete Mathematics (Discrete Structures) course is insufficient, or the professors pay more attention to other, “more important” topics. Often such lecturers do not realize the basic role and the importance of RR for some subsequent courses, the applications of RR in many other subjects, the relations between them, etc.

In this paper we discuss the reasons why it is important and necessary for the students of Informatics to study RR, their solving, applications and usage. In Section 2 we just summarize the basic definitions and methods for solving RR, since this is not the main goal of the paper – there are many excellent textbooks for this purpose and we mention some of them. In Section 3 we argue for the necessity of studying RR from the Informatics students. We point at some facts and arguments in this direction. In Section 4 we represent some specific methodological aspects of the training in this area.

2. Recurrence Relations

Here we just recall some basic notions and facts about RR and their solving. Generally said, a *recurrence relation*, or simply *recurrence*, for the sequence $\{a_n\} = a_0, a_1, \ldots, a_n, \ldots$ is an equation, which relates the nth term a_n to certain of the preceding terms $a_i,\ i < n$, for all integers $n \geqslant n_0$, where the integer $n_0 > 0$. The recurrence relation is called *linear*, if it expresses a_n as a linear function of fixed previous terms, otherwise it is called *nonlinear*. More precisely:

DEFINITION 1. A kth order linear recurrence relation with constant coefficients is an equation of the form

$$a_n + c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k} = f(n), \quad n \geqslant k, \tag{1}$$

where $c_1, \ldots, c_k$ are real constants, $c_k \neq 0$, and $f(n)$ is a function of n. When $f(n) = 0$ for all n, the corresponding recurrence relation is called **homogeneous**, otherwise it is called **nonhomogeneous**. The boundary values of the first k terms of the sequence, usually $a_0, a_1, \ldots, a_{k-1}$ should be specified. They are called **initial conditions** of the recurrence relation and allow to compute a_n, for each $n \geqslant n_0 = \mathit{const}$ – so the recurrence and the initial conditions determine the sequence $\{a_n\}$ in an unique way.

Further, *talking about RR we have in mind linear recurrence relation with constant coefficients only*. The well-known recurrence, given as an example in each textbook is $f_n = f_{n-1} + f_{n-2}$ with initial conditions $f_0 = 0, f_1 = 1$. This homogeneous RR defines the sequence of Fibonacci numbers.

2.1. *Solving Recurrence Relations*

To solve a recurrence relation of the type (1) means to express a_n in a closed form as a function of n and (in case of necessity) the initial conditions. A general method for solving RR is developed, but there are many cases when some simpler methods and techniques can be applied (Anderson J., 2001; Bogart *et al.*, 2006; Cormen *et al.*, 1990; Graham *et al.*, 1998; Grimaldi, 1999; Koshy, 2004), for example:

1) *The substitution method*. This method requires *to guess* the form of solution in accordance with the initial conditions. Then the truth of the guess must be proved by induction, because "mathematical induction is a general way to prove that some statement about the integer n is true for all $n \geqslant n_0$" (Graham *et al.*, 1998, pp. 19; Anderson J., 2001; Bogart *et al.*, 2006; etc.). Quite knowledge, wit and experience are necessary in application of this method.

EXAMPLE 1. Let us solve the RR $a_n = a_{n-1} + n - 1$ with initial condition $a_1 = 0$. This nonhomogeneous recurrence represents the number of comparisons needed to sort an array of n elements by the method bubble sort (or by straight selection); Grimaldi (1999), Koshy (2004). The corresponding algorithm compares each pair of elements and there are $\binom{n}{2}$ possibilities to choose such pair. It is natural to assume that $a_n = \binom{n}{2} = n.(n-1)/2$. We note that this formula corresponds to the trivial cases for $n = 1, 2, 3, 4$. Its truth can be proved easily by induction on n. Another, more convenient way to make the same assumption for a_n is to start from the initial conditions and to substitute as follows: $a_1 = 0$, $a_2 = a_1 + 1 = 1$, $a_3 = a_2 + 2 = 1 + 2$, $a_4 = a_3 + 3 = 1 + 2 + 3$, and so on.

EXAMPLE 2. Let us solve the recurrence $a_n = 2a_{n-1} + 1$ with initial condition $a_1 = 1$. It expresses the number of moves of disks, necessary to solve the Tower of Hanoi puzzle[1] for n disks. Following the recurrence we calculate: $a_2 = 2.1 + 1 = 3$, $a_3 = 2.3 + 1 = 7$, $a_4 = 2.7 + 1 = 15$, etc. So we assume that $a_n = 2^n - 1$. The proof of this assumption by induction on n is trivial.

2) *The iteration method*. This method consists of *iterating (expanding)* the recurrence, step by step, applying it to itself. So the RR converts to summation (sometimes product) of terms depending only on n and the initial conditions. Afterward techniques for evaluating summations, as these demonstrated in Cormen *et al.* (1990), Graham *et al.* (1998),

[1]Proposed by the French mathematician Edouard Lucas in 1883 for 8 disks. It is based on an old legend about the Tower of Brahma with 64 golden disks on three diamond needles.

Knuth (1969) are used to obtain the solution. Often sums of terms of arithmetic or geometric series are obtained and the corresponding formulas should be known. When this method is applied to some recursive algorithms *"a recursion tree* is a convenient way to visualize what happens when a recurrence is iterated, and it can help organize the algebraic bookkeeping necessary to solve the recurrence" (Cormen *et al.*, 1990, pp. 59);

EXAMPLE 3. Let us solve the recurrence $a_n = a_{n-1} + n$ with initial condition $a_0 = 1$. It represents the maximum number of regions defined by n lines in the plane[2]. The recurrence looks appropriate to guess its solution again (as in Example 1), but we shall apply and illustrate the iteration method instead. We expand ("unfold", "unwind") this recurrence as follows:

$$\begin{aligned} a_n &= a_{n-1} + n = (a_{n-2} + n - 1) + n = a_{n-2} + (n-1) + n \\ &= (a_{n-3} + n - 2) + (n-1) + n = a_{n-3} + (n-2) + (n-1) + n \\ &= \ldots = a_0 + 1 + 2 + \ldots + (n-1) + n \\ &= 1 + n(n+1)/2. \end{aligned}$$

3) *The master method* is specially developed for solving recurrences which describe the time-complexity of algorithms, obeying to the *"divide-and-conquer"* strategy. These RR have the form $T(n) = aT(n/b) + f(n)$, where $a \geqslant 1$ and $b > 1$ are integer constants, and $f(n)$ is asymptotically positive function. Because their importance, the solutions of this recurrence and the cases when they exist are formulated in special theorems in Bogart *et al.* (2006), Koshy (2004), Rosen (1998), Rosen *et al.* (2000), "The master theorem" (Cormen *et al.*, 1990).

4) *The general method* gives the solution of a homogeneous RR of the type (1) by the following classical theorem (Anderson J., 2001; Chen, 1992; Koshy, 2004; Manev, 2007; Rosen, 1998; etc.).

Theorem 1. *Let*

$$a_n + c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k} = 0 \tag{2}$$

be a kth order linear homogeneous recurrence relation with constant coefficients and initial conditions $a_0, a_1, \ldots, a_{k-1}$. The associated with it ***characteristic equation*** *is*

$$x^k + c_1 x^{k-1} + c_2 x^{k-2} + \ldots + c_{k-1} x + c_k = 0\,. \tag{3}$$

Let the equation (3) *has generally s distinct roots $\alpha_1, \alpha_2, \ldots, \alpha_s$ (complex in the general case), where α_i has multiplicity m_i, $i = 1, 2, \ldots, s$, and $m_1 + m_2 + \ldots + m_s = k$. Then the solution of the recurrence relation* (2) *is: $a_n = \sum_{i=1}^{s} P_i(n).\alpha_i^n$, where $P_i(n)$ is a polynomial of n, of degree $(m_i - 1)$ and having undetermined coefficients, for $i = 1, 2, \ldots, s$.*

[2]The first who solved this problem is the Swiss mathematician Jacob Steiner in 1826.

The polynomials $P_i(n)$, $i = 1, \dots, s$, have k undetermined coefficients generally, which are determined in an unique way by the initial conditions.

The solutions of the nonhomogeneous RR are determined by the following theorem (Grimaldi, 1999; Koshy, 2004; Manev, 2007; Rosen, 1998).

Theorem 2. *Let*

$$a_n + c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k} = f(n) \tag{4}$$

be a kth order linear nonhomogeneous recurrence relation with constant coefficients, initial conditions $a_0, a_1, \dots, a_{k-1}$, and $f(n)$ is not identically zero. Then the general solution of (4) *is:*

$$a_n = a_n^{(h)} + a_n^{(p)},$$

where $a_n^{(h)}$ is a general solution of the associated homogeneous RR (given by Theorem 1*), and $a_n^{(p)}$ is a particular solution of the nonhomogeneous RR* (4)*.*

The form of the particular solution $a_n^{(p)}$ depends on $f(n)$. A general algorithm for solving an arbitrary recurrence of the type (4) does not exist (Koshy, 2004), but there are two special cases:

i) if $f(n) = b.\alpha^n$, b and α are constants, then

$$a_n^{(p)} = c.n.(n+1).\cdots.(n+m-1).\alpha^n,$$

where m denotes the multiplicity of α as a root of the characteristic equation and c is a constant. If α is not a root of the characteristic equation, then $m = 0$ and therefore $a_n^{(p)} = c.\alpha^n$. The constant c is determined by substitution $a_n^{(p)}$ into the given recurrence;

ii) if $f(n) = Q(n)$, where $Q(n)$ is a polynomial of n of degree d, then $a_n^{(p)}$ has the form $a_n^{(p)} = n^m.P(n)$. Here m denotes the multiplicity of the integer 1 as a root of the characteristic equation ($m = 0$ when 1 is not a root), and $P(n)$ is a polynomial of n of degree d, having undetermined coefficient. They are determined by substitution $a_n^{(p)}$ into the given recurrence.

More general case, which unites (like linear combinations) the considered cases, i.e., $f(n) = b_1^n Q_1(n) + \ldots + b_m^n Q_m(n)$, is considered in Manev (2007). Here b_i are distinct constants and $Q_i(n)$ are polynomials of n of degree $(d_i - 1)$, $i = 1, 2, \dots, m$. If we denote $\alpha_{k+i} = b_i$, $i = 1, \dots, m$, and we consider it as a root of multiplicity d_i of the characteristic equation, then the solution of such recurrence gets the form of homogeneous one, given in Theorem 1. Then, excepting the given initial conditions, more $d_1 + \ldots + d_m$ initial conditions have to be computed by using the given recurrence. Thus the solving of such type nonhomogeneous RR reduces to solving of homogeneous one.

Another cases for the type of $f(n)$ and the form of $a_n^{(p)}$, which have to be look for, are given in Grimaldi (1999), Rosen (1998), Rosen *et al.* (2000).

Finally we note that after the particular solution is obtained, we use the initial conditions to determine the undetermined coefficients in $a_n^{(h)}$, i.e., they are determined in the end.

Another methods for solving recurrences are based on *generating functions*, *difference sequences* (also called sequences of differences); *the annihilator method*, etc. (Cameron, 1994; Chen, 1992; Graham *et al.*, 1998; Koshy, 2004; Merris, 2003; Rosen, 1998; Rosen *et al.*, 2000).

EXAMPLE 4. Let us solve the recurrence $a_n = a_{n-1}+n^2$ with initial condition $a_1 = 1$. It represents the number of all squares in a square grid of dimension n. We rewrite it as $a_n - a_{n-1} = n^2$. Its characteristic equation $x-1 = 0$ has a root $x = 1$ of multiplicity 1. So the solution of the homogeneous recurrence is $a_n^{(h)} = d.1^n$, where $d = const$. As we have noted in the second case, the particular solution has the form $a_n^{(p)} = n^1.(an^2 + bn + c)$. By substitution of $a^{(p)}$ in the recurrence we obtain:

$$\begin{aligned} an^3 + bn^2 + cn - a(n-1)^3 - b(n-1)^2 - c(n-1) &= n^2, \\ an^3 + bn^2 + cn - an^3 + 3an^2 - 3an + a - bn^2 + 2bn - b - cn + c &= n^2, \\ 3an^2 - 3an + a + 2bn - b + c &= n^2. \end{aligned}$$

We equalize the coefficients of equal degrees of n in the both sides of last equality and so we obtain:

$$\begin{aligned} 3a = 1 &\Rightarrow a = 1/3, \\ -3a + 2b = 0 &\Rightarrow -1 + 2b = 0 \Rightarrow b = 1/2, \\ a - b + c = 0 &\Rightarrow c = b - a = 1/6. \end{aligned}$$

Hence $a_n^{(p)} = n^3/3 + n^2/2 + n/6 = (2n^3 + 3n^2 + n)/6 = n(n+1)(2n+1)/6$ and so the solution of the nonhomogeneous recurrence is $a_n = d + n(n+1)(2n+1)/6$. We use the initial condition: $1 = a_1 = d + 1 \Rightarrow d = 0$ and therefore $a_n = n(n+1)(2n+1)/6$.

3. Why the Students of Informatics Need to Study Recurrence Relations

We can give many arguments about the necessity, importance and benefit of studying the recurrence relations and their solving by the Informatics students. The most important among them are:

1) *The recurrences are very powerful tool* (sometimes an unique one) for solving many counting problems, where it is difficult (or impossible) to count the objects by using the known combinatorial techniques. So the RR and their solving are important and useful complement to the knowledge in Combinatorics and thence in Probability theory and Statistics. Not by chance the topic of recurrences and their solving takes an important place in the known books in Combinatorics (for example, Cameron (1994),

Comtet (1974), Graham *et al.* (1998), Hall (1967), Merris (2003), Reingold *et al.* (1977), Vilenkin *et al.* (2006), etc.).

2) *The recurrences are connected directly with the recursion*, as their names prompts. Less or more, the RR are used in teaching recursion to students and they are considered together in some textbooks (Bogart *et al.*, 2006; Koshy, 2004; Rosen, 1998; Wilf, 1994). The recursive computing of $n!$, the nth Fibonacci number, etc. are classic examples in creating recursive functions in each textbook on programming. These functions are built on the corresponding recurrences (i.e., recursive definitions) in a most natural way. So they can be used in explanation the execution of recursion and also the *inherent key steps*, which it performs: forward steps, reaching the basic case (bottom) of the recursion and backward steps (optional in some recursions). For example, the recursive calls are determined by the corresponding recurrence and the forward steps correspond to expanding the terms of the recurrence as in the iteration method (but it happens automatically, by pushing stack frames in the program stack). The initial conditions serve as a basic case of the recursion. The backward steps correspond to computing in an inductive manner – starting from the initial conditions, if all terms up to the $(n-1)$st are computed, then the nth term will be computed (the topic "recursion and iteration" is important and extensive, so it needs more attention). The example with the Fibonacci numbers is a classic one for *ineffective recursion*. The most convenient way to illustrate and to explain why this recursion is ineffective is by using the corresponding recursion tree (Cormen *et al.*, 1990). By it for the efficiency of similar function we can conclude: "Let a recurrence of type (1) of order $k > 1$ and its initial conditions be given. A recursive function, which computes the nth term of the corresponding sequence by recursive calls of itself for each term in the recurrence is ineffective". Here is the place to mention two known techniques for avoiding the ineffective recursion: *memoization* (when some value is computed, it is stored in an array to be used every time when the recursion tries to compute it again) and *replacement of recursion by iteration and using a stack in case of necessity* (i.e., applying the bottom-up approach).

3) *The recurrences are used in the analysis of complexity of algorithms*, mostly recursive. As we mentioned above, the recurrences and the recursion trees are appropriate, powerful and unique tools for investigation the time-complexity of algorithms, based on a strategy "divide-and-conquer".

4) *The recurrences are in the foundations of the "dynamic-programming" strategy* and we underline the importance of this fact. The second step of the paradigm of this strategy is "Recursively define the value of an optimal solution" (Cormen *et al.*, 1990), which means to derive the corresponding recurrence and to prove its correctness. The third step is "Computing the value of an optimal solution in a bottom-up fashion", since the key ingredient, which the optimization problem should have for dynamic programming to be applicable is *overlapping subproblems*. This means ineffective recursion, which should be avoided.

5) *The ability to solve recurrences implies more effective solutions* of many problems, related to RR. Often the solutions are formulas in a closed form and so they have computational complexities of constant type – in a contrast to recursive or iterative computations

following formula (1). Often this fact is used in problems, given in competitions (tournaments, Olympiads) in programming. It also helps to improve the efficiency of some algorithms, based on dynamic programming.

6) *Knowing the linear RR with constant coefficients is a starting point for studying other recurrences*. For example, such recurrences are the recurrence tables (except the known Pascal's and Stirling's triangle, such tables appear in dynamic programming), RR with coefficients, which are not constant (as in the Catalan's numbers), the numbers of Stirling, Bell, Euler, Bernoulli, etc. Such recurrences and techniques for solving them are considered in Bakoev (2004), Cameron (1994), Graham *et al.* (1998), Rosen (1998), Rosen *et al.* (2000).

7) It is important to point some facts from *Computer Science Curriculum 2008: An Interim Revision of CS 2001*, a joint task of the ACM and IEEE Computer Society (ACM and IEEE Computer Society, 2008). It is not necessary to comment, we just cite the program in:

- *Discrete Structures (DS), Basic of counting* contains parts: "Arithmetic and geometric progressions", "Fibonacci numbers", "Solving recurrence relations" and "The Master theorem". Three of all four learning objectives are: "State the definition of the Master theorem", "Solve a variety of basic recurrence equations" and "Analyze a problem to create relevant recurrence equations or to identify important counting questions";
- *Programming Fundamentals (PF), Recursion* includes the topics: "The concept of recursion", "Recursive mathematical functions", "Simple recursive functions", "Divide-and-conquer strategies" and "Recursive backtracking". Among the learning objectives are: "Identify the base case and the general case of a recursively defined problem", "Compare iterative and recursive solutions for elementary problems such as factorial" and "Describe the divide-and-conquer approach";
- *Algorithms and Complexity (AL), Basic Analysis* contains the topic "Using recurrence relations to analyze recursive algorithms", and some of the learning objectives are: "Deduce recurrence relations that describe the time complexity of recursively defined algorithms" and "Solve elementary recurrence relations", and so on.

So we conclude that the RR and their solving are significant not only to themselves, they are also an important link between many subjects in teaching the students of Informatics – for example Combinatorics, Probability theory and Statistics, Programming, Algorithms and Data Structures, Complexity of Algorithms, Training for Competitions in Programming, Numerical Methods (because of the relation with the difference sequences (Merris, 2003; Rosen, 1998)), and even Differential Equations (because of the analogy in solving RR and linear differential equations (Cameron, 1994; Rosen *et al.*, 2000; Wilf, 1994)). The theory of RR and their solving is not an unmeaning mathematical one, without real applications. Contrariwise, this theory helps, explains and stays in the base of many subjects in the area of Informatics. The core of this theory is not large (as we have seen above) and it is not too hard for the students. So we consider that *studying the recurrences by the students of Informatics is justified and necessary*. Hence we can call our attention to the question "*How to teach recurrences so that the students realize their importance and become motivated to study them*?".

4. How to Teach Recurrences to the Informatics Students

Here we do not consider the common methodical treatments in teaching Mathematics and Informatics, we focus our attention on the particular and specific formulations in teaching recurrences to students of Informatics. We note, that the arguments represented in the previous section give some of the possible answers to the question. We shall add to them the following:

1) *The topic of recurrences should be taught in relation to other subjects*, pointing to the links between them and to the applications in them. On the other hand, the professors in subjects mentioned above can demonstrate why and how they use recurrences in the context of the relationship between the subjects;

2) *Only suitable textbooks should be used*, where the topic of recurrences is well-represented – consequently, systematically, with many and well-chosen examples and problems – as in the textbooks in Discrete Mathematics (Anderson I., 2001; Grimaldi, 1999; Koshy, 2004; Rosen, 1998), the textbooks in Algorithms and Data Structures (Cormen *et al.*, 1990; Nakov and Dobrikov, 2005), the textbooks in Combinatorics (Cameron, 1994; Graham *et al.*, 1998; Merris, 2003; Vilenkin *et al.*, 2006), etc. There are examples in the opposite direction – the textbooks, which just touch the topic, where only problems of the type "Solve the recurrence ..." are given. They develop technique and skills for solving hard recurrences only (of higher order, with complex roots, etc.), they do not contain even one problem to draw up a recurrence. So the students can not realize the meaning, applications and benefit of the recurrences;

3) *The problems for development technical skills for solving recurrences should be followed by problems for drawing up a recurrence and thereafter its solving*. It is important for the students to know, that "problems that require an answer depending on the integer n, where the solution to the problem for a given size n can be related to one or more cases of the problem for smaller sizes" should be solved by recurrences (Rosen *et al.*, 2000);

4) *The problems, which need to draw up a recurrence and thereafter to solve it* can be solved by the following *inductive scheme:*

- according to the integer parameter n we denote by a_n the number of objects which we have to count. For these values of n, which give meaning to the problem we consider the corresponding sequence, looking for a recurrence for a_n;
- we compute the initial conditions, i.e., the first terms of the sequence, for which the problem has a meaning;
- inductive suggestion – we assume that all terms of the sequence to a_{n-1} inclusive, are known to us;
- we express the relation between a_n and the previous terms of the sequence and prove the derived recurrence. Often we reason as follows: "We have (we have

placed, we have drawn, etc.) $n - 1$ elements. They form a_{n-1} objects of the type which we look for and, in accordance with the inductive suggestion, we know a_{n-1}. We add (we place, we draw, etc.) the nth element. How the number of objects which we look for is changed (increases)?". Once more we note that our *arguments in deriving the recurrence should have a form of proof*. We check whether the recurrence satisfies the initial conditions;
- we solve the obtained recurrence and check the solution.

The recurrences in the considered examples are obtained in this way. We recommend the students to master these steps and their application in solving such problems. We explicitly note that *the formation of such manner of thinking and mastering the technique for solving similar problems help the students in a crucial degree in creating algorithms, based on the dynamic-programing strategy*. The known to us textbooks do not pay the necessary attention to this important fact;

5) It is relevant to use in teaching *a classification of the problems, which need to draw up a recurrence and thereafter to solve it*. Many classifications are possible and each of them should help the recognition of the problems in order to solve them easily. We can consider as a successful each classification which helps the students in solving such problems and in mastering the topic of recurrences, generally. The classification of the problems, which need to draw up a recurrence and thereafter to solve it is a complex topic, it has many aspects and can be a subject of another study. To get an idea of this we just mention some of the many criteria, which can be used in classification:

- *the order of the obtained recurrence*;
- *the type of the obtained recurrence* – linear or nonlinear, homogeneous or nonhomogeneous, with constant coefficients or not, etc.;
- *the area, which the problem concerns* – for example, finances (problems for computing compound interest on deposits and credits), biology (problems for computing growth of populations), geometry (problems for counting figures), number theory (counting partitions or compositions of integers), etc.;
- *what should be determined* – a certain (nth) term of a sequence, or a sum of some (or all) terms of a given sequence, or the number of all sequences of a certain type;
- *the text of the problem contains characteristic keywords*, which require a specific manner of thinking, or prompt for certain recurrences or fundamental problems – for example, *to have no two consecutive equal elements*; or *to contain odd/even number of certain elements*, etc.;
- *problems for counting in recursively/inductively defined figures* – in fractal figures, or self-similar figures, etc., as in Koshy (2004), Rosen (1998);
- *counting problems, related to polygonal (figurate) numbers*, as in Bakoev (2004), Koshy (2004), Rosen (1998), etc.

6) *The meaning, which some sequences have is very important aspect in studying recurrences*. The same sequence can have many interpretations. This fact and the topic of recurrences help in a higher degree to introduce to the students *The On-Line Encyclopedia of Integer Sequences* (OEIS), Sloane (2009). This site is maintained by N.J.A. Sloane and

offers many interesting possibilities. The most important among them are: searching (by sequence, i.e., by its few consecutive terms, by word, by author, by sequence number, by keywords, etc.) and contribution to OEIS – to add comments, notes, formulas, program code, etc. to existing sequences, or to add new sequences. At the beginning of August 2010 the data-base of OEIS contains information about more than 178000 sequences. For example, the sequence in Example 4: 1, 5, 14, 30, 55, 91, 140, 204, 285, . . . has a number A000330 in OEIS and it is represented as "Square pyramidal numbers". Except this, about 20 comments, meanings, interpretations, etc. of it and its terms are given additionally. Many references, links, citations, formulas, program code, etc. are also given.

5. Conclusions

Here we discussed some aspects of teaching the topic "Recurrence relations" and its studying by students of Informatics. We also shared the most important of our opinions, conceptions and experience in teaching recurrences to the students. Of course, many other aspects, arguments and points of view can be added, a full pedagogical investigation can also be done. We hope, that the given arguments and facts are enough to aid the professors in teaching, as well as to motivate the students – about necessity, importance, applications and the ways of usage – in studying the recurrences.

We also hope that the considered topics can be useful to the teachers in the high schools, where Mathematics and Informatics are studied intensively. For example, when the arithmetic and geometric series are studied, it is possible to represent some appropriate first-order recurrences and to explain the relations between them. As an application of solving square equations it is worth to represent second-order recurrences and to solve some of them. We consider, that recurrences of first and second order can be used and applied successfully in training school-teams for competitions in Mathematics or Informatics.

References

ACM and IEEE Computer Society (2008). *Computer Science Curriculum 2008: An Interim Revision of CS 2001*. Accessible at `http://www.acm.org/education/curricula-recommendations`.

Aho, A.V., Hopcroft, J.E., Ullman, J.D. (1974). *The Design and Analysis of Computer Algorithms*. Addison-Wesley.

Akimov, O.E. (2001). *Discrete Mathematics. Logic, Groups, Graphs*. Laboratoriya bazovih znanii, Moskow (in Russian).

Anderson, I. (2001). *A First Course in Discrete Mathematics*. Springer-Verlag, London.

Anderson, J. (2001). *Discrete Mathematics with Combinatorics*. Prentice-Hall, New Jersey.

Bakoev, V. (2004). Algorithmic approach to counting of certain types m-ary partitions. *Discrete Mathematics*, 275, 17–41.

Bogart, K., Stein, Cl., Drysdale, R. (2006). *Discrete Mathematics for Computer Science*. Springer.

Cameron, P.J. (1994). *Combinatorics: Topics, Techniques, Algorithms*. Cambridge Univ. Press.

Chen, W.W.L. (1992). *Discrete Mathematics*. With Macquarie University, Free online book.

Comtet, L. (1974). *Advanced Combinatorics. The Art of Finite and Infinite Expansions*. Revised and enlarged D. Reidel (Ed.) Publ. Co., Dordrecht-Holland/Boston-USA.

Cormen, T., Leiserson, Ch., Rivest, R. (1990). *Introduction to Algorithms*. The MIT Press, New York.
Denev, J., Pavlov, R., Demetrovich, Y. (1984). *Discrete Mathematics*. Nauka i izkustvo, Sofia (in Bulgarian).
Erusalimskii, Y.M. (2001). *Discrete Mathematics: Theory, Problems, Applications*. Vuzovskaia kniga, Moskow (in Russian).
Garnier, R., Taylor, J. (2002). *Discrete Mathematics for New Technology*. Second edition, IOP Publishing Ltd.
Graham, R., Knuth, D., Patashnik, O. (1998). *Concrete Mathematics. A Foundation for Computer Science*. Second edition, Addison-Wesley.
Grimaldi, R. (1999). *Discrete and Combinatorial Mathematics. An Applied Introduction*. Fourth edition, Addison-Wesley.
Hall, M., JR. (1967). *Combinatorial Theory*. Blaisdell Publ. Co., Waltham (Massachusetts)–Toronto–London.
Knuth, D. (1969). *The Art of Computer Programming*. Volume 1: Fundamental Algorithms. Addison-Wesley, Reading, Massachusetts.
Koshy, T. (2004). *Discrete Mathematics with Applications*. Academic Press.
Kuznetsov, O., Adelson-Velskii, G. (1980). *Discrete Mathematics for Engineers*. Energiya, Moskow (in Russian).
Manev, K.N. (2007). *Introduction to Discrete Mathematics*. Fourth edition, KLMN, Sofia (in Bulgarian).
Merris, R. (2003). *Combinatorics*. Second edition, A John Willey&Sons, Hoboken, New Jersey.
Nakov, P., Dobrikov, P. (2005). *Programming = ++ Algorithms*. Third (revised) edition, TopTeam Co, Sofia.
Reingold, E., Nievergelt, J., Deo, N. (1977). *Combinatorial Algorithms, Theory and Practice*. Prentice-Hall, New Jersey.
Rosen, K.H. (1998). *Discrete Mathematics and its Applications*. Fourth edition, McGraw-Hill.
Rosen, K., Michaels, J., Gross, J., Grossman, J., Shier, D. (2000). *Handbook of Discrete and Combinatorial Mathematics*, CRC Press.
Sedgewick, R. (1983). *Algorithms*. Addison-Wesley.
Sedgewick, R. (1998). *Algorithms in C*. Third edition, Addison-Wesley Longman.
Sedgewick, R. (2002). *Algorithms in Java*. Third edition, Addison-Wesley.
Sloane, N.J.A. (2009). *The On-Line Encyclopedia of Integer Sequences (OEIS)*. Published electronically at `http://www.research.att.com/~njas/sequences/`.
Vilenkin, N., Vilenkin, A., Vilenkin, P. (2006). *Combinatorics*. FIMA, Moskow (in Russian).
Wilf, H. (1994). *Algorithms and Complexity*. Univ. of Pennsylvania Philadelphia, Pennsylvania, Internet edition.
Yablonski, S. (1979). *Introduction to Discrete Mathematics*. Nauka, Moskow (in Russian).

V.P. Bakoev is associate professor in the Department of Mathematics and Informatics at St. Cyril and St. Methodius University of Veliko Tarnovo, Bulgaria. He received his PhD in computer science in 2004.

Rekurentinis sąryšis mokant informacinių technologijų

Valentin P. BAKOEV

Šiame straipsnyje nagrinėjama rekurentinio sąryšio tema ir jos vieta mokant informacinių technologijų. Straipsnyje pateikiama daugybė argumentų, kodėl svarbu, būtina ir naudinga nagrinėti šią temą. Autoriai remiasi diskrečiosios matematikos, algoritmų ir duomenų struktūros, kombinatorikos ir kitų knygų bei vadovėlių medžiaga. Taip pat aptariami keli metodologiniai iškylančių sunkumų sprendimų ir mokymo būdai taikant rekurentinį sąryšį. Šis straipsnis turėtų būti naudingas ir matematikos, ir informatikos mokytojams.