# Modeling of a Real Situation as a Method of the Algorithmic Thinking Development and Recursively Given Sequences

ŠTĚPÁN HUBÁLOVSKÝ
Department of Informatics
Faculty of Education
University of Hradec Králové

EVA MILKOVÁ, PAVEL PRAŽÁK
Department of Informatics and Quantitative Methods
Faculty of Informatics and Management
University of Hradec Králové
Rokitanského 62
CZECH REPUBLIC
stepan.hubalovsky@uhk.cz, eva.milkova@uhk.cz, pavel.prazak@uhk.cz

*Abstract:* A method of developing students' algorithmic thinking is demonstrated in the paper first. This is followed by an example of a physical problem step by step modeling and several approaches to its solution. These approaches demonstrate possible interdisciplinary learning, which is considered to be a very important part of future teachers' education. Especially the relationship among physics, computer science and mathematics is introduced. Finally the benefit of CAS for explaining and visualizing recursively defined sequences is discussed.

*Key-Words:* Algorithms, interdisciplinary learning, modeling, visualization, motivation, CAS, Maple, recursively defined sequence, linear recursive equation, logistic equation, cobweb plot, bifurcation diagram.

## 1 Introduction

The ability to create algorithms develops logical thinking skills and imagination and is an inseparable part of a student's study skills for those studying the specializations "Applied Informatics" and "Information Management" at the Faculty of Informatics and Management and "Informatics" at the Faculty of Education. The students, future teachers, not only get as deep knowledge as possible in this area, but also are made familiar with various ways of teaching the creation of basic algorithms, which they can apply at primary and secondary schools.

In this paper we first briefly introduce one method for developing algorithmic thinking in our students. We describe the subject *Algorithms and Data Structures*, which is taught at both above mentioned faculties in the first term and is placed before the other subjects dealing with algorithmic and programming skills.

Secondly we introduce a case study illustrating step by step modeling of a real situation as a suitable example of algorithmic thinking development and also as an example of interdisciplinary learning that is useful to include in secondary school teacher education. (cf. [9])

Finally we discuss the benefit of CAS for explaining and visualizing recursively defined sequences. Using several animations prepared in the MAPLE environment we illustrate our approach to enhance the students' knowledge in the mentioned area. (cf. [3])

## 2 Creation of Basic Algorithms

Education at secondary schools and colleges in the area of informatics is directed mainly to a user attitude. Only students attending optional subjects dealing with programming languages are familiar with creating algorithms. Thus a lot of students studying at our university are without any algorithmic knowledge at the beginning of their studies.

In university departments training students in computer-related disciplines the creation of algorithms is still taught mostly within subjects dealing with programming languages. At our university we have been providing another approach to developing algorithmic thinking for more than 10 years and we are pleased to say that this approach is really suitable and more and more faculties have begun using it as well.

We devote the whole first term for increasing algorithmic thinking of our students in the subject called *Algorithms and Data Structures*. Education in this subject lacks all the structures and constructions connected with structured paradigms. Consequently students do not get any habits which might make the entrance into object oriented world more complicated.

**Our approach to the creation of algorithms is based on the imagination of a brick-box, a nice and useful game for children. There are only several base elements available from which children are able to create incredible buildings.**

Thus when we lead our students' first steps by creation of algorithms we explain to them that it is like building interesting objects out of just a few basic elements. In the subject *Algorithms and Data Structures* it means that we start our teaching with basic algorithmic structures (basic elements from the brick-box) and typical algorithmic structures (a few parts made out of these elements) written in Czech meta-language and then we let students get into the secrets of making whole algorithms (building whole constructions) written in Czech meta-language as well.

By basic algorithmic structures we consider only the basic structures: block of commands, conditional constructions (*incomplete* and *complete branching*) and loop construction (*while construction* and its shorter description for the case with known number of loops, i.e. *for construction*). By typical algorithmic structures we mean common structures typical for the whole group of problems, as e.g. determination of the number of elements with the given property, ***determination of the sum*** (see thereinafter) and the product of the elements, finding the first and last element with the given property, determination of the maximum and minimum of all elements or of elements with the given property.

At first we explain and practice all structures on algorithms, which deal with values sequentially saved into a simple variable. We very carefully distinguish two cases: if the number of saved values ***is*** or ***is not*** previously known. To be better understandable we describe this difference on a simple practical situation as e.g.:

*Previously known number of values*

Let us consider the following situation. A charitable auction is held, and people are transferring money into a certain bank account. We sum up the amounts sent and tell the organizer the sum ***obtained from the first fifty people***.

*Previously unknown number of values*

Now let us consider a similar situation but with the following modification. The charitable action is held, the people are sending money to a certain bank account. We sum up the amounts sent but we should tell the organizer the sum ***obtained during the first day***.

After a thorough exercise of basic algorithms on problems using single variables we proceed and explain the data structure one-dimensional array and later two-dimensional array as well.

Next, the students welcome the possibility to practice their knowledge also on tasks similar to the following ones:

***Complete the algorithm*** solving the following task. "In the sequence of *n* integers saved in the array a (in items a[1] , ..., a[n]) determine the first minimum value and then sum all integers behind the found minimum value."

```
begin
   minimum := a[1];
   sum := .....;
   for i := 2 to n do
      begin
       sum := sum + .....;
       if a[i] ≤ min then
         begin
           minimum := .....;
           sum := .....;
         end;
      end;
end.
```

"There are *n* integers saved in the array a (see the table). ***Determine the values*** in the array a after finishing the following algorithm. Write them to the table bellow." (see Table 1)

```
begin
   n:=6;
   x:=a[1];
   i := 2;
   while i ≤ n - 1 do
      begin
         if a[i] > x then
            begin
               a[1]:= a[i];
               a[i]:= x;
            end;
         i := i + 1;
      end;
end.
```

| a[1] | a[2] | a[3] | a[4] | a[5] | a[6] |
|---|---|---|---|---|---|
| 11 | 8 | 19 | 7 | 16 | 17 |
| | | | | | |

Table 1 Table of integers saved in the array a

During lessons students apply their knowledge to several tasks. Each task is solved by three groups including about three students. After some time when students prepare their solution on papers, each task is illustrated by three students (one student represents one group) at the blackboard and these three solutions are compared and discussed by all students. In this way students are led to try to find more solutions to the given task and to be able to understand the efficiency of algorithms as well. **This method of practice serves also**

**as useful inspiration for our students - future teachers.**

The whole above mentioned approach to the development of algorithmic thinking is described in the textbook [6] More than 150 problem assignments, questions and exercises are presented there. The accuracy of a solution can be verified with the help of the program *Algorithms* which is enclosed on, to the textbook attached, CD. The program is developed in Borland Delphi environment within a thesis [13]. It is user friendly and provides entire graphical comfort for users (see thereinafter in the section 3.4, figures 6 and 7). Using the program *Algorithms* students can place a solution (either the solution prepared by the authors of the textbook or their solution) in the program and the program shows them step by step how their algorithm works. The program shows actual values of variables in each step of the algorithm's process. The program is not only a substantial help to students in their self-study but it helps also teachers to prepare text materials (see more in [7], [8]).

Another useful web application *algds* enabling practice of this knowledge is available online at
http://lide.uhk.cz/fim/ucitel/milkoev1/
in the section ALGDS/lecture. The application is created in PHP language. It enables three types of questions:
*Complete the given algorithm*
*Order the given algorithm*
*Solve the given algorithm*

The test *Complete the given algorithm* demands to complete omitted parts in the algorithm solving the given task (cf. first task on the previous page). The test *Order the given algorithm* demands to order all commands written in wrong order to get algorithm, which correctly solves the given task. The test *Solve the given algorithm* demands to determine values of some variables used in the algorithm using the given enter data (cf. second task on the previous page).

## 3 Modeling of a Real Situation

As we have already mentioned at the beginning of the paper, an important part of future teachers education is teaching them with an emphasis on interdisciplinary learning.

In this section let us describe one real situation using interdisciplinary approaches to its solution.

### 3.1 Definition of the problem

Is it possible to build a leaning tower using identical homogenous blocks of the given length $L$ so that the most top block will be completely placed out of a table? If yes, determine the smallest number of the blocks enabling this (cf. [1]). The situation is shown in Fig 1.

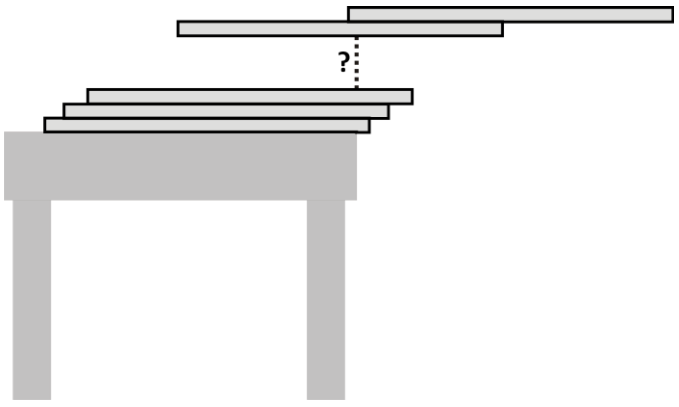


Fig.1 Illustration of the given problem

### 3.2 Physical point of view - analysis of the problem

Since the blocks are homogeneous, each of them has the center of gravity in its center. Block **1** (see Fig.2) can be therefore moved so that its center of gravity is located exactly above the edge of the table, thus with the maximum overlap

$$x_1 = \frac{L}{2}.$$

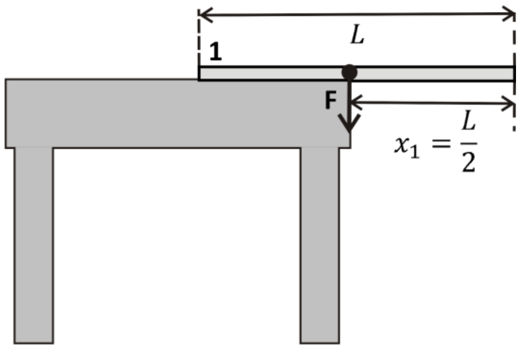


Fig.2 The center of gravity of the block **1**

Similar assumption can be taken for the system of blocks **1** and **2** (see Fig.3). The center of gravity of the pair of blocks **1** and **2** is located above the edge of the table.

Applying the moment theorem

$$F(\frac{L}{2} - x_2) = Fx_2$$

we get the overlap $x_2 = \frac{L}{4}$. Hence, the pair of blocks **1** and **2** can be moved over the table by 1/4 of the length $L$, which means that now the overlap of the block **1** is $\left(\frac{1}{2}+\frac{1}{4}\right)L=\frac{3}{4}L$.

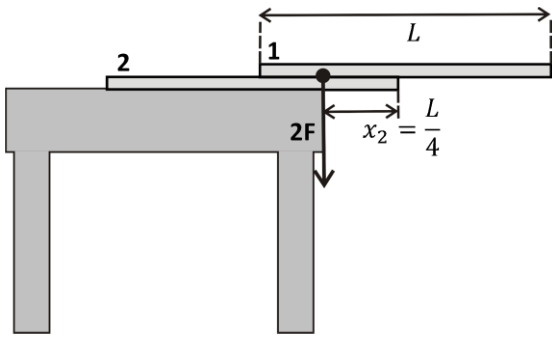


Fig.3 The center of gravity of the pair of blocks **1** and **2**

The next step is analogous – it's necessary to determine the overlap of the system of blocks **1**, **2** and **3** over the table. Again, the position of their common centre of gravity can be determined applying the moment theorem

$$F\left(\frac{L}{2}-x_3\right)=2Fx_3,$$

so we get the overlap $x_3=\frac{L}{6}$ (see Fig.4). All three blocks together can be therefore further moved by 1/6 of the length $L$, which means $\left(\frac{1}{2}+\frac{1}{4}+\frac{1}{6}\right)L=\frac{11}{12}L$ the overlap of the block **1**.

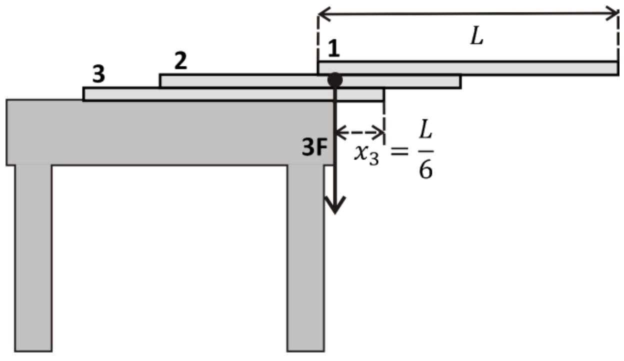


Fig.4 The center of gravity of blocks **1**, **2** and **3**

Obviously, applying the same approach, the system of blocks **1**, **2**, **3** and **4** can be further moved by 1/8 of the length $L$. Hence the overlap of the block **1** is $\left(\frac{1}{2}+\frac{1}{4}+\frac{1}{6}+\frac{1}{8}\right)L=\frac{25}{24}L$, which is found solution.

The described physical problem can be easily demonstrated by a simple experiment using playing cards, CDs, etc. (see Fig. 5).

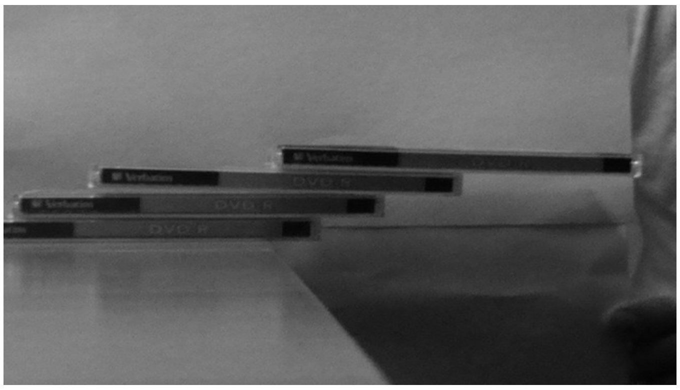


Fig.5 Experiment to the given physical problem

## 3.3 Mathematical point of view – partial sums

The sum $\frac{1}{2}+\frac{1}{4}+\frac{1}{6}+\frac{1}{8}$ is the partial sum $s_4$ of the numerical series $\sum_{k=1}^{N}\frac{1}{2k}$.

Thus we can transform the given physical problem into a mathematical task:

Let us find the smallest partial sum $s_N$ of the numerical series $\frac{1}{2}+\frac{1}{4}+\frac{1}{6}+\ldots$ greater than 1, i.e. let us find the smallest $N$ fulfilling the formula

$$s_N=\sum_{k=1}^{N}\frac{1}{2k}=\frac{1}{2}\sum_{k=1}^{N}\frac{1}{k}\geq 1.$$

Since the harmonic series $\sum_{k=1}^{+\infty}\frac{1}{k}$ is divergent, the series $\frac{1}{2}\sum_{k=1}^{+\infty}\frac{1}{k}$ is also divergent. The problem can be therefore extended in the general task: Given a positive real number $S_{MAX}$. Let us find the smallest $N$ fulfilling the formula

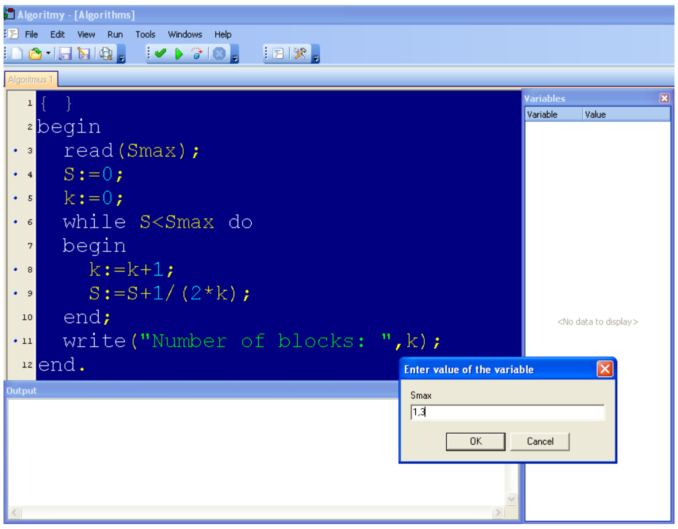
$$s_N=\frac{1}{2}\sum_{k=1}^{N}\frac{1}{k}\geq S_{MAX}. \quad (1)$$

## 3.4 Algorithmic description of the problem

Thanks to the step by step solution illustrated in the section 3.2., the above mathematical generalization (1) can be described by a simple algorithm using the typical algorithmic structure ***determination of the sum*** (see the section 2).

```
begin
   read(Smax);
   S:=0;
   k:=0;
   while S<Smax do
   begin
      k:=k+1;
      S:=S+1/(2*k);
   end;
   write(“Number of blocks: “,k);
end.
```

This algorithmic description can be placed into the program *Algorithms* (see the section 2 and the figures 6 and 7 below) and students can follow actual value of the variable `S`.

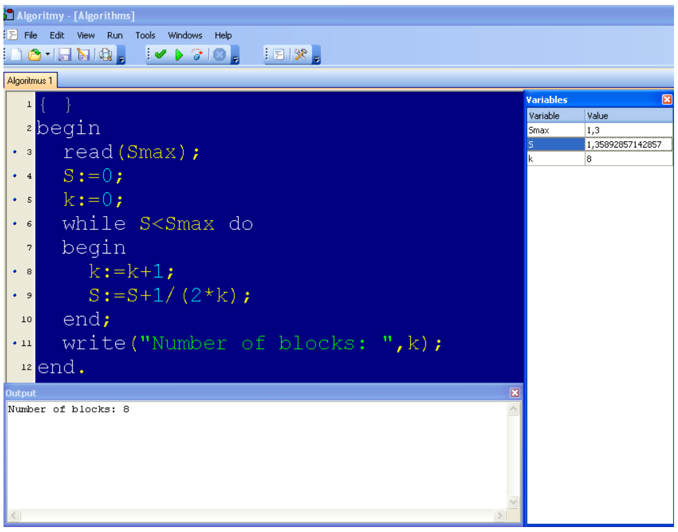


```
{ }
begin
  read(Smax);
  S:=0;
  k:=0;
  while S<Smax do
  begin
    k:=k+1;
    S:=S+1/(2*k);
  end;
  write("Number of blocks: ",k);
end.
```

Enter value of the variable

Smax: 1,3

Fig.6 Above described algorithm solving the task (1) placed in the program *Algorithms*



```
{ }
begin
  read(Smax);
  S:=0;
  k:=0;
  while S<Smax do
  begin
    k:=k+1;
    S:=S+1/(2*k);
  end;
  write("Number of blocks: ",k);
end.
```

| Variable | Value |
|---|---|
| Smax | 1,3 |
| S | 1,35892857142857 |
| k | 8 |

```
Number of blocks: 8
```

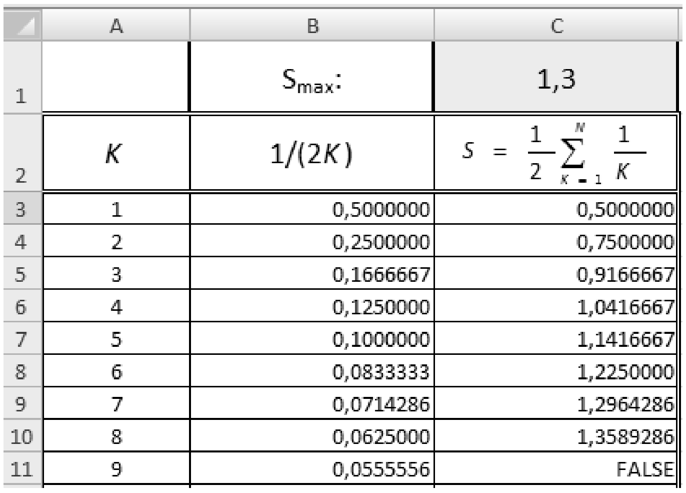
Fig.7 The result of the task (1) obtained in the program *Algorithms* for the given value `Smax` = 1,3

However, students can also follow the consequently obtained values using e.g. MS Excel and practice the work with this environment.

### 3.5 Solution of the problem using the MS Excel

The task (1) can be also easy demonstrated using formulas entered in the MS Excel spreadsheet cells.

In column *A* the quantity of blocks (numbers *k*) are placed there and in column *B* the appropriate values $\frac{1}{2k}$ are counted. In column *C* the formula *IF*(*C*4<*$C$*1; *C*3+*B*4;*"FALSE"*), that is copied using relative addressing to the other cells of column *C*, is used to count the appropriate partial sums (see Fig.8) and to show the result according to the given positive real number $S_{MAX}$.



|  | A | B | C |
|---|---|---|---|
| 1 |  | $S_{max}$: | 1,3 |
| 2 | $K$ | $1/(2K)$ | $S = \frac{1}{2}\sum_{K=1}^{N}\frac{1}{K}$ |
| 3 | 1 | 0,5000000 | 0,5000000 |
| 4 | 2 | 0,2500000 | 0,7500000 |
| 5 | 3 | 0,1666667 | 0,9166667 |
| 6 | 4 | 0,1250000 | 1,0416667 |
| 7 | 5 | 0,1000000 | 1,1416667 |
| 8 | 6 | 0,0833333 | 1,2250000 |
| 9 | 7 | 0,0714286 | 1,2964286 |
| 10 | 8 | 0,0625000 | 1,3589286 |
| 11 | 9 | 0,0555556 | FALSE |

Fig.8 Illustration of the task (1) in MS Excel environment (cf. the result gained on the Fig. 7) for the value $S_{MAX} = 1{,}3$

### 3.6 Physical point of view – a more general approach

In this paragraph we briefly describe a more general approach how to derive the total overlap distance of blocks of the leaning tower over the table's edge. This approach can be characterized as follows: we construct a recurrence relation for a sequence that expresses the maximum overlap distance from the edge of the table. This way allows us to build another relationship among physics, mathematics and computer science, particularly among the moment theorem, the concept of finite or infinite sequence and the general concept of recurrence.

Similarly as before, let *L* denote the length of one block. Further let $d_n$ be the maximum overlap distance of a tower of *n* identical homogenous blocks over the edge of the table, see Fig. 9. It is clear that $n \in N$ and that $d_1=L/2$.

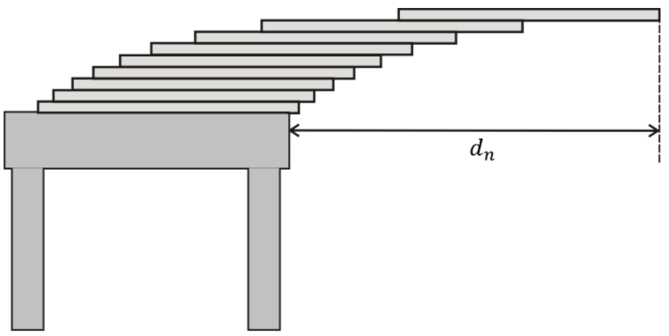


Fig. 9 The length of the overlap distance of a tower of $n$ identical blocks.

The stability of a body depends on its centre of gravity. If the maximum overlap of the tower constructed from $n$+1 blocks is reached, the centre of gravity of the $n$+1 blocks lies right above the edge of the table and the centre of gravity of the $n$ top blocks lies right above the edge of the bottom block. This is the condition of the stability of the tower. Let $F$ be the gravitational force that act on a single block. Then the total moment of the gravitational force of the $n$+1 blocks with respect to the right edge of the tower can be written in two ways as follows:

Firstly, the total moment of the gravitational force of $n$+1 identical blocks can be expressed directly as

$$(n+1)F.d_{n+1},$$

where $d_{n+1}$ is the distance of the centre of gravity from the right edge of the tower, see Fig. 10.

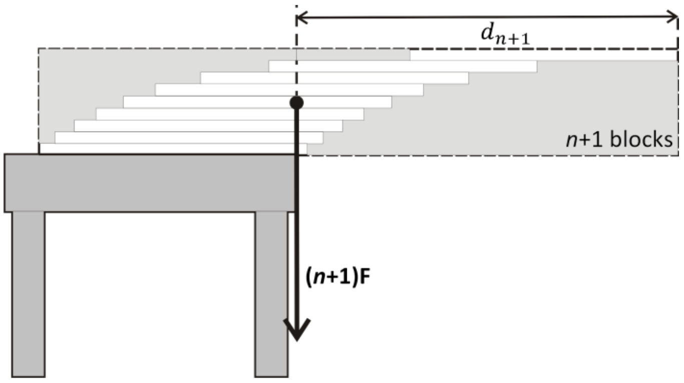


Fig.10 Total moment of the gravitational force of the tower from $n$+1 blocks with respect to the right edge of the tower is $(n+1)F.d_{n+1}$.

Secondly, this moment can be expressed as

$$nF.d_n + F\left(d_n + \frac{L}{2}\right)$$

where $nF.d_n$ is the moment of the gravitational force of $n$ top blocks ($d_n$ is the distance of the centre of gravity of the tower of $n$ blocks) and $F.(d_n+L/2)$ is the moment of the gravitational force of one additional block (see Fig.11).

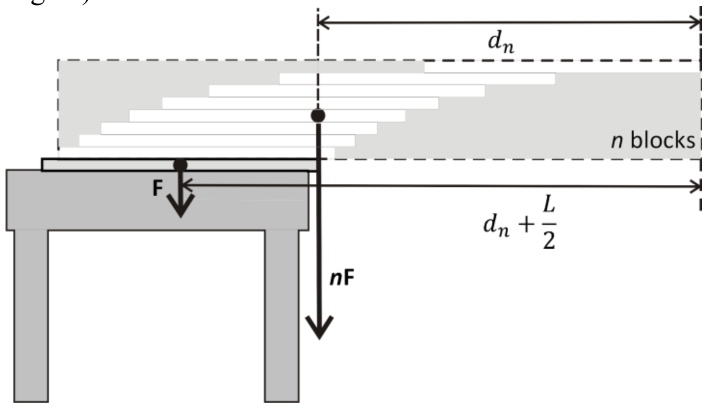


Fig.11 Formula $F(d_n+L/2)+nFd_n$ represents the total moment of the gravitational force of the leaning tower, containing $n$+1 blocks, with respect to the right edge of the tower.

Since the both above given formulas express the same we get the following equation

$$(n+ \ )Fd_{n+} = nFd_n + F\left(d_n + \frac{L}{2}\right) \qquad (2)$$

## 3.7 Mathematical point of view – recurrence sequences

The latter equation (2) can be solved for $d_{n+1}$, so the following recurrence relation can be obtained

$$d_{n+} = l_n + \frac{1}{2(n+ \ )}L, d_1 = \frac{L}{2}.$$

This recursively given sequence can be further analysed either directly by methods of mathematics, see e.g. [2], or by the CAS (see thereinafter in the section 5).

## 4 Recursively Given Sequences

The problem of leaning tower brings us to the concept of recursively given sequences; an important concept that is sometimes not sufficiently explained to students. In the rest of the paper, let us illustrate a possible way of teaching this interesting topic using IT, namely the Maple environment.

It is a well known fact that *sequences* are defined either by explicit *formulas* for the $n$-th terms or *recursively* which means that the previous values of terms of the sequence are used to generate the value of the next term of the sequence. In other words one needs to know the first couple of values of the given sequence and the recurrence construction to be able to find subsequent terms of the sequence.

Let us briefly remind a basic mathematical formulation of recursively defined sequences. Given a real function *f*: $N_0 \times R \to R$ and an *initial value* $x_0 \in R$, consider the sequence of *iterates* of $x_0$ under the given function *f*:

$$x_0,$$
$$x_1 = f(0, x_0),$$
$$x_2 = f(1, f(0, x_0)),$$
$$x_3 = f(2, f(1, f(0, x_0))),$$
$$\ldots$$

Such a sequence is called *recursively defined sequence* and it is evident that its terms can be written in a more concise form as

$$x_0,$$
$$x_1 = f(0, x_0),$$
$$x_2 = f(1, x_1),$$
$$x_3 = f(2, x_2),$$
$$\ldots$$

It means that for given $x_0$, we can compute an arbitrary term of sequence $x_t$, where $t = 0, 1, 2, \ldots$, according to the relation

$$x_{t+1} = f(t, x_t).$$

This relation, together with an initial value $x_0$ is called a *first order recurrence equation.* The *solution* to this equation is any sequence $x_t$ generated by this equation. It is worthwhile to notice that the given formulation is not the most general one but for the purposes of our consideration it is fully sufficient.

Recursively defined sequences have a lot of interesting applications. Moreover, with the help of IT and especially with the help of CAS it is possible to present such concepts as stationary point, cobweb plot, stability, parameter sensitivity, bifurcation or chaos for instance, for more details see [12]. These concepts could be presented graphically and one of the main advantages of the graphic approach is to see parameter dependence behaviour of problems.

Although there exists this quick way how to present various applications and recent results that involve recurrence relations, cf. [10], teachers not always either know it or devote time to prepare and present such applications to their students.

In the following sections we describe a kind of animation that we prepared for students to enrich their knowledge about recursively defined sequences and, in this way, to motivate them for deeper interest in mathematics problems.

## 5 Solving Recurrence Equation with CAS

Sometimes it is possible to find a simple explicit formula for solution $x_t$ to a given recurrence equation, but more often it is not possible. Students of basic courses of mathematics are usually able to find these formulas only for arithmetic and geometric sequences. If we want to show them more, we can use CAS at the beginning. Particularly in the Maple environment it is possible to use the command called `rsolve`.

For instance we can consider the following recurrence equation

```
> eq1:=x(t+1)=a*x(t)+b,
```

where *a* and *b* are real constants. This equation is known as *linear recurrence equation* of the first order and the function *f*: $y = ax+b$ as *linear map*. If $a = 1$ we obtain an arithmetic sequence. If $b = 0$ and $a \notin \{0,1\}$ we obtain a geometric sequence. The given equation has the constant solution

$$p = \frac{b}{1-a}$$

that is called an *equilibrium state* or a *stationary point.* To find a general formula for solution to the linear recurrence equation we can simply use

```
> rsolve(eq1, x).
```

Particularly for $a \neq 1$ we get

$$x_t = a^t\left(x_0 - \frac{b}{1-a}\right) + \frac{b}{1-a}.$$

Once we have a formula we can start to study how the behaviour of the solution changes when values of parameters are changed. The most straightforward graph is to plot $x_t$ as a function of positive integer $t$ – such graph is called a *time graph*. We prepared an animation of time graphs that demonstrates changes in the behaviour of the solution. One of these parameter dependence observations is shown in Fig. 12. Please note that this figure is only one graph that is a part of the set of graphs generating animation.

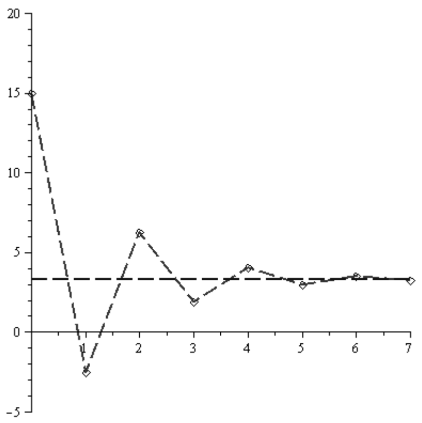


Fig. 12 Solution to a linear recurrence equation. Sequence $x_t$ exhibits damped oscillations around stationary point $p$, $x_0 > p = b/(1-a)$, $-1 < a < 0$, $b > 0$.

With the prepared animation it can be easy to demonstrate that the sequence $x_t$ that solves linear recurrence equation of the first order can either decrease monotonically and *converge to the equilibrium state p,* or it can exhibit *damped oscillations* around *p,* or it can *diverge* to $+\infty$, $-\infty$, or, finally, it can exhibit *explosive oscillations* around the stationary point *p*.

Now let us get back to the problem of leaning tower described in the section 3. We have showed that the maximum overlap distance of a tower of *n* identical homogenous blocks can be expressed by the linear recurrence equation

$$d_{n+} = l_n + \frac{1}{2(n+\ )}L, d_1 = \frac{L}{2}.$$

If we use the program Maple, particularly its command `rsolve` and put L=1, we obtain a formula that is abstruse for secondary school students, because it involves the gamma function. It is the main reason why the graphical output, which can be easy obtained by the following sequence of commands, is preferred:

```
> with(plots):
> eq:=d(n+1)=d(n)+1/(2*(n+1)):
> rsolve({eq, d(1)=1/2}, d(n)):
> f:=unapply(%, n);
> posl:=[seq([n, f(n)], n=1..10)]:
> graph:=plot(posl, linestyle=3):
> points:=pointplot(plot, symbolsize=16,
  symbol=circle):
> display([graph, points]);
```

The result can be seen in Fig. 13.

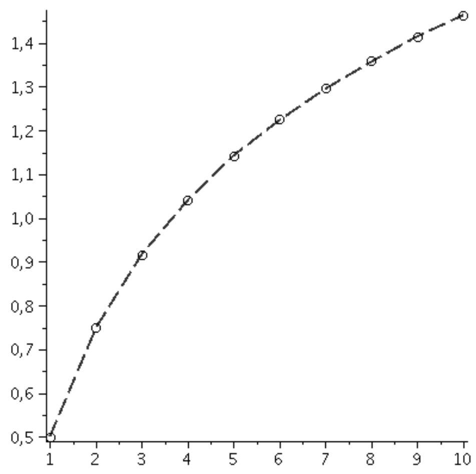


Fig. 13 Graph of the solution to the linear recurrence equation of leaning tower.

It is possible to experiment with many recursively defined sequences and it is not difficult to find that recurrence equations exist for which it is not possible to find formulas for *n*-th term. One of them is the *logistic equation*; see Fig. 14. We are going to deal with this equation thereinafter.

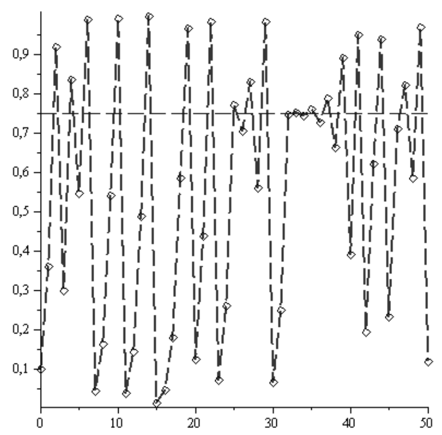


Fig. 14 A turbulent development of solution $x_t$ to the logistic equation $x_{t+1}$=3.99027 $x_t$(1-$x_t$), $x_0$=0.1.

## 6 Recurrence Equation and Cobweb Plot

A simple graphical technique called *cobweb plot* can help us to study the behaviour of solutions to recurrence equations. We prepared an animation that instructively demonstrates both the construction of the cobweb plot and the behaviour of the solution to a given recurrence equation. We considered a less general recurrence equation than in the previous part, particularly we consider that there is an initial value $x_0$ and a recurrence rule $x_{t+1} = f(x_t)$, where *f* is a map from [0, 1] on [0, 1], but not necessarily a linear one. Such a problem is called an *autonomous recurrence equation* and in this case it is possible to construct a plane *coordinate system* with the horizontal axis as $x_t$ and the vertical axis as $x_{t+1}$. Now we can review the construction of a cobweb plot. First we plot the graph of the function *f* as well as the diagonal line $y = x$. Then we draw a vertical segment that is drawn from $x_0$ to the intersection with the graph of the function. This intersection has a corresponding ordinate $x_1 = f(x_0)$ it means we obtain a point ($x_0$, $x_1$). In the next step we draw a horizontal segment from this point to the diagonal. The point of intersection has the abscissa $x_1$, it means we obtain a point ($x_1$, $x_1$). Now we can repeat the same steps and we subsequently obtain terms of recurrently defined sequence $x_2$, $x_3$, $x_4$, ... A screen capture of the animation can be seen at Fig. 15. More iteration for different map is given at Fig. 16.

Notice that the intersection of the graph of the function *f* and the diagonal is a stationary point p that can be considered as a solution to the equation $x = f(x)$. If the cobweb spirals inward, then the stationary point is stable, and if it spirals outwards, then it is an unstable stationary point. If the stationary point of a given recurrence equation is its limit, the stationary point is called *asymptotically stable*.

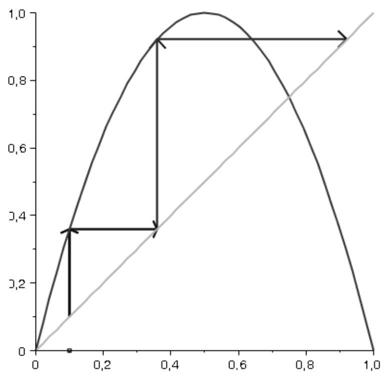


Fig. 15 One screen of animation, that shows a construction of cobweb plot for the logistic equation $x_{t+1}=4x_t(1-x_t)$.

Moreover we can see that the iteration process is a way how to find a root of the equation $x = f(x)$. The approximation of the limit of the iteration process and therefore the root of the latter equation can be found with Maple. For evaluation we can use the first 100 iteration, then we simply use the command

```
> stationary_point:=(f@@100)(x0)
```

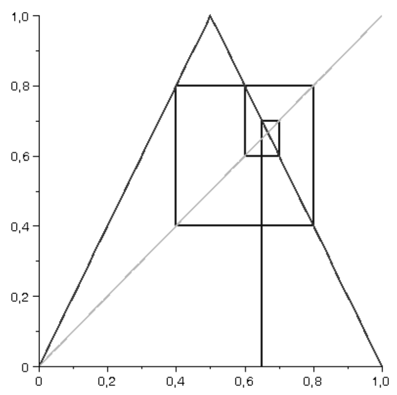


Fig. 16 A more detailed cobweb plot for the recurrence equation $x_{t+1}=1-2|x_t-1/2|$, $x_0=0.64$, 2-cycle {0.8, 0.4}.

The sequence of iterates of $x_0$ under the function $f$: $x_0, x_1=f(x_0), x_2=f(x_1)=f^{(2)}(x_0), x_3=f(x_2)=f^{(3)}(x_0), \dots$ can exhibit many different phenomena. To be able to study these phenomena of recurrence equations, we will need the following concept: a string of distinct points

$$p_1, p_2, \dots, p_K,$$

where $K \in N$ and $N$ is the set of all natural numbers, forms a *cycle of length K* or $K$-cycle if $f^{(K)}(p_i)=p_i$ for all $1 \le i \le K$, and points $p_i$ are called *periodic points* of $f$ with period $K$. It means that each periodic point with period $K$ is a stationary point of the derived map $f^{(K)}$. This concept can be also visualized by cobweb plot, cf. Fig. 16, where we can observe 2-cycle. The derived map has very important consequences in behaviour patterns.

## 7 Logistic Equation

Let us consider the following recurrence equation

```
> eq2:=x(t+1)=a*x(t)*(1-x(t)),
```

which is known as *logistic equation* and the function $f$: $y = ax(1-x)$, $x \in [0,1]$ that generates its right hand side is known as *logistic map*. If we choose $a = 2$ then it is possible to find a formula for solution but for other values the Maple command `rsolve` does not find any formulas. To find a time plot of the solution to this equation it is necessary to use recurrence formula directly.

We prepared an animation of several time plots which is suitable for studying of behaviour of solutions to logistic equation that depends on the value of the parameter $a$. The time plot for $a = 3.99027$ can be find at Fig. 17.

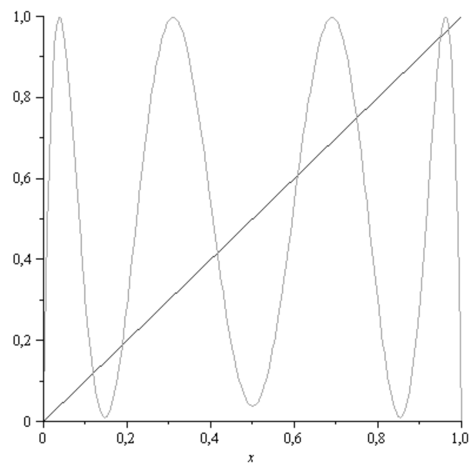


Fig. 17 Graph of $f^{(3)}$, for f: $y= 3.99027\,x(1-x)$, except the stationary point we can see periodic points with period 3 that can be found on the line $y=x$.

A logistic map is a simple quadratic function. This can be a reason that the logistic recurrence equation looks innocuous at the first glance. But in [5] it was noticed, that the sequence of iterates of logistic maps exhibit many unexpected phenomena. We concentrate here only to the phenomena that are concerned with the number of $K$-cycles depending up the values of parameter $a \in [0,4]$. For this reason, we prepared an animation of maps $f^{(k)}$, for different values of $k$ and different values of $a$. Intersections graphs of $f^{(k)}$ and diagonal $y = x$ denote periodic points with period $k$, one of the screen can be seen at Fig. 17.

The prepared animation can also help us to present a familiar statement that “period three implies chaos”, cf.

[4]. The number of *k* periodic points can be also depicted at bifurcation diagram.

## 8 Summary

Let us go back to the first part of the paper and recall the possibility of various ways of students' algorithmic thinking development. The ability to create algorithms in Maple environment to get graphical descriptions of the explained matter is another valuable algorithmic challenge for students, especially for future teachers.

Let us finish our paper with a sample of an algorithm, using Maple commands, constructing the bifurcation diagram (see Fig.18).

```
>with(plots):
>posl:=[]:
>init:=0.1:
>f:=(a, x)->a*x*(1-x):
>for r from 1 by 0.01 to 4 do
      value:=init:
      for j from 1 to 300 do
            value:=f(r, value)
      end do;
      for k from 301 to 350 do
            value:=f(r, value):
            bod:=[r, value]:
            posl:=[op(posl), bod]:
      end do;
end do;
>plot(posl,1..4,0..1,style=point,
symbol=POINT,symbolsize=5, labels=[a, x]);
```

The algorithm can be briefly described as follows:

1) Choose the value *a* of the parameter of the logistic function;

2) Choose an initial value of *x* (it can be a random number from [0, 1]);

3) Calculate a few first iterations and ignore them, let us say 300 iterations;

4) Calculate a few following iterations and plot them, let us say from 301 to 350;

5) Increment the value *a* of the parameter of the logistic equation, etc.

If we make small changes to the algorithm it also allows us to study different maps with different domains, so we can freely see and study details of different bifurcation diagrams.

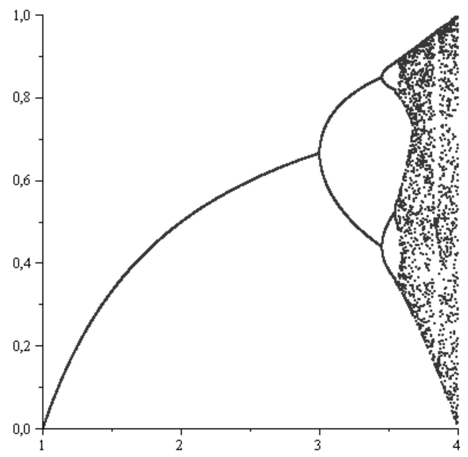


Fig 18 Bifurcation diagram for logistic map, it summarize number of either stable stationary points or *k*-periodic points for different values of parameter *a*.

## 9 Conclusion

*Information technology has changed many things in the world. Computers, networks are used widely in almost everything. Anytime and anywhere are the slogans of people living in the 21st century.*[11]

In the paper we offered our teaching/learning strategy as an inspiration for all who still are looking for a way to explain these, in the paper discussed, interesting and challenging topics and support interdisciplinary learning, which is considered to be a very important part of education of future teachers.

***Acknowledgements***
This research has been partially supported by the Specific research project of the Faculty of Education of University of Hradec Králové No. 2110.

*References:*

[1] Fenclová, J.: *Didaktické myšlení jednání učitele fyziky*. SPN, Praha 1984.

[2] Graham R.L., Knuth D.E., Patashnik O.: *Concrete Mathematics*, Addison-Wesley Longman, Reading, MA, 1988

[3] Hubálovský, Š., Milková, E.: Modeling of a real situation as a method of the algorithmic thinking development. In: *Advanced Educational Technologies, Proceedings of 6th WSEAS/IASME International Conference on Educational Technologies* (EDUTE'10), WSEAS Press, Tunisia, May 3-6, 2010, pp. 68–72.

[4] Li T.Y., Yorke J.: Period three implies chaos, *The American Mathematical Monthly*, Vol. 82, No. 10, 1975, pp. 985-992.

[5] May, R. M.: Simple mathematical models with very complicated dynamics. *Nature*, Vol. 261, No. 5560, 1976, pp. 459-467.

[6] Milková, E.: *Algoritmy -Objasnění, procvičení a vizualizace základních algoritmických konstrukcí*. Alfa nakladatelství, Praha 2008, pp. 114. (in English: Algorithms - Explanation, Exercises and Visualization of the Basic Algorithmic Constructions)

[7] Milková, E.: Algorithms: The Base of Programming Skills. In: *ITI 2007 Proceedings of the 29th International Conferences on INFORMATION TECHNOLOGY INTER-FACES*, University of Zagreb, Croatia 2007, pp. 765 – 770.

[8] Milková, E.: Multimedia Applications and their Benefit for Teaching and Learning at Universities. In: *WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS*, Issue 6, Volume 5, June 2008, pp.869-879.

[9] Prazak P.: Recursively Defined Sequences and CAS, In: *Advanced Educational Technologies, Proceedings of 6th WSEAS/IASME International Conference on Educational Technologies* (EDUTE'10), WSEAS Press, Kantoui, Sousse, Tunisia, May 3-6, 2010, pp. 58–61.

[10] Prazak P., Prazakova B.: Excel and recursively defined sequences in financial mathematics, *Matematika fyzika-informatika,* Vol. 18, No. 1, 2008, pp. 45-51. (in Czech)

[11] Sirkemaa, S., Implementing Information Technology in the Learning Process, In: *6th WSEAS International Conference on E-ACTIVITIES*, Tenerife, Spain, 2007, pp. 263-267.

[12] Verhulst F.: *Nonlinear Differential Equations and Dynamical Systems*, Springer, 1990.

[13] Voborník P., *Programovací jazyk pro podporu výuky algoritmů*, Hradec Králové, thesis, 2006.