Recursive Computations in Spreadsheets

Jozef Hvorecky
Department of Mathematics
University of Papua New Guinea
hvorecky@upng.ac.pg

Ivan Trencansky
Department of Applied Informatic
University of Economics Bratislava
ivo@euba.sk

Abstract

In the paper two methods of implementation of recursive formulas in spreadsheets are presented. Both methods use (rectangular) fields of cells for their computations. First method is "direct": Initial values (corresponding to trivial cases of recursion) are written in selected fields. Then, the recursive formula is written to other(s). Finally, the formula is spread along a necessary large area - until the required result is reached. Second method allows to compute "recursive tables"- tables where the result of a two-argument recursive function is located on the intersection of the corresponding rows and columns. In this case each field in the area must contain the complete recursive formula. The computation is iterative - all values in the table are computed "simultaneously". Actually, the number of iterations equals to the maximum depth of recursion. Examples of such computations as well as sufficient conditions for the success of the method will also be presented.

1 Role of Standard Software

One of the main obstacles in a more intensive creation of educational software is that special programming tools and long time for its design and implementation are usually needed. Thus, teachers do not prepare their own courseware, even if they are not satisfied with professional products and willing doing it. In this paper we can exemplify that standard software can sometimes become handy.

There are good reasons for its preference:

- Standard software is relatively cheap.
- Its manuals are widely available.
- Many people can advice the author if he/she is not a computer specialist.
- The same software package can be used for a variety of courseware. For instance, spreadsheets (the subject of our paper) can also be used for the explanation of problems from accountancy, presentation of functions in science, various mathematical computations etc.

In our paper we use spreadsheets in rather uncommon way – for a special kind of recursive computation where the result of the computation in a cell depends on its location in the worksheet. The examples have been used at the University of Economics in Bratislava. They should show to students that spreadsheets are much more powerful computation tool than people usually think.

2 Basic Notions

A spreadsheet program is a tool for (pseudo) parallel computations over large groups of numbers. The numbers are located in a ***sheet*** – a matrix with ***rows*** and ***columns***. The elements of the matrix are called ***cells***. The maximum number of rows and columns depends on a particular software product. In our paper we limit it to a *finite* number only. Usually, the letters identify the columns. Single letters (A, B, …, Z) denotes the first 26 columns, their pairs (AA, AB, AC, …, AZ, BA, BB, …) the following ones. The numbers (1, 2, 3, …) identify the rows.

The manipulation with the spreadsheet values is usually done in three steps:

- The ***initialization*** of (a relatively small number of) cells.
- The ***definitions of relationships*** between the initialized cells and some selected ones.
- ***Spreading*** of the relationship through the sheet.

The user has to initialize cells, define the relation and select the cells the relation is to be spread to. Thus, the computer only executes a part of the third step. But, the

real benefit can be substantial as an arbitrary large area of the sheet can be bound by the same relationship.

The fourth important property of spreadsheets is their ***adaptability***. When a value of a cell is changed, the values of all other related cells are recalculated. The effect of any small change is spread to the rest of the sheet immediately and automatically. We will use the adaptability for complex iterative computations.

3 Simple recursive computations

The simplest recursive computations exactly follow the above (initialize – define – spread) pattern. The factorial of a natural number is one of the most popular recursive functions:

$$n! = \begin{cases} 1 & \text{if } n = 1 \\ n*(n-1)! & \text{if } n > 1 \end{cases}$$

1
2
6
24
120
720
5040
40320
362880
3628800
39916800
479001600
6227020800
87178291200
1.30767E+12
2.09228E+13

Figure 1

Assume the results should occupy the first column of the sheet. To compute that, one has to initialize the first cell in the first row (i.e., A1). Then he/she has to define the relationship for the cell in the second row. Because the cell A1 already contains a result (the factorial of 1) and we need to compute the factorial of 2, the relationship has to express: *Multiply the content of the previous row by 2*. To generalize this expression we prefer the form: *Multiply the content of the previous row by the order number of the given row*. This is expressed as follows:

```
= ROWS($A$1:A2)*A1
```

where the function ROWS computes the number of rows between A1 and the given cell.

This expression can be spread as far as we need (Figure 1).

4 Recursive functions with several variables

Euclid's algorithm for the greatest common divisor has the form:

$$GCD(a,b) = \begin{cases} a & \text{if } a = b \\ GCD(a-b,b) & \text{if } a > b \\ GCD(a,b-a) & \text{if } a < b \end{cases} \qquad (1)$$

This computation uses two columns. The first one (the column A) contains values of the variable *a*, the second (B) the values of *b*. The cells in their first rows have to be initialized –for example to 84 (in A1) and 35 (in B1). Putting the formula

```
=IF(A1>B1;A1–B1;A1)
```

into cell A2 and the formula

```
= IF(B1>A1;B1–A1;B1)
```

into the cell B2 and spreading the both along their respective columns starts the computation. Its result is shown in Figure 2. Starting from the sixth row the contents of cells repeat. The repeating value is the result (the greatest common divisor of A1 and B1).

Compared to the previous example, it is harder to predict how far the relationships are to be spread. For we discuss the following problems with students:

- Will every computation starting with pairs of natural numbers (in the first row) produce a pair of the identical numbers in a row?[1]
- If yes, what is the maximum length of the sequence of different pair (as a function of *a* and *b*)?
- Does each successive pair of numbers (in the same row) have the same greatest common divisor? Why?

	A	B
1	84	35
2	49	35
3	14	35
4	14	21
5	14	7
6	7	7
7	7	7
8	7	7

Figure 2

The answers require a good comprehension of the high school math and are a preparatory stage for the next step. The adaptability of the spreadsheet (the change of the values in the first row) can be used for a description of different nuances of the process. On the other hand, it is quite difficult to adapt such a computation using just a change of A1 and B1, because the number of rows strongly varies for different pairs. The computation of the greatest common divisor of another pair of numbers ought to require (much) more rows.

5 Iterative computations

Now we construct Euclid's table – the table that contains the greatest common divisor of ***r*** and ***s*** in the cell on the crossing of ***r***-th row and ***s***-th column. The

[1] If the answer to this question would not be always positive, we could not use the method with spreadsheets. Some computations would require an infinite number of rows.

computation is based on a (rather paradoxical) idea: *No cells are initialized. Each cell contains the complete equation Equation 1. The content of all cells is computed simultaneously* (by iterations). The length of computation depends on the number of iterations.

All cells at the main diagonal (A1, B2, C3, …) have $r = s$. Thus, the greatest common divisor equals to this value and can be computed using the function ROWS.

Cells with $r > s$ are located under the main diagonal. The corresponding greatest common divisor is the same as of the cell located in the same (r-th) column and s rows higher. The function INDEX can be used for copying the value into the cell: INDEX(*array*, *r*, *s*) returns the value of the cell in *r*-th column and *s*-th row (when the array is specified from A1 to the given cell).

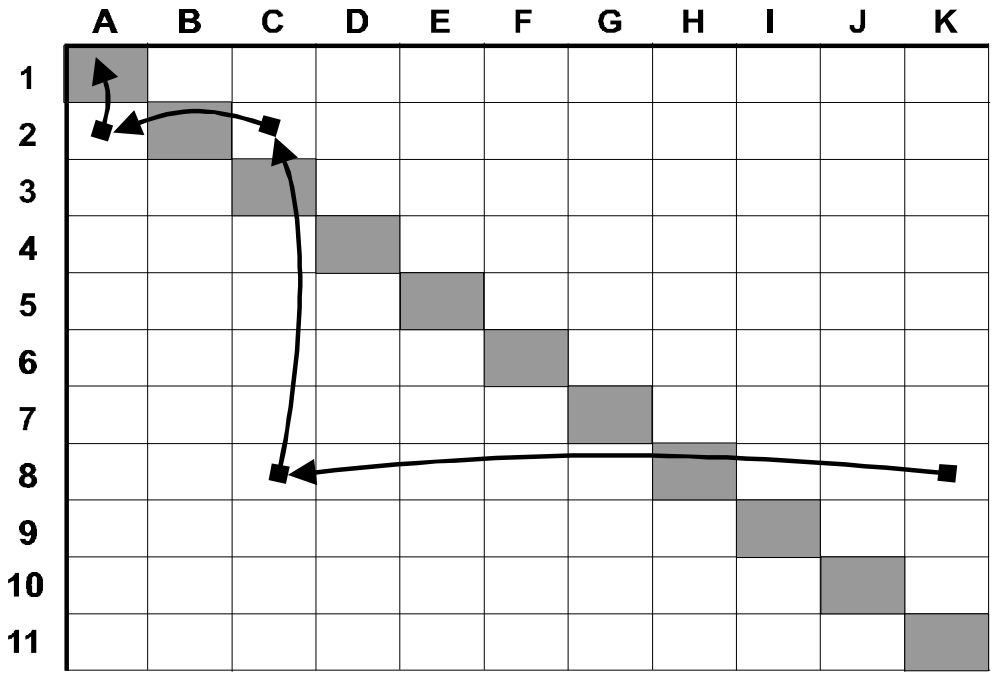

Figure 3

Similarly, the value of a cell above the main can be taken from the cell in the same row, but *s* cells to the left. The connection between selected cells in shown in Figure 3. It shows the computation of the value of the cell K8 (in the 8th row and 11th column). The value is to be taken from C8 (the 8th row and 3rd column). Its value is to be taken from C2, etc. up to A1, because their greatest common divisor is 1. In fact, the real order of the assignments is just opposite: All values of all cells on the main diagonal are computed in the first iteration "wave" – the value of A1 among them. In the second wave they are copied into "neighboring" cells – those pointing to the main diagonal. Thus, A2 gets its value in the second wave, C2 in the third, C8 in the fourth and K8 in the fifth.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
3	1	1	3	1	1	3	1	1	3	1	1	3	1	1	3	1	1	3	1	1
4	1	2	1	4	1	2	1	4	1	2	1	4	1	2	1	4	1	2	1	4
5	1	1	1	1	5	1	1	1	1	5	1	1	1	1	5	1	1	1	1	5
6	1	2	3	2	1	6	1	2	3	2	1	6	1	2	3	2	1	6	1	2
7	1	1	1	1	1	1	7	1	1	1	1	1	1	7	1	1	1	1	1	1
8	1	2	1	4	1	2	1	8	1	2	1	4	1	2	1	8	1	2	1	4
9	1	1	3	1	1	3	1	1	9	1	1	3	1	1	3	1	1	9	1	1
10	1	2	1	2	5	2	1	2	1	10	1	2	1	2	5	2	1	2	1	10
11	1	1	1	1	1	1	1	1	1	1	11	1	1	1	1	1	1	1	1	1
12	1	2	3	4	1	6	1	4	3	2	1	12	1	2	3	4	1	6	1	4
13	1	1	1	1	1	1	1	1	1	1	1	1	13	1	1	1	1	1	1	1
14	1	2	1	2	1	2	7	2	1	2	1	2	1	14	1	2	1	2	1	2
15	1	1	3	1	5	3	1	1	3	5	1	3	1	1	15	1	1	3	1	5
16	1	2	1	4	1	2	1	8	1	2	1	4	1	2	1	16	1	2	1	4
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	17	1	1	1
18	1	2	3	2	1	6	1	2	9	2	1	6	1	2	3	2	1	18	1	2
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	19	1
20	1	2	1	4	5	2	1	4	1	10	1	4	1	2	5	4	1	2	1	20

Figure 4

This is expressed by the formula:

```
=IF(ROWS($A$1:A1)=COLUMNS($A$1:A1); ROWS($A$1:A1);
   IF(ROWS($A$1:A1)>COLUMNS($A$1:A1);
     INDEX($A$1:A1; ROWS($A$1:A1)-COLUMNS($A$1:A1);
      COLUMNS($A$1:A1)),
     INDEX($A$1:A1, ROWS($A$1:A1),COLUMNS($A$1:A1)-
     ROWS($A$1:A1))))
```

When it is written in the cell A1, it produces the value 1. When it is spread along the sheet, the execution generates the greatest common divisor in all activated cells. In Figure 4 the relation was spread along the area of 20 x 20 cells.

6 Limits of spreadsheets computations

There exists one important restriction on this type of recursive computations: *No cell can refer to a cell outside the active area*. It is clear why: *The cells outside the area have no values, nor formulas. Thus, the computation can not be completed*. If such a danger exists, the user should find out a "safe" area (with all references pointing into it) or do another type of computation.

In this section we discuss such a case using a problem from [Hvorecký, 1990]: *Given a natural number* $\boldsymbol{p_0}$*. Split its digits and sum their squares. The result is another number* $\boldsymbol{p_1}$*. Repeat the procedure with* $\boldsymbol{p_1}$*, then with* $\boldsymbol{p_2}$ *etc. as long as 1 or 4 is produced. Denote* $\boldsymbol{L(p_0)}$ *the length of the sequence. Generate a (linear) table that will contain* $\boldsymbol{L(n)}$ *in its* $\boldsymbol{n}$*-th cell.*

For instance, the sequence for 54 looks as follows:

$5^2 + 4^2 = 25 + 16 =$ **41**,
$4^2 + 1^2 = 16 + 1 =$ **17**,
$1^2 + 7^2 = 1 + 49 =$ **50**,
$5^2 =$ **25**,
$2^2 + 5^2 = 4 + 25 =$ **29**,
$2^2 + 9^2 = 4 + 81 =$ **85**,
$8^2 + 5^2 = 64 + 25 =$ **89**,
$8^2 + 9^2 = 64 + 81 =$ **145**,
$1^2 + 4^2 + 5^2 = 1 + 16 + 25 =$ **42**,
$4^2 + 2^2 = 16 + 4 =$ **20**,
$2^2 =$ **4**.

Consequently, L(54) = 11. The proof of the termination of the sequence for any natural number p_0 is done in [Hvorecký, 1990]. One of the ways of computation of L(p_i) is:

$$L(p_i) = \begin{cases} 0 & \text{if } p_i = 1 \\ 0 & \text{if } p_i = 4 \\ L(p_{i+1}) + 1 & \text{otherwise} \end{cases}$$

Thus, when a table is generated, the value 0 is to be stored in the first and fourth cell. The value in all other cells is to be computed by adding 1 to the value stored in their successor in the sequence. For instance, the value in 54th cell is the value of 41st cell increased by 1.

As we have seen above, the successor of a number is sometimes be bigger than it. This implies that in a linear table some values refer to the "left", the others to the

"right". The spreadsheet table can only be created for those groups, which values do not refer outside the group. Now we will show that the smallest such group for our problem contains 162 first natural numbers.

First, the group has to contain 1 and 4, because they are the trivial cases of the recursive formula and all other values (finally) refer to one of them. The group has to contain all one-digit numbers, because square of all of them is bigger than the number[2]. Thus, the group has to have at least 81 members.

As we already shown, some two-digit numbers refers to three-digit ones. For instance, the number 79 refers to 130. Thus, all two-digit numbers have also to be in the group. The biggest of them – 99 – refers to 162.

On the other hand, it is easy to prove that three-digit numbers only refer to smaller ones. Each three-digit number has a form:

$$a * 100 + b * 10 + c$$

where

$$1 \le a \le 9,$$
$$0 \le b \le 9,$$
$$0 \le c \le 9.$$

Let us prove that its successor is a smaller number:

$$(a * 100 + b * 10 + c) > a^2 + b^2 + c^2$$
$$(a * (100 - a + a) + b * (10 - b + b) + c) > a^2 + b^2 + c^2$$
$$((a^2 + a * (100 - a)) + (b^2 + b * (10 - b)) + c) > a^2 + b^2 + c^2$$
$$(a * (100 - a) + b * (10 - b)) > (c^2 - c)$$

The left side has its smallest possible value for $a = 1$ and $b = 0$. In this case the result is 99. The right side has its biggest possible value – 72 – for $c = 9$. Thus, the relationship always holds[3].

Our table will be vertical (placed in the column A). First we define a function *smdc* having one argument p_0. The function computes p_1 and returns the value of the cell in p_1-th row increased by 1:

---

[2] The only exception could be for 1, but our previous reasoning requires its membership in the group.

[3] A similar proof for the numbers with four and more digits is even easier. If the number has n digits, its value is at least 10^n. On the other hand, it refers to the number not bigger than $n * 81$. Evidently, $10^n > n * 81$.

```
scdm
=ARGUMENT("number";1)
=RESULT(1)
=SET.NAME("sumsquares";0)
=SET.NAME("tt";TEXT(number;0))
=SET.NAME("length";LEN(tt))
=FOR("counter";1;length)
=SET.NAME("vv",VALUE(MID(tt;counter;1)))
=SET.NAME("sumsquares";sumsquares+vv*vv)
=NEXT()
=RETURN(sumsquares)
```

The value 0 is set into cells A1 and A4 and the formula and

```
=INDEX(A$1:A162;scdm(A2);2)+1
=INDEX(A$1:A163;scdm(A2);2)+1
```

into A2 and A3. Then spread it also into cells A5 to A162. This ("minimum") table is shown in Figure 5.

1	0	**22**	10	**43**	8	**64**	8	**85**	5	**106**	7	**127**	12	**148**	10
2	1	**23**	3	**44**	4	**65**	8	**86**	2	**107**	9	**128**	13	**149**	5
3	11	**24**	2	**45**	11	**66**	11	**87**	4	**108**	9	**129**	3	**150**	10
4	0	**25**	7	**46**	8	**67**	6	**88**	14	**109**	4	**130**	2	**151**	11
5	8	**26**	9	**47**	9	**68**	2	**89**	4	**110**	2	**131**	3	**152**	12
6	13	**27**	10	**48**	10	**69**	12	**90**	10	**111**	12	**132**	11	**163**	10
7	5	**28**	3	**49**	4	**70**	5	**91**	4	**112**	14	**133**	5	**154**	3
8	9	**29**	6	**50**	8	**71**	9	**92**	6	**113**	3	**134**	10	**155**	11
9	10	**30**	11	**51**	10	**72**	10	**93**	11	**114**	10	**135**	10	**156**	10
10	1	**31**	2	**52**	7	**73**	6	**94**	4	**115**	11	**136**	9	**157**	11
11	2	**32**	3	**53**	9	**74**	9	**95**	8	**116**	8	**137**	9	**158**	11
12	9	**33**	10	**54**	11	**75**	10	**96**	12	**117**	11	**138**	10	**159**	10
13	2	**34**	8	**55**	9	**76**	6	**97**	3	**118**	12	**139**	5	**160**	7
14	10	**35**	9	**56**	8	**77**	5	**98**	4	**119**	8	**140**	10	**161**	8
15	10	**36**	12	**57**	10	**78**	4	**99**	12	**120**	9	**141**	10	**162**	11
16	7	**37**	6	**58**	5	**79**	3	**100**	1	**121**	14	**142**	10		
17	9	**38**	7	**59**	8	**80**	9	**101**	2	**122**	11	**143**	10		
18	9	**39**	11	**60**	13	**81**	9	**102**	9	**123**	11	**144**	11		
19	4	**40**	8	**61**	7	**82**	3	**103**	2	**124**	10	**145**	3		
20	1	**41**	10	**62**	9	**83**	7	**104**	10	**125**	12	**146**	10		
21	9	**42**	2	**63**	12	**84**	10	**105**	10	**126**	11	**147**	12		

Figure 5

7 Conclusions

We have shown that the spreadsheets represent a very convenient tool for recursive computations. Some restrictions, which limit their usage, can often be overcome by a correct analysis of the problem. However, there always will exist computations that simply can not be made in this environment – partially those, in which some cells refer to cells outside the active area.

References

[1] Hvorecký, J.: On a Connection between Programming and Mathematics. *SIGCSE Bulletin*, Vol. 22, No. 4, Dec. 1990.