Spreadsheets in Education (eJSiE)

Volume 6 | Issue 3 Article 1

10-13-2013

Letters & Numbers in Excel: fostering student engagement in some fundamental concepts of mathematics & computing

Steve Sugden
Queensland University of Technology, ssugden@bond.edu.au

Phil A. Stocks
Bond University, pstocks@bond.edu.au

Letters & Numbers in Excel: fostering student engagement in some fundamental concepts of mathematics & computing

Abstract
The television quiz program *Letters and Numbers*, broadcast on the SBS network, has recently become quite popular in Australia. This paper considers an implementation in Excel 2010 and its potential as a vehicle to showcase a range of mathematical and computing concepts and principles.

Keywords
Letters & Numbers, student engagement, Excel, expression tree, letter bag, Delphi

Letters & Numbers in Excel: fostering student engagement in some fundamental concepts of mathematics & computing

Stephen Sugden[1]
School of Mathematical Sciences
Queensland University of Technology, Brisbane, Australia
Phone +61 7 3138 6979
Email stephen.sugden@qut.edu.au

Phil Stocks
Division of Informatics
Bond University, Gold Coast, Australia
Phone +61 7 559 53354
Email pstocks@bond.edu.au

1 Abstract

The television quiz program *Letters and Numbers*, broadcast on the SBS network, has recently become quite popular in Australia. This paper considers an implementation of algorithms to solve both components of *Letters and Numbers* in Excel 2010 and also the potential of the game as a vehicle to showcase a range of mathematical and computing concepts and principles.

Keywords: Letters & Numbers, student engagement, expression tree, letter bag, Excel, Delphi.

2 Introduction

The television quiz program *Letters & Numbers*, broadcast on the SBS network, has become popular in Australia over the past few years. This paper looks at the game's potential as a vehicle to illustrate a range of fundamental concepts of computer science and mathematics and to capture student interest. The Numbers Game in particular has a very rich mathematical structure whose analysis and solution involves concepts of counting and problem size, discrete (tree) structures, expression trees, expression tree evaluation, recurrences, and many other bedrock topics. For further discussion, see (Sugden & Stocks, 2013). Students tend to like games, even maths games, as part of learning new theory (Bellotti, Bottio, & Nadolski, 2013). A brief analysis of both components of *Letters & Numbers* is presented and connections to common topics in Discrete Mathematics, Mathematics of Computing, and Computer Science are made. Again, further discussion on the game's use as

[1] Corresponding author

a basis for assignment work in a Discrete Mathematics class at Bond University may be found in (Sugden & Stocks, 2013).

3 Letters game

3.1 *Description of the game*

Nine letters of the alphabet are chosen, with one contestant choosing a sequence of *vowels* or *consonants*. Duplicates are possible. Then both contestants are given 30 seconds in which to form the longest possible English word from a subset of the chosen letters. Such words must either exist in the standard dictionary used on the program or be well-defined extensions of dictionary words.

3.2 *Solution in Excel/VBA*

We now describe how to solve the problem in Excel, with the aid of VBA. The first step was to find a large dictionary, i.e., collection of words, just as text. The idea was to put these into one or more Excel sheets and use some form of lookup, either with lookup functions, or with VBA, to find working compatible with the chosen letters. Finding the word was not difficult, as on-line dictionaries are quite common, as a public-domain wordlist was downloaded into one Excel sheet, which we called "lexicon".

The user enters nine letters into cell A2 of the Letters worksheet. There is a slider to set the minimum word length, from 4 to 9. Then the user clicks the "Search for words" button and the VBA code begins to execute. The first thing is to check that the number of letters entered is 9. If not, the code stops with an error message. Otherwise, the "letter bag" of the nine chosen letters is then computed. This is a simple array of 26 integers, one corresponding to each letter of the alphabet. The frequency of each letter is compiled as the "letter bag". For example, the bag for the word "APPLE" is shown in Figure 1. It is easy to see that there is one "A", one "E", one "L", and finally two copies of "P". No other letters appear, so the frequencies corresponding to them are all zero.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0

Figure 1 Letter bag for the word "APPLE"

The rest of the VBA code consists of a loop which scans down the word of the dictionary just mentioned. For each dictionary word, the letter bag is computed, as outlined above. The final piece of logic concerns the detection of *sub-bags*. Suppose we have two word bags, say b1 and b2. Recall that these are each arrays of 26 non-negative integers. Then b1 is said to be a sub-bag of b2 if and only if each of the 26 frequencies of b1 does not exceed the corresponding frequency of b2. This is an obvious extension of the concept of subset, made necessary by the possible duplication of letters. Finally, if a sub-bag is detected, then the

current word being scanned is a solution (hit) and is duly output to the same Excel sheet where the user entered the word and the Scan button. Apart from counting the hits, that's pretty much all there is to it! There mechanics of detecting sub-bags are simple enough to implement and may be checked by the interested reader, as complete VBA code is supplied with the Letters & Numbers Excel model.

3.3 Solutions from book compared with Excel solutions

The matter of whether a "solution" to the word game is correct depends, of course, on the acceptability of any given word, with respect to a set of rules or a prescribed dictionary, with clearly-defined extensions from words appearing in it. According to one of the SBS Letters & Numbers books (2010, author unknown – hereafter referred to as the "turquoise book"), on the SBS TV program, the word had to appear in the dictionary, or be a clearly-defined extension of such a word. More precisely:

> *Words that are accepted as answers in Letters and Numbers are words that can be found in The Macquarie Dictionary. Accepted words are the headwords (that appear in bold at the beginning of listings) and all their inflected forms both regular and irregular. Some words have run-on headwords which we also accept, these appear in bold and with a hyphen at the end of the listing (e.g. -word). We also accept variant spellings if they appear in bold within the listing. Colloquial, slang, archaic or obsolete terms, foreign words, and specialised jargon words are allowed ONLY if they have a headword listing in the dictionary. We do not accept words that are hyphenated, capitalised (such as proper nouns), words that require an apostrophe or words that only appear in combination. Nor do we accept abbreviations or acronyms unless they have made it into common language and therefore have a headword listing.*

Our model ignores these rules.

4 Numbers Game

The following description is abridged from (Sugden & Stocks, 2013).

In this part of the game, one contestant is invited to choose a "combination" of six numbers. In normal mathematical parlance, the term "combination" refers to a *subset*; however, in this game, it does not refer to a set at all, but rather to two set cardinalities. Even this is not quite true as the collection of numbers ultimately chosen may contain duplicates. For example, in response to the invitation to choose a combination, a contestant may typically say "two large and four small numbers". These are references to two underlying sets of possible small and large numbers, respectively S = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} and L = {25, 50, 75, 100}.

In fact it is not the contestant but "Numbers Queen", Lily Serna, who chooses the numbers by selecting the specified quantities of large and small numbers from two separate collections of face-down cards. Duplicates are possible, so selection may be regarded as being with replacement. For the choice "two large and four small numbers", an example of

the final "set" of chosen numbers might be {2, 3, 2, 7, 50, 75}. Strictly, this is not a set but a *bag* (as in the *Letters* component of the game), since duplicates may be present. Extreme choices such as all large or all small are permitted but rarely chosen, as there seems to be a perception that such extremes may make the next part (generating the target number) more difficult. Frequent "combination" choices are three each of large and small, the so-called "family mix" of two large and four small, and less commonly, the "rat-pack" of six small numbers.

Once the six numbers (hereafter called *operands*), with possible duplicates, are selected, Lily then presses a button, which presumably directs a computer to generate a pseudo-random three-decimal-digit number (hereafter called the *target*) from 101 to 999 inclusive. Contestants then are given 30 seconds to find a way of combining any or all of the six operands, with the usual arithmetic operations of addition, subtraction, multiplication and division, and parentheses where necessary, with the aim of generating the target. There is no special benefit to players in using more or fewer operands; rather the aim is to generate the target exactly, or within 10 either side. It is not explicitly stated, but nevertheless clear that no intermediate result may be non-integer, or negative.

4.1 How many trees are possible?

For several reasons, it is of interest to know something about the "size" of the numbers problem. Given the six operands, how many possible arithmetic expression trees (see section 5.1) may be constructed? We do not go into detail here, merely presenting a summary. If T_n is the number of trees (expressions) which may be constructed by choosing n of the nodes from 6, then , ignoring equivalences and mathematical impossibilities (like 6/(3-3)), we have $T_1 = 6$, and

$$T_n = \frac{8(2n-3)(7-n)}{n} T_{n-1} \qquad \text{if} \quad 2 \le n \le 6 \qquad (1)$$

This recurrence leads to Table **1**. We refer the reader requiring further detail to (Sugden & Stocks, 2013), and to exercise 9 here, where an outline of its derivation is given.

5 Implementations

The benefits of Excel for supporting mathematical modelling are well known, but few would claim that one of these is speed of calculation. Thus, we created an alternative implementation in Delphi (whose underlying language is (Object Pascal), and which has a general reputation for run-time efficiency. The Excel/VBA model stores expressions as string values in arrays. It avoids the generation of commutatively equivalent expressions. It may take up to about two minutes or even more to solve the Numbers Game on a modern architecture, although many examples from the *Letters & Numbers* program are solved in less than 20 seconds, and often just a few seconds.

Table 1 Number of expression trees with n operand nodes

n	T_n
1	6
2	120
3	3,840
4	115,200
5	2,580,480
6	30,965,760

As noted, a Delphi implementation was created for comparison purposes as the Excel model seemed to be rather slow on some of the harder Numbers Game problems. Our Delphi solution runs about 100 times as fast as the VBA/Excel model, generating and checking for feasibility all possible (roughly 2 or 3 million, depending on how many equivalents are eliminated) candidate trees/expressions in about 2 seconds on a modern architecture running Windows 7. For further details of alternative implementations, see (Sugden & Stocks, 2013).

6 Teaching Applications

The rich mathematical and computational structure of the Numbers Game make it amenable to use as an example in a wide variety of teaching applications for Discrete Mathematics, Programming, Data Structures, Theory of Programming Languages and Complexity Analysis. Here, we focus on the important topic of arithmetic expression trees, with a brief mention of counting, recurrences, data structures and algorithms. Discussion of the wider range of topics is beyond our scope here, but more details may be found in (Sugden & Stocks, 2013).

6.1 Arithmetic expression trees

A topic we believe is accessible to first year students with only modest background in algebra is that of *arithmetic expression trees* (AETs). Such material might also be suitable for the senior years of high-school, given an appropriate curriculum, but is unfortunately and notably absent from the new "national curriculum" in both mathematics and "Technologies" in Australia, where the authors reside and work. It is even being forced out of tertiary offerings, to make way for more "business-oriented" units. However, all of this need not be totally bad news, as technology such as Excel and other tools may be used to help illustrate and make mathematical and Computer Science fundamentals more accessible to math-deficient students.

One of the many applications of binary trees is to arithmetic expressions. In this, they beautifully illustrate the recursive structure of arithmetic expressions, and also aid in their evaluation. When represented in such a form, ambiguity disappears, and a variety of

seemingly abstract concepts can be made clear. One example: the binary tree canonical form may be flattened into a linear form (prefix, infix or postfix string) using various recursive traversal algorithms. A simple, stack-based algorithm can be used for evaluation of postfix expressions, where parentheses are not necessary to preserve meaning. A conventional infix expression, discussed below, becomes ambiguous when more than one binary operator is used. Only by convention does multiplication have precedence over addition, and parentheses are used to force order of evaluation, if necessary. We discuss this further below.

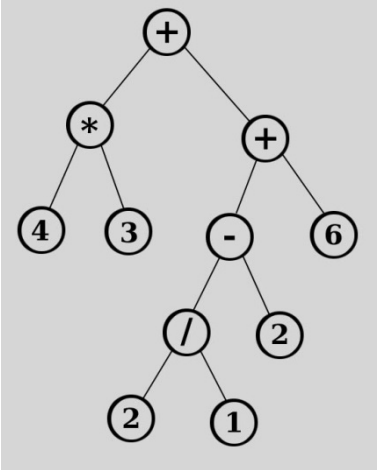


Figure 2 An arithmetic expression tree with six operands

The tree of Figure 2 is an arithmetic expression tree with six operand nodes (often called "leaves" as they have no descendants) and five interior nodes. It is a binary tree, in which all interior nodes are operators and all leaves are operands. Evaluation of the expression represented by the tree may be thought of as "flattening" the tree into a linear expression, with parentheses inserted where ambiguity might otherwise result. This process is usually called in-order traversal, and proceeds recursively as left sub-tree, root-node, right sub-tree. In the present case, we obtain (4*3) + (((2/1) - 2) + 6).

Of course, centuries of mathematical tradition dictate that expressions are evaluated from left to right and that multiplication and division are performed before addition and subtraction. So, we could remove some sets of parentheses without altering the intended meaning (value) of the expression. In fact, given the above conventions, *every* set of parentheses may be removed in this example without damaging its meaning. To see this put the original expression (4*3) + (((2/1) - 2) + 6) into Excel and then remove the parentheses, yielding 4*3 + 2/1 - 2 + 6, and then put this also into Excel. Of course, you will need to prefix each with an equals sign. The first thing to note is that Excel does not complain about any poorly-formed expressions. Secondly, it evaluates both as 18.

We have remarked that parentheses can sometimes be removed without altering the meaning of an expression. However, this is not true, in general. To see this, consider the tree

of Figure 3. A flattened version, obtained by in-order traversal and inserting parentheses, gives (2 + 3) * (4 + 5), which evaluates to 45. However, removal of parentheses produces 2 + 3*4 + 5 and introduces ambiguity, customarily resolved by invoking the "centuries of tradition" rules mentioned above. This makes the expression effectively 2 + (3*4) + 5, which evaluates to 19.

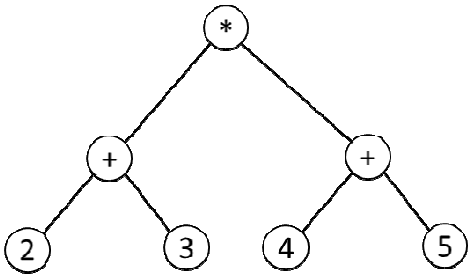


Figure 3 AE tree requiring parentheses to preserve meaning when flattened

We have included some exercises on arithmetic expressions trees for students in section 8. These make a useful introduction before embarking on spreadsheet modelling on the Numbers Game. It is recommended that the teacher cover the topic of arithmetic expressions and their associated binary trees before detailed consideration of the Numbers Game.

6.2 Counting, recurrences, data structures and algorithms

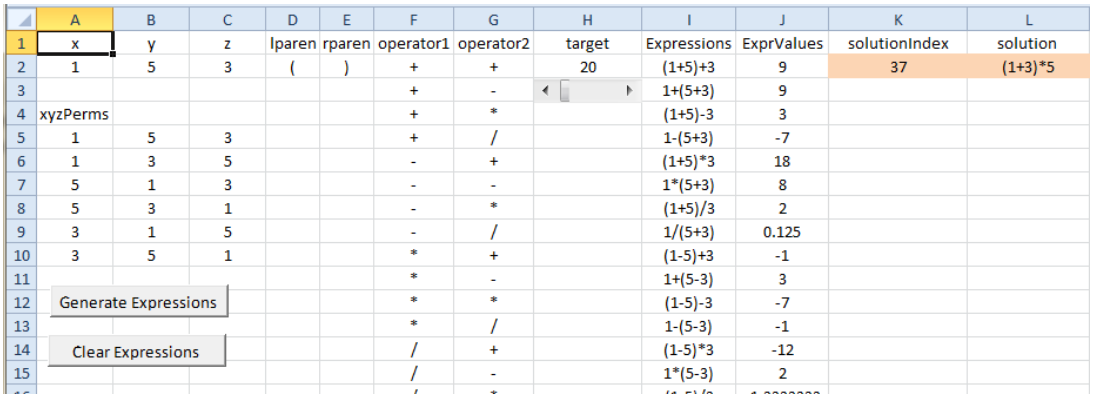
Determining the size of the Numbers Game problem for a given number of operands is a non-trivial exercise in counting, as discussed in Section 3.1. It provides a practical example of using factorials in counting, and also introduces Catalan numbers. The Numbers Game is a simple application requiring binary expression trees and evaluating such an expression tree is an example of post-order traversal, commonly using a stack for postfix expression evaluation. Again, the Numbers Game is a good example on which to apply this algorithm. Further detail is available in (Sugden & Stocks, 2013).

7 Experience using Letters and Numbers as a student assignment

In Bond University's Bachelor of IT, there is just one mathematics unit, known as Analytical Toolkit. It consists of a typical set of introductory discrete mathematics topics, ending with a few weeks of very basic introduction to probability and statistics. In the January semester of 2012, it was decided to set an assignment for the students based on the *Letters & Numbers* game. Since most of the class were first semester students, and, from past experience, their mathematical backgrounds are typically very poor, this exercise had to be carefully planned. Further, most students in this class do not have any programming experience. How could we present sufficient background material for the students firstly to understand the problem, and secondly to implement some kind of a solution? It was decided that the idea could only become feasible if we:

1. reduced the scope of the problem by relaxing the numbers game to 3 operands instead of 6,
2. cast the problem into an environment where at least some of the solution logic may be expressed without having to write code, and
3. supplied the class with some skeleton code which outlines an overall solution strategy.

These changes were achieved by putting a reduced version of the problem into Microsoft Excel 2010 with VBA code and Excel formulae and tables. For the Letters part of the problem the class was supplied with public domain word lists, again in Excel. A session on elementary VBA was also taught. Although the class was small and the number of responses even smaller, feedback was uniformly positive; again, more detail is in (Sugden & Stocks, 2013).



	A	B	C	D	E	F	G	H	I	J	K	L
1	x	y	z	lparen	rparen	operator1	operator2	target	Expressions	ExprValues	solutionIndex	solution
2	1	5	3	(	)	+	+	20	(1+5)+3	9	37	(1+3)*5
3						+	-		1+(5+3)	9		
4	xyzPerms					+	*		(1+5)-3	3		
5	1	5	3			+	/		1-(5+3)	-7		
6	1	3	5			-	+		(1+5)*3	18		
7	5	1	3			-	-		1*(5+3)	8		
8	5	3	1			-	*		(1+5)/3	2		
9	3	1	5			-	/		1/(5+3)	0.125		
10	3	5	1			*	+		(1-5)+3	-1		
11						*	-		1+(5-3)	3		
12						*	*		(1-5)-3	-7		
13						*	/		1-(5-3)	-1		
14						/	+		(1-5)*3	-12		
15						/	-		1*(5-3)	2		

Figure 4 Excel model of Numbers Game for 3 operands

8 The Letters & Numbers Book – some results and comments after solving the puzzles in Excel

The Excel model was used to solve all of the Letters & Numbers puzzles in the "turquoise" Letter & numbers book, with interesting results. This volume, like several others of the same form but having differently coloured jackets, contains 50 "rounds", each consisting of 5 letter games and 3 number games, plus 2 word mixes and the conundrum. As with the TV show, the puzzles are somewhat biased toward the letters game, each having 250 word games and 150 numbers games. It was very interesting to see how the Excel model performed with each of these puzzles. We give a summary here and also supply full details online in the accompanying Excel files.

An interesting aspect of the puzzles from the turquoise book is that not all Numbers games had solutions. There were 150 Numbers games, of which four were implied by the "turquoise book" to have no solution. These cases are summarized in Table 2. Noteworthy is the fact that the Excel & Delphi models were able to find closer "solutions" than those found in the "turquoise book" in two cases. Of particular note is the case of Round 23, Game 6, for

which the book claims "no solution", whereas the present model easily finds a solution. For the 146 cases where the book has solutions, 59% were solved in 10 seconds or less and 78% in 30 seconds or less by the Excel model. The search for solutions proceeds by increasing the number of operands used. Thus, the ones taking more time will requires a greater number of operands, or, for the cases of 5 and 6 operands, occur later in the tree generation process.

The longest solution time in Excel was 326 seconds for Round 40, Game 6. Even for this one, the Delphi program finds the same solution in about 2 seconds, after constructing and checking some 2.1 million expression trees. Since the Delphi program was a port of the VBA/Excel version without using any special features of Delphi's Object Pascal, it seems that the Delphi array and string handling is drastically more efficient than that of VBA.

Table 2 Four "impossible" cases from the turquoise book.

Round	Game	a	b	c	d	e	f	Target	Nearest solution in book	Nearest solution (Excel)
13	8	4	1	5	25	50	100	533	50 + (25 + 4) + 1 = 530	4 + (25 + 5*(1 + 100)) = 534
17	8	10	50	100	25	75	3	883	(75*10) + 100 + 25 + 10 = 885	3*(100*75 - 50)/25 - 10 = 884
23	6	4	4	2	5	50	25	873	"No solution"	5*((4*50) - 25) – 2 = 873 (exact)
40	8	8	2	6	2	75	25	687	"No solution"	25 + 8*(6 + (2 + 75)) - 2 = 687 (exact)

9 Some suggested classroom exercises

1. Compute word bags for the following strings, and create a frequency table in Excel for each.
    a. INVESTIGATIONS
    b. YABBADABBADOO
    c. INDIVISIBILITIES
    d. CONTRAINDICATION
    e. DEOXYRIBONUCLEIC
    f. ANTIDISESTABLISHMENTARIANISM
2. Construct an Excel formula or model to compute a word bag from a given word in a cell. Hint: use the MID and COUNTIF functions. You could make an array formula of write a VBA function; alternatively, just write a single formula for letter "A", then fill down to "Z".
3. Draw expression trees for the following arithmetic expressions.
    a. 25 – (3*4)
    b. (3 + 4)*5 + 100
    c. (75 - 3)*(50 + 1)
    d. 1 + 2*(75 - (10 + (2*3)))
    e. 9 – ((6 – (5 – (3 – 2)))*4)
4. Solve the following Numbers Games on paper, that is, produce at least one solution.
    a. Numbers 5, 7, 9, 2, 100, 75; Target 55. (p120 of turquoise book [9])
    b. Numbers 10, 8, 2, 7, 75, 25; Target 604. (p40 of turquoise book [9])
    c. Numbers 2, 5, 7, 25, 50, 75; Target 886. (simplest solution uses 5 operands)

5. Solve the problems of the previous question using the Letters & Numbers Excel model, and the Delphi model *JustNumbers*.
6. Write down some heuristics to help solve the Numbers Game. For example, it may be possible to get close to the target by using the large numbers first, perhaps with small adjustments by making use of the smaller numbers. Lily sometimes mentions these on the TV program.
7. The Excel model can be quite slow, especially when a few million separate cases are checked, only to find that all are fruitless – there is no solution. Therefore, use the fast Delphi Letters & Numbers model to find cases which cannot be solved. Start with a reasonable set of 6 numbers and try different targets. See if you can spot a pattern for targets which are impossible for a given set of 6 inputs (operands).
8. Following on from the previous question, can you come up with a way (i.e., a rule) to help decide if a given Numbers problem has no solution? For example, 1, 1, 1, 1, 1, 1; 999 cannot be solved, but this is the most extreme case.
9. Derive the recurrence of equation 1. Hint: T[n] is the product of three separate quantities: (i) the number of binary trees with *n-1* nodes (Catalan number), (ii) number of operator choices, 4^(n-1), and (iii) the number of permutations of 6 objects (operands) chosen from n. Once you have this product, take the quotient T[n]/T[n-1] and simplify to obtain equation 1. Details may be found in (Sugden & Stocks, 2013).

10 Conclusion

The TV program Letters & Numbers provides a rich source of examples illustrating a wide variety of mathematical and computing principles. Although working with a very small class, we have found that the game captures the interest of the students, and allows an unusual approach to some of these topics, in some cases at least appearing to be "inverses" of standard problems. For example, in the Letters Game, we are not using an electronic dictionary as the basis of a spelling checker, but finding all words in the dictionary whose spelling is an anagram of a given "word". In the Numbers Game, instead of evaluating an arithmetic expression, we need to generate one with a given value. A linear array traversal algorithm is illustrated by the Letters Game, while principles of counting and enumeration, along with recursive structures and relations are well-illustrated by the Numbers Game. Our models are provided along with this paper and we encourage teachers to use the models as a basis for an interesting lesson plan for classes in discrete mathematics, data structures and theory of programming languages. The game, especially the Numbers Game, has great scope for investigation of many further topics. The Delphi program *JustNumbers* is also provided for a very fast version of the Numbers Game. You could use it while watching the program, or to check solutions from one of the many SBS Letters & Numbers books. Happy solving!

11 Acknowledgment

The authors wish to thank the three referees for very useful comments on the originally submitted version of this paper.

12 Bibliography

_____. (2010). *Letters & Numbers Word and number puzzles from the SBS TV show, Book 2 ("the turquoise book").* Prahran, Victoria: Hardie Grant books.

Bellotti, F., Bottio, R. M., & Nadolski, R. (2013). Special Issue on “Game based learning for 21st century transferable skills: challenges and opportunities". *Journal of Educational Technology & Society (in press).*

Knuth, D. E. (1997). *The Art of Computer Programming Volume 3: Sorting and Searching (2nd Ed).* Addison-Wesley.

Sugden, S. J. (2001). Bits, binary, binomials and recursion: Helping IT students understand mathematical induction. *Quaestiones Mathematicae, Journal of The South African Mathematical Society Supp. #1*, 133-140.

Sugden, S. J. (2009). *Problem Solving with Delphi.* Hauppage, NY: Nova Science Publishers.

Sugden, S. J., & Stocks, P. A. (2013). Letters & Numbers: A Vehicle to Illustrate Mathematical & Computing Fundamentals. *Proceedings of Lighthouse Delta 2013, November 2013.* Kiama, NSW, Australia.

13 Appendix

Here, we give some general advice on how to use the Excel and Delphi models. As mentioned in the paper, the Delphi model was created as a comparison to the rather slow Excel/VBA version. Both models will only find exact solutions to the *Numbers* component of the game. That is, no attempt is made in either model to find approximate solutions, such as is allowed in the television program. In the SBS program, contestant still may score points in the Numbers game by generating a target within 10 of the desired target.

How to use the Delphi model JustNumbers

The interface is pretty much self-explanatory. The user simply fills out the edit boxes with the six operands and the target. There is a checkbox option to stop after the first solution is found and it is generally recommended that this be checked. The main reason for this is that some problems have many solutions and we are not usually interested in all of these, especially as the program logic is such as to generate “simpler solutions” (those with fewer operands) first.

How to use the Excel model L&N

There are several sheets in the Excel model which may be ignored by the user. These are those labelled 1, 2, 3, 4, 5 and 6. They exist only to store temporary possible expressions for the Numbers game and their names refer to the number of operands used.

There are several sheets that may be ignored as these are used for storing intermediate structures and dictionaries. In short, you may ignore the sheets labelled "1", "2", "3", "4", "5", "6", "lexicon" and "letters from turq book". Stated another way, the only sheets you need to be concerned with are those labels "Letters" and "Numbers".

Letters game

Click on the sheet labelled "Letters". In cell A2, type exactly 9 lower-case letters, without spaces; e.g., "oatgpeasc". There is a slider which allows for minimum length of words to be found, and you can vary that to take any value from 4 to 9 inclusive. Then click on the "Search for Words" button. After a short delay the words found will appear in columns C through H, according to their length. That is pretty much all there is to running the Letters component of the Excel model. For the chosen letters "oatgpeasc", the model should return "scapegoat" as the only 9-letter word, plus 293 other, shorter words.

Numbers game

Click on the sheet labelled "Numbers". In each cell of the range C2:C7, place one of the six operands (numbers); for example 4, 1, 5, 25, 50, 100. Then, in cell D2 place the target, say 226. Then click on the button "Search for solutions". Progress information in the form of "# Cases checked" will appear and be updated in cell A2. If a solution is found, it will be displayed in cell B2 along with its *reverse Polish* form in A3. This form uses the operand names "a" through "f" to represent the six operands, and these can be seen in the range E2:E7. That's pretty much it, but one should be prepared for slow search times if the problem is one of the more difficult ones. If the problem has no solution at all, then the Excel/VBA model will crunch away for a few minutes before stopping with the message "No solution found".