TACTILE TOOLS FOR TEACHING: AN IMPLEMENTATION OF KNUTH'S ALGORITHM FOR MASTERING MASTERMIND

THOMAS M. FIORE, ALEXANDER LANG, AND ANTONELLA PERUCCA

Abstract. We present a tactile tool that implements Donald Knuth's algorithm for winning the game of Mastermind in 5 or less moves. This tool, called the *Mastermind Rad*, can be easily printed using a color printer and assembled with a fastener. University teachers of general education math classes can use this tool to teach subtleties of the important notion of algorithm. This hands-on approach to algorithms may also be useful to high school teachers. In the article, we explain Knuth's algorithm and provide a few activities for students to explore it with the *Mastermind Rad*.

1. Introduction

NOTE: SEE THE STUDENT WORKSHEETS AFTER THE BIBLIOGRAPHY, and SEE SEPARATE FILE WITH RAD

This paper continues the long tradition of mathematical exploration via games and puzzles. We provide a printable tool for teachers called the *Mastermind Rad*[1] that implements Knuth's algorithm for winning the game of Mastermind in 5 or less moves. Some recent examples of work in the expansive and entertaining genre of playful mathematical exploration are [3, 4, 7, 9, 10, 11, 26, 32, 35, 37, 44, 49]. And of course, the *oeuvre* of Martin Gardner, including [18] and many books, introduced millions to the joys of mathematical puzzles and games.

Mordecai Meirowitz's 1970s hit Mastermind is a two-player game: *codemaker* against *codebreaker*. The codemaker first chooses a concealed sequence of four colors, called a *code*, and the codebreaker then attempts to determine the code. The codebreaker presents a test code, and the codemaker responds with the number of correct colors in the correct places and the number of correct colors in the wrong places. This process continues until the codebreaker has enough information to determine the code, or until the maximum number of attempts is exhausted. Today, Mastermind can also be played against the computer at websites such as [1] or on a smartphone.

In 1976, the famed computer scientist Donald Knuth presented a codebreaker algorithm [31] in the now defunct *Journal of Recreational Mathematics*. Surprisingly, the codebreaker (with the help of logic) can always find the concealed code after four tries and responses. The slogan and key idea behind Knuth's algorithm is that each codebreaker move should "*minimize the maximum number of remaining possibilities*." In Section 3 we explain this slogan, see in particular Table 1. Much has been written about Mastermind since Knuth's algorithm. See for instance [8, 12, 13, 15, 16, 14, 17,

[1] *Rad* means wheel in German.

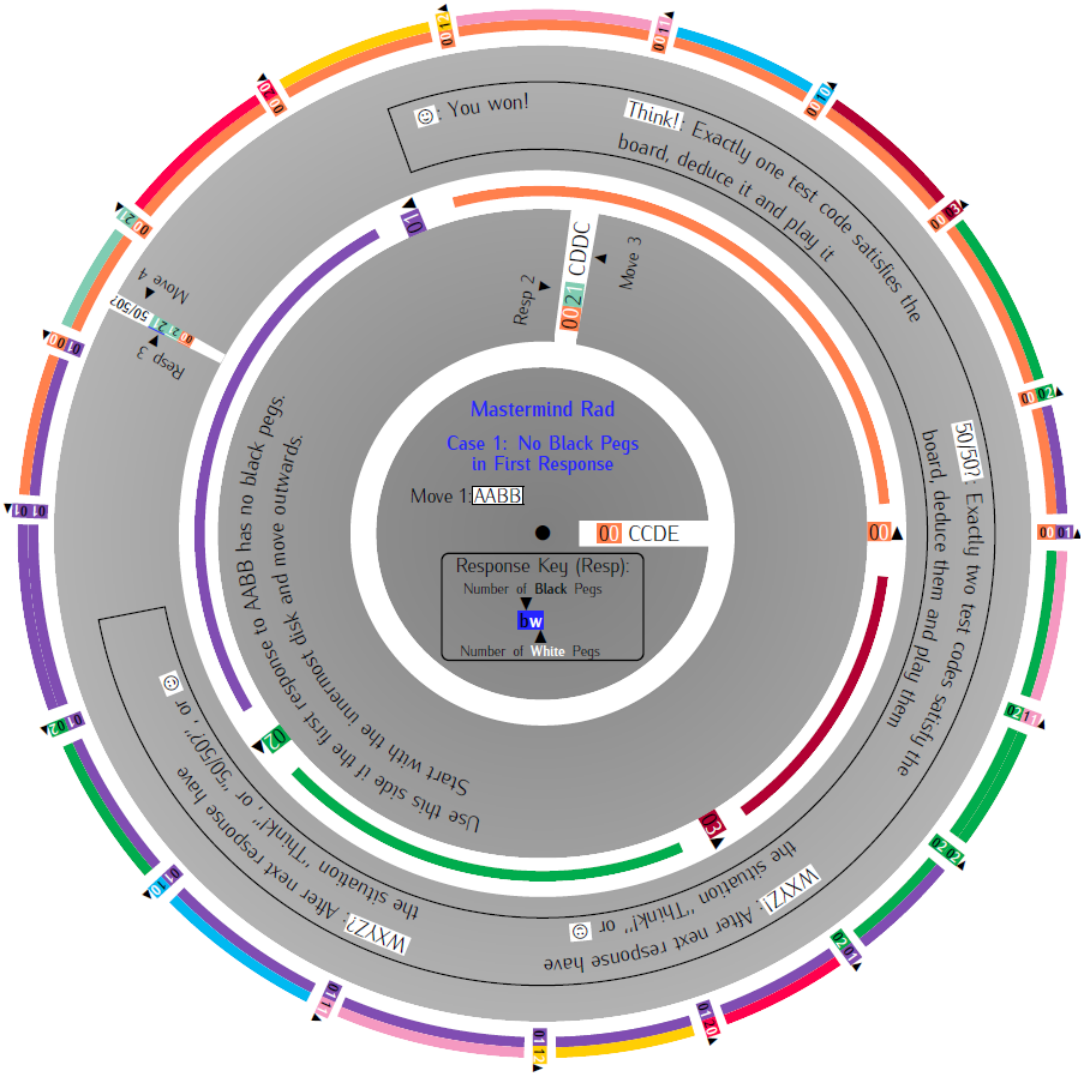

FIGURE 1. Side 1 of the *Mastermind Rad*. Print your own *Mastermind Rad* using the supplementary pdf file!

19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 33, 34, 38, 39, 40, 42, 41, 43, 45, 46, 47, 48] and Chapter 32 of the book [6].

Alexander Lang's enumeration implementation of Knuth's algorithm, called the *Mastermind Rad*, consists of two sets of four printable concentric rings that display the next move for the codebreaker. In addition to illustrating the idea of an algorithm in a tactile way, Lang's tool offers several other lessons for students: an algorithm can have multiple kinds of output, the user must sometimes consult logic and the outputs of previous steps to construct an outcome, and finally, an algorithm can involve guesswork. The *Mastermind Rad* helps students learn how to play the game and builds deductive reasoning skills. See Figure 1 for a picture of the *Mastermind Rad*.

2. Description of the Game Mastermind and Codebreaker Win Situations

In the game of Mastermind, the codemaker first selects and conceals a *code*, which is a sequence of four colors from the six colors[2] A, B, C, D, E, F. Repeat colors are allowed, but the positions matter, so there are $6 \times 6 \times 6 \times 6 = 1296$ possible codes. The codebreaker's objective is to determine the concealed code. The codebreaker presents a four-color test code, such as

AABB,

and then the codemaker responds about the correctness of the guess code with feedback in the form of 0 to 4 pegs: one black peg for each correct color in the correct position, and one white peg for each correct color in an *incorrect* position. Correct colors in the incorrect position should not be double counted. For instance, if the code is ABBB and the codebreaker tests CAAA, then the response is 0 black pegs and 1 white peg, not 3 white pegs. We record the response of b black pegs and w white pegs as

bw.

The 14 possible responses are listed in the first column of Table 1. Notice that response 31 is physically impossible, so is not included in this list. Notice also that some responses are not possible for specific moves, for instance, AAAA cannot have response 01. A noteworthy observation is that a total of $4 = b + w$ response pegs means the test code is correct up to reordering. In fact the sum $b + w$ is the maximum number of black pegs that can be obtained by reordering the test code.

After the codemaker response, the codebreaker presents an improved test code (improved using the ambiguous information from the response). The codemaker responds again with 0 to 4 black/white response pegs, and the process repeats. All of the previous attempts and responses are present on the playing board, so the codebreaker can use all of the previous information to deduce the code, not only the present response. The game ends when the codemaker responds with 4 black pegs (meaning the correct code has been presented by the codebreaker), or when the codebreaker has used up all the allowed attempts.[3]

Illustrated instructions on how to play Mastermind are found at [2]. There are other variants of Mastermind with codewords longer than 4 and with more than 6 colors. There is also the simplification that does not allow repeat colors in the codeword. Further variants appear in the literature list of Section 1.

Sections 2.1 and 2.2 contain detailed examples of winning and pre-winning situations. The logic is somewhat intricate, so we also incorporate some of these examples into the Activity Handouts for partner work in the appendices. The examples below serve two purposes: illustrate features of the algorithm for the reader, and present solutions to the partner work in the appendices.

[2]Different versions of the game have different colors, so we generally denote colors by letters. Knuth used numbers instead of letters in his 1976 paper [31].

[3]The original game board allowed 12 attempts.

2.1. **Winning Situations for the Codebreaker.** We next give two situations in which the codebreaker can make a move and win, these are called "Think!" and "50/50?". We also present the situation in which the codebreaker immediately wins, called ☺. These situations foreshadow Knuth's algorithm and the notation of the *Mastermind Rad*.

Example 2.1 (Think!)**.** In this situation, all the information necessary to deduce the code is already on the playing board, and the codebreaker must only think and then present the correct code. The *Mastermind Rad* does not find the code. For example, suppose the first codebreaker move and first codemaker response are as follows.

	Move	Response
1.	AABB	04

The codebreaker understands the response that all four colors are correct and in the wrong place. This is all the necessary information to deduce the code because of the particular configuration AABB. So the codebreaker's next test code is BBAA, and the codemaker responds with 4 black pegs, and the game is over.

Example 2.2 (☺)**.** In this situation, there is not enough information on the playing board to deduce the code, but the codebreaker presents a test code and by luck guesses the correct code. For example, in the following situation, after the first response, the codebreaker only knows that the colors A and B do not appear in the code, so presents a test code with other colors, namely CCDE and wins by a combination of luck and logic.

	Move	Response
1.	AABB	00
2.	CCDE	40

Example 2.3 (50/50?)**.** In this situation, exactly two test codes satisfy the information on the playing board, and one of them is the correct code. The codebreaker must simply try the two test codes, and has a 50/50 chance of getting it right on the next try. For instance, consider a modification of the situation of Example 2.2, with a second response of 04 instead of 40.

(2.1)

	Move	Response
1.	AABB	00
2.	CCDE	04

In the second move CCDE, all four colors are correct but in the wrong position. So in the correct code, the colors CC must be in the last two positions, and the colors D and E must be in the first two positions, ordered either as DE or ED. As a third move, the codebreaker should now do DECC or EDCC. If the third response is 40, the codebreaker wins. If the third response is not 40, then the codebreaker should next play the other respective option of DECC or EDCC, and the fourth response will be 40 and the codebreaker wins. See Figure 2 for a record of these two possibilities in the case that DECC is the concealed code.

Concealed Code DECC		
	Move	Response
1.	AABB	00
2.	CCDE	04
3.	DECC	40

Concealed Code DECC		
	Move	Response
1.	AABB	00
2.	CCDE	04
3.	EDCC	22
4.	DECC	40

FIGURE 2. The only two logically consistent continuations of the game (2.1) when the concealed code is DECC. If the concealed code were instead EDCC, there would similarly be only two possible logically consistent continuations.

2.2. **Pre-Winning Situations for the Codebreaker.** For the purposes of Knuth's algorithm and Lang's implementation, it is also useful to notate the situations that immediately precede "Think!" and "50/50?". That is, in the moment after a codebreaker move but before the codemaker response.

Example 2.4 ("WXYZ!" Means its Response gives "Think!" or ☺)**.** In this situation, the codebreaker has played a test code WXYZ and is awaiting the response, and *any* response will bring us into the situation "Think!" of Example 2.1 or the situation ☺ of Example 2.2. In other words, any possible response to WXYZ will give enough information to deduce the code (of course different codes for different responses to WXYZ), or the correct code is played by chance. For instance, consider another modification of the situation of Example 2.2, with a second response of 13 instead of 40.

	Move	Response
1.	AABB	00
2.	CCDE	13
3.	CDEC!	

By the way, the exclamation point in the test code "CDEC!" is meant to remind us of the exclamation point in "Think!".

We claim that any third response (along with the previous information) uniquely determines the concealed code. First note that we know the colors are C, C, D, E because the second response to CCDE is 13. Thus, the third response (to the reordering CDEC of the correct colors) must have a sum of 4: the only possible third responses are 04, 13, 22, or 40 (the response 31 is always physically impossible so we do not consider it). Another consequence of the second response to CCDE as 13 is that exactly one of these two C's is in the correct position.[4] Next we go through the possible third responses and show that each one (along with the previous information) determines a unique code, thus confirming that each possible third response is the "Think!" situation or ☺ situation.

[4]By response 13 to CCDE, exactly one of the two C's in CCDE is in the correct position, by the following argument. If D were correct, then both C's would have to move to the fourth position, which is impossible. Similarly, if E were correct, then both C's would have to move to the third position, which is impossible. So one of the two C's in CCDE is in the correct position.

Suppose the third response to CDEC is 04. Then in CDEC, both the first and last C are in the wrong position, so the concealed code must be of the form □CC□. But from the second response, we know the final color cannot be E, so must be D. Finally, the concealed code is ECCD.

Suppose the third response to CDEC is 13. As mentioned above, we know from Response 2 that either the first or second entry of the correct code is C, so we consider the various cases.

(i) Suppose in Move 2 only second entry C is correct and suppose in Move 3 only third entry E is correct. Then the 13 response to Move 3 implies the code must be of the form [not C]CE[not C]. This is a contradiction, as the second C must be somewhere!
(ii) Suppose in Move 2 only second entry C is correct and suppose in Move 3 only fourth entry C is correct. Then the 13 response to both Moves 2 and 3 implies the code must be of the form □C[neither D nor E]C. This is a contradiction, since D and E cannot both be in the first entry.
(iii) Suppose in Move 2 only first entry C is correct, hence also in Move 3 only first entry C is correct. Consequently, the final three letters in Move 2 and the final three letters in Move 3 are all in the wrong position. So the concealed code must be CECD.

Suppose the third response to CDEC is 22. As mentioned above, we know from Response 2 that either the first or second entry of the correct code is C, so we consider the various cases.

(i) Suppose in Move 2 only first entry C is correct and in Move 3 additionally second entry D is correct. Then in Move 3 only the third and fourth positions are wrong and the code is CDCE. Contradiction: this final E cannot be correct according to Response 2. So the code is not of the form CD□□.
(ii) Suppose in Move 2 only first entry C is correct and in Move 3 additionally third entry E is correct. Then in Move 3 only the second and fourth positions are wrong and the code is CCED. Contradiction: this second C cannot be correct according to Response 2. So the code is not of the form C□E□.
(iii) Suppose in Move 2 only first entry C is correct and in Move 3 additionally fourth entry C is correct. Then in Move 3 only the second and third positions are wrong and the code is CEDC. Contradiction: this third position D cannot be correct according to Response 2. So the code is not of the form C□□C. Combining (i), (ii), and (iii), we see the code is not of the form C□□□.
(iv) Suppose in Move 2 only second entry C is correct. Then first entry C is incorrect, and in Move 3, first entry C and second entry D are both incorrect, so the final EC must be correct. Hence the code is DCEC.

Suppose the third response to CDEC is 40. Then the codebreaker has guessed the correct code by a combination of luck and logic, precisely the situation ☺.

Example 2.5 ("WXYZ?" Means its Response gives "Think!", or "50/50?", or ☺)**.** In this situation, the codebreaker has played a test code WXYZ and is awaiting the response, and *any* response will bring us into the "Think!" situation of Example 2.1, or

the "50/50?" situation of Example 2.3, or the ☺ situation of Example 2.2, and "50/50?" will occur for at least one of responses. In other words, any possible response to WXYZ and the other information on the playing board is satisfied by at most two codes, and for some responses exactly two test codes (the correct code could also be played by chance). For instance, consider another modification of the situation of Example 2.2, with a second response of 22 instead of 40.

	Move	Response
1.	AABB	00
2.	CCDE	22
3.	CDCE?	

By the way, the question mark in the test code "CDCE?" is meant to remind us of the question mark in "50/50?".

We claim that any third response (along with the previous information) uniquely determines the concealed code, or determines exactly two candidate codes, or is a win by luck and logic.

First note that we know the colors are C, C, D, E because the second response to CCDE is 22. Thus, the third response (to the reordering CDCE of the correct colors) must have a sum of 4: the only possible third responses are 04, 13, 22, or 40 (the response 31 is always physically impossible so we do not consider it). Next we go through the possible third responses and show that each one (along with the previous information) determines a unique code, or exactly two candidate codes, or is a lucky win.

Suppose the third response to CDCE is 04. Then in CDCE, both the first and third C are in the wrong positions, so the concealed code must be of the form □C□C. In Move 2 from response 22, exactly two of CCDE are correct, so third entry D must be correct (first entry C and last entry E are known to be wrong from Response 3). Hence the concealed code is uniquely determined to be ECDC.

Suppose the third response to CDCE is 13. Then we leave as an exercise to verify that there exactly two candidates: CCED and CEDC. The method is to explicitly consider all 6 cases of two correct entries in Move 2, and what that will mean in combination with 13 as Response 3.

Suppose the third response to CDCE is 22. Then we leave as an exercise to verify that the code is DCCE. The method is to explicitly consider all 6 cases of two correct entries in Move 2, and what that will mean in combination with 22 as Response 3.

Suppose the third response to CDCE is 40. Then the codebreaker has guessed the correct code by a combination of luck and logic, this is precisely the situation ☺.

3. Knuth's Algorithm

The key idea behind Knuth's algorithm can be understood by considering the easier game "Guess who?". There one has to determine a person among a set of 24 people, and Yes/No questions are allowed. A question like 'Does he/she wear glasses?' is bad if few people wear glasses because most likely the answer is 'No' and hence only few people get discarded. A good question is about a characteristic that half of the people

possess, so that with any answer we can discard half of the candidates. For a good strategy, the candidates should be at most uniformly distributed among the possible answers. This achieves that most candidates can be discarded independently of the response. In other words, the worst case scenario (maximum number of remaining possibilities by going through the responses) is under control.

The main idea of Knuth's algorithm is that the codebreaker always plays a test code that "*minimizes the maximum number of remaining possibilities*" given the previous information on the playing board, amongst the possibilities for the upcoming response [31, page 3]. Moreover, when there is more than one test code that minimizes the maximum number of remaining possibilities, the algorithm chooses (if possible) to play a test code that is compatible with the previous information on the playing board, i.e., a test code that can actually win. Among these, the smallest in lexicographic order is played.

As an illustration of the slogan "*minimizing the maximum number of remaining possibilities*", consider the first move of the game, when nothing is known. What should the codebreaker's first move be? Is it clever to use all different colors in the first move? We analyze all possible first moves in Table 1.

Table 1 shows the number of candidate codes that remain after an initial test code is played and a response is given, or in other words, how many codes satisfy a given response to a given initial test code. The column labels are various initial test codes.[5] Other initial test codes are of course possible, but we do not list them because their columns will be equal to an indicated one obtained by permuting positions and/or colors. The row labels in Table 1 are the various responses. As an example of interpreting an entry, if the codebreaker's initial test code is AAAA and the codemaker's response is 00, then all $5^4 = 625$ sequences of the five colors B to F could be the code and the set of candidate codes after the first move and response has 625 elements.

Each column maximum in Table 1 is indicated in bold. The minimum column maximum 256 is located in the AABB column. Thus, the first codebreaker move should be AABB according to Knuth's algorithm.

Of course, CCDD or EEFF *et cetera* would theoretically work just as well for the first test code but the algorithm further stipulates that if more than one test code minimizes the maximum number of remaining possibilities, *then the first one in lexicographic ordering is selected.* Hence AABB rather than CCDD or EEFF.

The conceptual picture behind the algorithm is that each response narrows down as much as possible (while independently of the response) the set of candidate test codes, until finally there is a unique candidate, exactly two candidates, or the correct code is presented by a combination of luck and logic. Initially, before the first response, the set of candidate codes consists of all $1296 = 6^4$ four-color sequences. After the first

[5]One representative from each permutation class appears in a column heading. More precisely, two four-letter sequences are considered equivalent if one can be transformed into the other via a permutation of the six letters *and* a permutation of the positions. Equivalent four-letter sequences will have equal column number entries, so we only list one representative from each equivalence class, in a normal form. The idea of code equivalence class just explained is valid only for the first move. For later moves a sharpened notion of equivalence is needed, see Sections 5.1.2 and 5.2 of [36].

Remaining Candidate Test Codes After First Move and Response					
	First Test Code				
Response	AAAA	AAAB	AABB	AABC	ABCD
00	**625**	256	256	81	16
01		308	256	**276**	152
02		61	96	222	**312**
03			16	44	136
04			1	2	9
10	500	**317**	**256**	182	108
11		156	208	230	252
12		27	36	84	132
13				4	8
20	150	123	114	105	96
21		24	32	40	48
22		3	4	5	6
30	20	20	20	20	20
40	1	1	1	1	1

TABLE 1. The initial test code AABB minimizes the maximum number of remaining candidate test codes, so is the first move by the codebreaker in Knuth's algorithm. This table can be found in several references, see for example Table 32.1 of Bewersdorff [6].

response the set becomes smaller because of the new information. By choosing the first move AABB the worst case scenario is the response 10 shown in Table 1, in which 256 candidate codes remain. So with any response to AABB we have already got ridden of 80% of the codes!

The second move is similarly determined by the slogan and such a table, however the compilation of the table also takes the first response into consideration. The slogan does not prevent Move 2 from being a test code that contradicts the information on the board. In other words, in Move 2 and beyond, the algorithm may suggest a test code that does not satisfy the previous information on the playing board, and hence cannot win. This is a subtlety of the algorithm: sometimes the remaining candidate codes do not minimize the maximum number of remaining possibilities and in this case the codebreaker should use a test code that for sure will not have 40 as a response. Such a code will not make us win in the present move, but can provide much better information and allow us to win in five moves. An example of such an occurrence is given in [31, p.3]. See Activity 11 for example of a "smart move" that gains additional information but cannot win.

Of course, a computer is actually needed to realize Knuth's algorithm; the sets of candidate codes for various responses are too large for a human codebreaker to determine and compare quickly. The algorithm does not give basic rules that a codebreaker

can simply apply by hand in a given situation. Thus, there is a need for a geometric visualization of Knuth's algorithm...

4. Instructions for Using the *Mastermind Rad*

Alexander Lang's *Mastermind Rad* is a tool to help the codebreaker determine the code selected and concealed by the codemaker, using Knuth's algorithm. In fact, the *Mastermind Rad* is a geometric visualization of Knuth's Figure 1 [31, pages 4 and 5]. His visualization leaves out only one bit of information from Knuth's figure, namely the number of remaining candidate codes after each move.

The codebreaker tells the *Mastermind Rad* the responses, and the *Mastermind Rad* tells the codebreaker which test code to play. To use the *Mastermind Rad*, the codebreaker positions each ring according to the codemaker responses, and reads from the center disc outwards to the outermost ring. Each codebreaker test code move corresponds to a ring, except the fifth. The *Mastermind Rad* terminates in one of the situations of Examples 2.1, 2.2, 2.3, 2.4, or 2.5, so that in total five[6] or fewer tries are made by the codebreaker to win the game.

An important, but perhaps counterintuitive point, is that the *Mastermind Rad* does *not* find the concealed code for the user. Rather, the *Mastermind Rad* leads, eventually, to three situations:

(i) "Think!" All the information necessary to deduce the code is already on the playing board, and the codebreaker must only think and then present the correct code. See Example 2.1.

(ii) ☺ There is not enough information on the playing board to deduce the code, but the codebreaker presents a test code and by luck and logic guesses the correct code. See Example 2.2.

(iii) "50/50?" Exactly two test codes satisfy the information on the playing board, and one of them is the correct code. The codebreaker must think and deduce the two candidates, and then simply try them one after the other. See Example 2.3.

To represent the large number of possible game sequences, the *Mastermind Rad* sometimes stops just before two of the preceding situations. Namely, the *Mastermind Rad* stops at the pre-winning situations "WXYZ!" and "WXYZ?" that occur just before "Think!" and "50/50?", as described in Examples 2.4 and 2.5.

Next we describe how the codebreaker uses the *Mastermind Rad*. Suppose the codemaker has selected and concealed an ordered sequence (*code*) of four colors, chosen from the colors A, B, C, D, E, F. Repeat colors are allowed in the code.

Step 1. The codebreaker plays AABB, and the codemaker gives the 1st response.

(a) The codebreaker finds the correct face of the wheel, depending on whether the response has *no black peg*, or *at least one black peg*.

(b) The codebreaker rotates the innermost ring until the window displays the number of black pegs and the number of white pegs in the codemaker's 1st response.

[6]The count of five includes also the final presentation of the correct code.

Step 2. The codebreaker plays the code displayed in the innermost window, and the codemaker gives the 2nd response.
   (a) The codebreaker rotates the 2nd ring until the window displays the codemaker's 1st and 2nd response. Notice that the sector of interest on the 2nd ring is also indicated by an arc with the same color as the codemaker's 1st response.

Step 3. The codebreaker plays the code displayed in the 2nd window, and the codemaker gives the 3rd response.
   (a) The codebreaker rotates the 3rd ring until the window displays the codemaker's 1st, 2nd, and 3rd response. Notice that the sector of interest on the 3rd ring is also indicated by an arc with two stripes with the same colors as the stripes corresponding to the codemaker's 1st and 2nd response.

Step 4. The codebreaker plays the code displayed in the 3rd window, and the codemaker gives the 4th response.

Step 5. The codebreaker successfully determines the code from the 4th response, without consulting the *Mastermind Rad*.

As mentioned above, the *Mastermind Rad* may finish before Step 5. In most cases, the *Mastermind Rad* ultimately leads to situation "Think!" or "50/50?", both of which are combinatorial riddles (because the codebreaker must use logic to find the only possible code or the only two remaining codes). Notice that in general one can produce Mastermind riddles in the same fashion as chess puzzles, where a configuration of chess pieces is given, and the task is to find checkmate in one or two moves, without playing a whole game to reach the given configuration.

5. Using the *Mastermind Rad* to Teach Algorithms in a Class

The *Mastermind Rad* illustrates the idea of an algorithm in a tactile way, and can be used in a general education college course, or for enrichment of high school students. Key lessons of the *Mastermind Rad* are: an algorithm can have multiple kinds of output, the user must sometimes consult logic and the outputs of previous steps to construct an outcome, and finally, an algorithm can involve guesswork.

We have prepared two handouts for partner work that illustrate the various outcomes, see the appendices of this paper. Each group of two students should receive a printout of Student A Activity Sheet and Student B Activity Sheet, and have pencil and paper and a *Mastermind Rad*. The students alternate the roles of codemaker and codebreaker and play several complete games on paper (the codemaker is told to use a specific code). The activities are fairly short and ordered according to difficulty, and are mostly special cases of Examples 2.1-2.5. These activities assume that the students are familiar with the game Mastermind. Students may of course use a Mastermind playing board instead of pencil and paper. It may be instructive for students to first construct the *Mastermind Rad* themselves, in which case they should be furnished with the construction instructions, printouts of the *Mastermind Rad* components, scissors, and a fastener. Approximately 20 minutes should be planned for students to build their own *Mastermind Rad*.

The correspondence of the game activities to the previous examples and situations is as follows. The students should of course not see this table, nor this paper!

Activity	Concealed Code	Leads to Situation
1	BBAA	Think! Example 2.1
2	CBAA	Think!
3	DECC	50/50? Example 2.3
4	EDCC	50/50? Example 2.3
5	ECCD	WXYZ! Example 2.4 Third Response 04
6	CEDC	WXYZ? Example 2.5 Third Response 13

Another activity is to verify several of the results in Table 1. In other words, given an initial test code WXYZ and its response, the students are to determine the number of codes that satisfy this information (but not the codes themselves). The solutions to the problems in both Student Activity Sheets are as follows. The final AAAB 01 is quite challenging for students.

AAAA 00: There are exactly 5 colors left, hence $5^4 = 625$.

AAAB 00: There are exactly 4 colors left, hence $4^4 = 256$.

AABB 00: There are exactly 4 colors left, hence $4^4 = 256$.

AABC 00: There are exactly 3 colors left, hence $3^4 = 81$.

ABCD 00: There are exactly 2 colors left, hence $2^4 = 16$.

AAAB 01: If A is at the wrong place it must be in the 4th place so in the places 1 to 3 we get colors not AB hence $4^3 = 64$. If B is at the wrong place then in the 4th place we have non-AB colors while B is in one of the places 1 to 3, the other 2 places have non-A colors (1B gives $4 \times (3 \times 4^2) = 192$, 2B gives $4 \times (3 \times 4) = 48$, 3B gives 4.) Adding up we get 64+192+48+4=308.

The Student Activity Sheets have a number of activities that illustrate Knuth's algorithm and searches in Mastermind, see Activities 8 through 14.

Additionally, the *Mastermind Rad* illustrates the notion of algorithm in several more ways, especially regarding the geometric arrangement of the cases. The initial case distinction of "no black peg" versus "at least one black peg" corresponds to the two sides. The color coded arcs enable the user to visualize the tree of cases and navigate through it with ease, radially from the center outward (when the user lifts a ring to reveal all the responses, the sector color matches the case colors). The recording of the previous responses in the movable windows shows that the cases depend only on the set of previous responses. Finally, the number of rings corresponds to how many moves are needed to win: the *four* rings illustrate that the algorithm wins Mastermind in *five* or less moves.

6. Conclusion

The *Mastermind Rad* and Student Activity Sheets provide a fun, concrete introduction to the notion of algorithm. We have mentioned several key lessons students can learn from this visualization of Knuth's algorithm. We conclude with several more thought-provoking questions that can be used in class for meta-reflection. Students may notice in the activities, that Knuth's algorithm does *not* provide a repertoire of

moves for the codebreaker, nor does it give general advice on how to play in typical situations.

To wrap-up, an instructor could challenge students to debate the following questions: what should a good algorithm for Mastermind do? Should it provide a repertoire of moves that a human could use, or is it sufficient that a computer somehow searches a solution space with intelligent brute force? Should we keep track of how the algorithm works? Is it important that the algorithm always produces the same output or can the strategy be random, for example making a random guess from the set of remaining candidate code sequences? Should the algorithm just win the game, or win the game as fast as possible?

What does "as fast as possible" even mean? There are different ways to compare the speed at which algorithms win the game. Knuth's algorithm guarantees a win after 5 or fewer codebreaker moves, but other algorithms may have a lower *expected number of moves*, even though sometimes they require more than 5 moves. This is the case for *genetic algorithms* approaches to Mastermind, see for instance [5] and [38].

Why are algorithms relevant to students in everyday life? An instructor could challenge students to debate the sometimes antagonistic roles of algorithms in a 21st century economy and democracy. For instance, some online businesses optimize profits with "dynamic pricing" algorithms, while consumers can find their best price with aggregating platforms, hence algorithms can serve both profit maximization and consumer interests. Stock markets are another illustrative domain: algorithms in *high frequency trading* are stiff competition for individual investors. Mass surveillance is perhaps a final topic to liven up the mathematics classroom: email, telephone, and internet surveillance algorithms lie in the balance of national security and civil liberties. Also noteworthy is the inherent risk of algorithms and their implementations: Knight Capital Group's improper implementation of a stock trading algorithm lost untold millions in less than an hour a few years ago! Algorithms are not just about fun and games!

Acknowledgements

The original idea of the Mastermind Rad came to Alexander Lang while writing his *Lehramt* thesis [36] at the Universität Regensburg, mentored by Antonella Perucca. This paper complements and elaborates on selected topics of the thesis. Thomas Fiore thanks them for bringing him into the project during his 2015-2016 sabbatical at the Universität Regensburg, while supported by the Humboldt Foundation.

NOTE: SEE THE STUDENT WORKSHEETS AFTER THE BIBLIOGRAPHY, and SEE SEPARATE FILE WITH RAD

References

[1] Web Games Online. http://www.web-games-online.com/mastermind/.
[2] WikiHow: How to Play Mastermind. http://www.wikihow.com/Play-Mastermind.
[3] Jeremy Alm, Michael Gramelspacher, and Theodore Rice. Rubik's on the torus. *American Mathematical Monthly*, 120(2):150–160, 2013.
[4] J. Beck. Tic-Tac-Toe. In *Contemporary combinatorics*, volume 10 of *Bolyai Society Mathematical Studies*, pages 93–137. János Bolyai Math. Soc., Budapest, 2002.

[5] Lotte Berghman, Dries Goossens, and Roel Leus. Efficient solutions for mastermind using genetic algorithms. *Computers and Operations Research*, 36(6):1880 – 1885, 2009.
[6] Jörg Bewersdorff. *Luck, logic, and white lies*. A K Peters, Ltd., Wellesley, MA, 2005. The mathematics of games, Translated from the 2001 German edition by David Kramer.
[7] Maureen T. Carroll and Steven T. Dougherty. Tic-tac-toe on a finite plane. *Mathematics Magazine*, 77(4):260–274, 2004.
[8] V. Chvátal. Mastermind. *Combinatorica*, 3(3-4):325–329, 1983.
[9] Ben Coleman and Kevin Hartshorn. Game, set, math. *Mathematics Magazine*, 85(2):83–96, 2012.
[10] Mark A. Conger and Jason Howald. A better way to deal the cards. *American Mathematical Monthly*, 117(8):686–700, 2010.
[11] Benjamin Lent Davis and Diane Maclagan. The card game SET. *The Mathematical Intelligencer*, 25(3):33–40, 2003.
[12] Maria Camilla de Gaetano. Iterative methods in a finite set containing a pseudodistance. *Matematiche (Catania)*, 40(1-2):45–61 (1988), 1985.
[13] Benjamin Doerr, Reto Spöhel, Henning Thomas, and Carola Winzen. Playing Mastermind with many colors. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 695–704. SIAM, Philadelphia, PA, 2012.
[14] M. M. Flood. Sequential search strategies with mastermind variants – Part 2. *Journal of Recreational Mathematics*, 20(3):168–181, 1988.
[15] Merill M. Flood. Mastermind strategy. *Journal of Recreational Mathematics*, 18(3):194–202, 1985-86.
[16] Merill M. Flood. Sequential search strategies with mastermind variants – Part 1. *Journal of Recreational Mathematics*, 20(2):105–126, 1988.
[17] Riccardo Focardi and Flaminia L. Luccio. Guessing bank PINs by winning a mastermind game. *Theory of Computing Systems*, 50(1):52–71, 2012.
[18] Martin Gardner. *Martin Gardner's mathematical games*. MAA Spectrum. Mathematical Association of America, Washington, DC, 2005. The entire collection of his Scientific American columns, With a booklet containing a biography of the author by Donald J. Albers and Peter L. Renz.
[19] Nina Gierasimczuk, Han L. J. van der Maas, and Maartje E. J. Raijmakers. An analytic tableaux model for deductive mastermind empirically tested with a massively used online learning system. *Journal of Logic, Language and Information*, 22(3):297–314, 2013.
[20] Wayne Goddard. Static mastermind. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 47:225–236, 2003.
[21] Wayne Goddard. Mastermind revisited. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 51:215–220, 2004.
[22] Michael T. Goodrich. On the algorithmic complexity of the Mastermind game with black-peg results. *Information Processing Letters*, 109(13):675–678, 2009.
[23] Michael T. Goodrich. Learning character strings via mastermind queries, with a case study involving mtDNA. *Institute of Electrical and Electronics Engineers. Transactions on Information Theory*, 58(11):6726–6736, 2012.
[24] Richard K. Guy. The strong law of small numbers. *The American Mathematical Monthly*, 95(8):697–712, 1988.
[25] Richard K. Guy. The strong law of small numbers. In Richard K. Guy and Robert E. Woodrow, editors, *The Lighter Side of Mathematics: Proceedings of the Eugène Strens Memorial Conference on Recreational Mathematics & Its History*, pages 265–280. Mathematical Association of America, Washington, DC, 1994.
[26] Arthur Holshouser, Ben Klein, and Harold Reiter. The commutative equihoop and the card game SET. *Pi Mu Epsilon Journal*, 14(3):175–190, 2015.
[27] R. W. Irving. Towards an optimum mastermind strategy. *Journal of Recreational Mathematics*, 11(2):81–87, 1978-79.

[28] Gerold Jäger and Marcin Peczarski. The number of pessimistic guesses in Generalized Black-peg Mastermind. *Information Processing Letters*, 111(19):933–940, 2011.
[29] Gerold Jäger and Marcin Peczarski. Playing several variants of Mastermind with constant-size memory is not harder than with unbounded memory. In *Combinatorial algorithms*, volume 8986 of *Lecture Notes in Computer Science*, pages 188–199. Springer, Cham, 2015.
[30] Gerold Jäger and Marcin Peczarski. The worst case number of questions in generalized AB game with and without white-peg answers. *Discrete Applied Mathematics*, 184:20–31, 2015.
[31] Donald E. Knuth. The computer as master mind. *Journal of Recreational Mathematics*, 9(1):1–6, 1976/77.
[32] David Kong and Peter D. Taylor. Skunk redux. *Mathematics Magazine*, 85(4):267–276, 2012.
[33] Barteld Kooi. Yet another Mastermind strategy. *International Computer Games Association Journal*, 28(1):13–20, 2005.
[34] K. Koyama and T. W. Lai. An optimal mastermind strategy. *Journal of Recreational Mathematics*, 25(4):251–256, 1993.
[35] Igor Kriz and Paul Siegel. Rubik's cube inspired puzzles demonstrate math's "simple groups". *Scientific American*, 299(1):84–89, 2008, http://www.scientificamerican.com/article/puzzles-simple-groups-at-play/.
[36] Alexander Lang. Mastermind. *Zulassungsarbeit zur Examensprüfung für das Lehramt an Realschulen in Bayern. Universität Regensburg*, 2016.
[37] John Lorch. Magic squares and sudoku. *American Mathematical Monthly*, 119(9):759–770, 2012.
[38] J. J. Merelo, Pedro Castillo, Antonio Mora, and Anna I. Esparcia-Alcázar. Improving evolutionary solutions to the game of MasterMind using an entropy-based scoring method. In *GECCO'13—Proceedings of the 2013 Genetic and Evolutionary Computation Conference*, pages 829–836. ACM, New York, 2013.
[39] M. Mitchell. *Mastermind mathematics: logic, strategies, and proofs*. Key Curriculum Press, Emeryville, CA.
[40] E. Neuwirth. Some strategies for Mastermind. *Zeitschrift für Operations Research*, 26(8):B257–B278, 1982.
[41] J. H. O'Geran, Henry P. Wynn, and Anatoly A. Zhiglyavsky. Mastermind as a test-bed for search algorithms. *Chance*, 6:31–37, 1993.
[42] J. H. O'Geran, Henry P. Wynn, and Anatoly A. Zhiglyavsky. Rényi-type randomization theory for search-length upper bounds. *Nova Journal of Mathematics, Game Theory and Algebra*, 4(2):147–160, 1996.
[43] K. R. Pearson. Reducing two person, zero sum games with underlying symmetry. *Journal of the Australian Mathematical Society Series A*, 33(2):152–161, 1982.
[44] Ira Rosenholtz and Keith Wolcott. Tic-tac-toe on topological surfaces: does O always win? *Journal of Recreational Mathematics*, 37(3):192–218, 2008.
[45] Amy Strom and Scott Barolo. Using the game of mastermind to teach, practice, and discuss scientific reasoning skills. *PLoS Biology*, 9(1), 2011.
[46] P. F. Swaszek. The mastermind novice. *Journal of Recreational Mathematics*, 30(3):193–198, 1999-2000.
[47] D. Viaud. Une formalisation du jeu de Mastermind. *RAIRO Recherche Opérationnelle*, 13(3):307–321, 1979.
[48] D. Viaud. Une stratégie générale pour jouer au Mastermind. *RAIRO Recherche Opérationnelle*, 21(1):87–100, 1987.
[49] Emmanuel Volte, Jacques Patarin, and Valérie Nachef. Zero knowledge with Rubik's cubes and non-abelian groups. In *Cryptology and network security*, volume 8257 of *Lecture Notes in Computer Science*, pages 74–91. Springer, Cham, 2013.

Student A Activity Sheet

In these activities you will play Mastermind with pencil and paper. The six colors are A, B, C, D, E, F and codewords have length four. If you are using a Mastermind playing board instead of pencil and paper, first list the six colors in your Mastermind game in alphabetical order and label them A, B, C, D, E, F.

Suppose you are the codebreaker. The *Mastermind Rad* is a tool to help you determine the code selected and concealed by the codemaker. This tool is a visualization of Donald Knuth's 1976 algorithm published in "The computer as master mind," *Journal of Recreational Mathematics*, Volume 9, Issue 1.

You begin with move AABB, and then tell the *Mastermind Rad* the response in the inner disk, and the *Mastermind Rad* tells you which test code to play next, and so on. To use the *Mastermind Rad*, you position each ring according to the codemaker responses, and you read from the center disc outwards to the outermost ring. Your first move is AABB, find the response on the center disc, make the corresponding move, and then find the response on the next ring, and so on.

The *Mastermind Rad* finishes in one of the following situations.

(i) **"Think!"** All the information necessary to deduce the code is already on the playing board, and then you must think and then present the correct code.

(ii) ☺ There is not enough information on the playing board for you to deduce the code, but you have presented a test code, and by luck and logic guessed the correct code.

(iii) **"50/50?"** Exactly two test codes satisfy the information on the playing board, and one of them is the correct code. Then you must think and deduce the two candidates, and simply try them one after the other.

(iv) **"WXYZ!"** After you play WXYZ, the response will put you into situation "Think!" or situation ☺.

(v) **"WXYZ?"** After you play WXYZ, the response will put you into situation "50/50?", or situation "Think!", or situation ☺.

Notice: the *Mastermind Rad* does *not* find the concealed code for you! Rather, the *Mastermind Rad* leads to a situation where you must use logic to deduce the code from the available information.

Activities Using the *Mastermind Rad* to Play Mastermind

Activity 1. You are the codebreaker. Use the *Mastermind Rad* to determine the code selected by your partner.

Activity 2. Next switch roles: you are now the codemaker. Select and conceal the code CBAA, and respond to your partners attempts.

Activity 3. You are the codebreaker. Use the *Mastermind Rad* to determine the code selected by your partner.

Activity 4. You are the codemaker. Select and conceal the code EDCC, and respond to your partner's attempts.

Activity 5. You are the codebreaker. Use the *Mastermind Rad* to determine the code selected by your partner.

Activity 6. You are the codemaker. Select and conceal the code CEDC, and respond to your partner's attempts.

Activities on Counting Candidate Codes After First Response

Activity 7. Suppose you do the initial test code AAAA and receive response 00. How many candidate codes are there? Similarly, for each of the following, if WXYZ is an initial test code, and bw is the response, find the number of candidate codes.

AAAB 00
AABB 00
AABC 00
ABCD 00
Challenge: AAAB 01

Activities about Knuth's Algorithm

In the previous activities you used the *Mastermind Rad* to play and win Mastermind. Next let's take a look at the ideas behind the *Mastermind Rad*, namely Knuth's algorithm from 1976. The conceptual picture behind the algorithm is that each response narrows down as much as possible (while independently of the response) the set of candidate test codes, until finally there is a unique candidate, exactly two candidates, or the correct code is presented by a combination of luck and logic. But what is the selection principle of the codebreaker's test codes? The next exercises elucidate Knuth's idea.

Activity 8. The table below shows the number of candidate codes that remain after an initial test code is played and a response is given, or in other words, how many codes satisfy a given response to a given initial test code. Find your answers to Activity 7 in the table. Hint: if your answers are not in the corresponding positions on the table, then revise your answers to Activity 7.

Remaining Candidate Test Codes After First Move and Response					
	First Test Code				
Response	AAAA	AAAB	AABB	AABC	ABCD
00	**625**	256	256	81	16
01		308	256	**276**	152
02		61	96	222	**312**
03			16	44	136
04			1	2	9
10	500	**317**	**256**	182	108
11		156	208	230	252
12		27	36	84	132
13				4	8
20	150	123	114	105	96
21		24	32	40	48
22		3	4	5	6
30	20	20	20	20	20
40	1	1	1	1	1

Activity 9. The main idea of Knuth's algorithm is that the codebreaker always plays a test code that "*minimizes the maximum number of remaining possibilities*" given the previous information on the playing board, amongst the possibilities for the upcoming response. Let's consider the meaning of this test code selection principle for the codebreaker's very first move. Look at the table above. If the codebreaker plays the first move AAAA, what is the maximum number of remaining candidate test codes, amongst the possibilities for the upcoming response? If the codebreaker plays the first move ABCD, what is the maximum number of remaining candidate test codes, amongst the possibilities for the upcoming response? Now, amongst the listed first test codes, which one minimizes the maximum number of remaining candidate test codes? Compare your answer with Move 1 on the *Mastermind Rad*.

Activity 10. For Move 2, Knuth's algorithm continues the test code selection principle "*minimize the maximum number of remaining possibilities*" explained in Activity 9. However, beginning with Move 2, information on the playing board must also be considered in the minimization of maxima slogan. At this point, there are too many possibilities for a human to compute and count, and a computer (or the *Mastermind Rad!*) is needed. By analogy with the previous exercise, we can understand what the computer does. Namely, in the previous exercises a table was partially computed and presented, namely a table of remaining candidate test codes for each possible first test code and possible response, and then you observed that AABB should be Move 1. What should Move 2 be? The computer would make a similar table of remaining candidate test codes, for all possible second moves and all possible second responses, however this table must take into account the information already on the board. Then the computer would find the second move which minimizes the maximum number in

the respective column. The *Mastermind Rad* is a geometrically organized enumeration of the computer outputs, the *Mastermind Rad* itself does not compute. Your task in this activity is to explain to your partner how Knuth's algorithm selects the test code for Move 2. Your partner will also explain it to you.

Activity 11. A perhaps counter-intuitive feature of Knuth's algorithm is that sometimes it proposes the codebreaker play a test code that is not among the remaining candidate test codes, i.e. a test code that contradicts information already on the board. Such test codes can actually be smart moves and yield information. For instance, suppose the codebreaker knows from previous moves that there are no A's in the concealed code, and wants to know if any B's are in the concealed code. Then the codebreaker can play AAAB. Why is this code not among the remaining candidate test codes, and why does the response tell the codebreaker whether or not B is in the concealed code? For a very clever and more involved example, see page 3 of Knuth's article.

Activity 12. In order for a computer to implement Knuth's algorithm, a formula is needed to compute the response bw to a given test code for a given concealed code. The number b of black pegs is straightforward to compute by comparing the respective digits of the test code and concealed code, and counting the number of digits in agreement. On the other hand, the number w of correct colors in wrong positions is surprisingly difficult to precisely define and compute. One formula[7] for w is

$$w = \left(\sum_{i=1}^{6} \min(c_i, g_i) \right) - b,$$

where c_i is the number of times the color i is in the concealed code and g_i is the number of times i is in the test code. Verify that the formula holds for your moves in Activities 1 and 2.

Activity 13. Every algorithm has an input and an output. What are the input and the output for Knuth's algorithm and the *Mastermind Rad*? Hint: strictly speaking, the output is not the concealed code. As a follow up activity, can each move be considered an algorithm in itself?

Activity about General Searches in Mastermind

Activity 14. This activity does *not* concern Knuth's algorithm. Which of the following problems will require going through few codes and which all codes once? Which of the following problems require a more intensive search? For which of the following problems can we make use of the concealed code?

- Developing the next move
- Computing the response
- Checking if a potential test code satisfies the information on a given playing board
- Checking that a code fulfills an information

[7] From the website http://mathworld.wolfram.com/Mastermind.html

- Checking that a code does not fulfill an information
- Finding a test code that distinguishes two codes

Student B Activity Sheet

In these activities you will play Mastermind with pencil and paper. The six colors are A, B, C, D, E, F and codewords have length four. If you are using a Mastermind playing board instead of pencil and paper, first list the six colors in your Mastermind game in alphabetical order and label them A, B, C, D, E, F.

Suppose you are the codebreaker. The *Mastermind Rad* is a tool to help you determine the code selected and concealed by the codemaker. This tool is a visualization of Donald Knuth's 1976 algorithm published in "The computer as master mind," *Journal of Recreational Mathematics*, Volume 9, Issue 1.

You begin with move AABB, and then tell the *Mastermind Rad* the response in the inner disk, and the *Mastermind Rad* tells you which test code to play next, and so on. To use the *Mastermind Rad*, you position each ring according to the codemaker responses, and you read from the center disc outwards to the outermost ring. Your first move is AABB, find the response on the center disc, make the corresponding move, and then find the response on the next ring, and so on.

The *Mastermind Rad* finishes in one of the following situations.

(i) **"Think!"** All the information necessary to deduce the code is already on the playing board, and then you must think and then present the correct code.

(ii) ☺ There is not enough information on the playing board for you to deduce the code, but you have presented a test code, and by luck and logic guessed the correct code.

(iii) **"50/50?"** Exactly two test codes satisfy the information on the playing board, and one of them is the correct code. Then you must think and deduce the two candidates, and simply try them one after the other.

(iv) **"WXYZ!"** After you play WXYZ, the response will put you into situation "Think!" or situation ☺.

(v) **"WXYZ?"** After you play WXYZ, the response will put you into situation "50/50?", or situation "Think!", or situation ☺.

Notice: the *Mastermind Rad* does *not* find the concealed code for you! Rather, the *Mastermind Rad* leads to a situation where you must use logic to deduce the code from the available information.

Activities Using the *Mastermind Rad* to Play Mastermind

Activity 1. You are the codemaker. Select and conceal the code BBAA, and respond to your partner's attempts.

Activity 2. Next switch roles: you are now the codebreaker. Use the *Mastermind Rad* to determine the code selected by your partner.

Activity 3. You are the codemaker. Select and conceal the code DECC, and respond to your partner's attempts.

Activity 4. You are the codebreaker. Use the *Mastermind Rad* to determine the code selected by your partner.

Activity 5. You are the codemaker. Select and conceal the code ECCD, and respond to your partner's attempts.

Activity 6. You are the codebreaker. Use the *Mastermind Rad* to determine the code selected by your partner.

Activities on Counting Candidate Codes After First Response

Activity 7. Suppose you do the initial test code AAAA and receive response 00. How many candidate codes are there? Similarly, for each of the following, if WXYZ is an initial test code, and bw is the response, find the number of candidate codes.

AAAB 00
AABB 00
AABC 00
ABCD 00
Challenge: AAAB 01

Activities about Knuth's Algorithm

In the previous activities you used the *Mastermind Rad* to play and win Mastermind. Next let's take a look at the ideas behind the *Mastermind Rad*, namely Knuth's algorithm from 1976. The conceptual picture behind the algorithm is that each response narrows down as much as possible (while independently of the response) the set of candidate test codes, until finally there is a unique candidate, exactly two candidates, or the correct code is presented by a combination of luck and logic. But what is the selection principle of the codebreaker's test codes? The next exercises elucidate Knuth's idea.

Activity 8. The table below shows the number of candidate codes that remain after an initial test code is played and a response is given, or in other words, how many codes satisfy a given response to a given initial test code. Find your answers to Activity 7 in the table. Hint: if your answers are not in the corresponding positions on the table, then revise your answers to Activity 7.

Remaining Candidate Test Codes After First Move and Response					
	First Test Code				
Response	AAAA	AAAB	AABB	AABC	ABCD
00	**625**	256	256	81	16
01		308	256	**276**	152
02		61	96	222	**312**
03			16	44	136
04			1	2	9
10	500	**317**	**256**	182	108
11		156	208	230	252
12		27	36	84	132
13				4	8
20	150	123	114	105	96
21		24	32	40	48
22		3	4	5	6
30	20	20	20	20	20
40	1	1	1	1	1

Activity 9. The main idea of Knuth's algorithm is that the codebreaker always plays a test code that "*minimizes the maximum number of remaining possibilities*" given the previous information on the playing board, amongst the possibilities for the upcoming response. Let's consider the meaning of this test code selection principle for the codebreaker's very first move. Look at the table above. If the codebreaker plays the first move AAAA, what is the maximum number of remaining candidate test codes, amongst the possibilities for the upcoming response? If the codebreaker plays the first move ABCD, what is the maximum number of remaining candidate test codes, amongst the possibilities for the upcoming response? Now, amongst the listed first test codes, which one minimizes the maximum number of remaining candidate test codes? Compare your answer with Move 1 on the *Mastermind Rad*.

Activity 10. For Move 2, Knuth's algorithm continues the test code selection principle "*minimize the maximum number of remaining possibilities*" explained in Activity 9. However, beginning with Move 2, information on the playing board must also be considered in the minimization of maxima slogan. At this point, there are too many possibilities for a human to compute and count, and a computer (or the *Mastermind Rad!*) is needed. By analogy with the previous exercise, we can understand what the computer does. Namely, in the previous exercises a table was partially computed and presented, namely a table of remaining candidate test codes for each possible first test code and possible response, and then you observed that AABB should be Move 1. What should Move 2 be? The computer would make a similar table of remaining candidate test codes, for all possible second moves and all possible second responses, however this table must take into account the information already on the board. Then the computer would find the second move which minimizes the maximum number in

the respective column. The *Mastermind Rad* is a geometrically organized enumeration of the computer outputs, the *Mastermind Rad* itself does not compute. Your task in this activity is to explain to your partner how Knuth's algorithm selects the test code for Move 2. Your partner will also explain it to you.

Activity 11. A perhaps counter-intuitive feature of Knuth's algorithm is that sometimes it proposes the codebreaker play a test code that is not among the remaining candidate test codes, i.e. a test code that contradicts information already on the board. Such test codes can actually be smart moves and yield information. For instance, suppose the codebreaker knows from previous moves that there are no A's in the concealed code, and wants to know if any B's are in the concealed code. Then the codebreaker can play AAAB. Why is this code not among the remaining candidate test codes, and why does the response tell the codebreaker whether or not B is in the concealed code? For a very clever and more involved example, see page 3 of Knuth's article.

Activity 12. In order for a computer to implement Knuth's algorithm, a formula is needed to compute the response bw to a given test code for a given concealed code. The number b of black pegs is straightforward to compute by comparing the respective digits of the test code and concealed code, and counting the number of digits in agreement. On the other hand, the number w of correct colors in wrong positions is surprisingly difficult to precisely define and compute. One formula[8] for w is

$$w = \left(\sum_{i=1}^{6} \min(c_i, g_i) \right) - b,$$

where c_i is the number of times the color i is in the concealed code and g_i is the number of times i is in the test code. Verify that the formula holds for your moves in Activities 1 and 2.

Activity 13. Every algorithm has an input and an output. What are the input and the output for Knuth's algorithm and the *Mastermind Rad*? Hint: strictly speaking, the output is not the concealed code. As a follow up activity, can each move be considered an algorithm in itself?

Activity about General Searches in Mastermind

Activity 14. This activity does *not* concern Knuth's algorithm. Which of the following problems will require going through few codes and which all codes once? Which of the following problems require a more intensive search? For which of the following problems can we make use of the concealed code?

- Developing the next move
- Computing the response
- Checking if a potential test code satisfies the information on a given playing board
- Checking that a code fulfills an information

[8]From the website http://mathworld.wolfram.com/Mastermind.html

- Checking that a code does not fulfill an information
- Finding a test code that distinguishes two codes

Thomas M. Fiore, Department of Mathematics and Statistics, University of Michigan-Dearborn, 4901 Evergreen Road, Dearborn, MI 48128
and
NWF I - Mathematik, Universität Regensburg, Universitätsstrasse 31, 93040 Regensburg, Germany
E-mail address: tmfiore@umich.edu
URL: http://www-personal.umd.umich.edu/~tmfiore/

Alexander Lang, NWF I - Mathematik, Universität Regensburg, Universitätsstrasse 31, 93040 Regensburg, Germany
E-mail address: alexander.lang@stud.uni-regensburg.de

Antonella Perucca, NWF I - Mathematik, Universität Regensburg, Universitätsstrasse 31, 93040 Regensburg, Germany
E-mail address: Antonella.Perucca@mathematik.uni-regensburg.de
URL: http://www.uni-regensburg.de/Fakultaeten/nat_Fak_I/perucca/