DIDACTICA MATHEMATICA, Vol. 43 (2025) , pp. 125–127

THE (ALMOST) IMPOSSIBLE PUZZLE

Mihai Iancu

Abstract. Tom, Jerry and Spike play the following game. Tom generates a random binary code of length n, which he sends to Spike along with a *secret number* between 1 and n. Spike flips exactly one bit in the code and then sends only the modified binary code to Jerry. Is there a pre-established strategy for Spike and Jerry such that Jerry can determine the *secret number* solely by looking at the code sent by Spike?

MSC 2000. 00A08.
Key words. bit, puzzle.

1. SOLUTION

The puzzle and its solution(s) have been discussed in [4, 2, 1, 3] and others, mainly focusing on the case $n = 64$ (often formulated as a chessboard puzzle). See [5] for a generalization. We present a short proof of the following theorem, understandable for a first-year student taking a course in Linear Algebra.

THEOREM 1. There is a pre-established strategy for Spike and Jerry if and only if $n = 2^m$ for some $m \in \mathbb{N}$.

Proof. Let $I = \{1, \dots, n\}$, where $n \geq 2$ (the case $n = 1$ is trivial). In the following, we consider $\mathbb{Z}_2^m = \mathbb{Z}_2 \times ... \times \mathbb{Z}_2$ as a vector space over $\mathbb{Z}_2$, where $m \in \mathbb{N}^*$. For $i \in I$, let $e_i \in \mathbb{Z}_2^n$ be the canonical vector with 1 on the i-th position and zeros elsewhere. Note that $z + e_i$ is the code z with the i-th bit flipped, for every $z \in \mathbb{Z}_2^n$ and $i \in I$.

Spike and Jerry have a pre-established strategy if and only if there exists $f : \mathbb{Z}_2^n \to I$ such that for every $z \in \mathbb{Z}_2^n$ and every $k \in I$ there exists $i \in I$ such that $f(z + e_i) = k$.

"⇒" Assume there exists f as above. For every $z \in \mathbb{Z}_2^n$, let

$$B_z = \{z + e_1, \dots, z + e_n\}$$

and note that

$$f(B_z) = I. \tag{1}$$

Claim 1: $\bigcup_{x \in f^{-1}(1)} B_x = \mathbb{Z}_2^n$.

Indeed, if $z \in \mathbb{Z}_2^n$, then there exists $i \in I$ such that $f(z + e_i) = 1$, and thus there exists $x \in f^{-1}(1)$ such that $z = x + e_i \in B_x$.

Claim 2: $B_x \cap B_y = \emptyset$, for $x, y \in f^{-1}(1)$ with $x \neq y$.

Let $x, y \in f^{-1}(1)$. If there exist $i, j \in I$ such that $x + e_i = y + e_j = z$, then $f(z + e_i) = f(z + e_j) = 1$, hence, by (1), $i = j$, and thus $x = y$.

Claims 1 and 2 imply $n \cdot \text{card}\, f^{-1}(1) = 2^n$, therefore $n = 2^m$ for some $m \in \mathbb{N}$.

"$\Rightarrow$" Assume that $n = 2^m$ for some $m \in \mathbb{N}$. First, Spike and Jerry define the one-to-one map $\varphi : I \to \mathbb{Z}_2^m$ such that $\varphi(i)$ is the binary representation of $i-1$ with m bits, for every $i \in I$. Next, they define $g : B_0 \to \mathbb{Z}_2^m$ such that $g(e_i) = \varphi(i)$, for every $i \in I$. Next, they extend g by linearity to have $g : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$. Finally, they take $f = \varphi^{-1} \circ g : \mathbb{Z}_2^n \to I$.

Let $z \in \mathbb{Z}_2^n$ and $k \in I$. Since $g(B_0) = \mathbb{Z}_2^m$, there exists $i \in I$ such that $g(e_i) = g(z + e_k)$. Then $g(z + e_i) = g(z) + g(z + e_k) = g(z + z + e_k) = g(e_k)$, so $f(z + e_i) = k$.

□

2. EXECUTING THE SOLUTION

Assume Spike and Jerry want to program their strategy (which is viable if $n = 2^m$, for some $m \in \mathbb{N}$) in Python. They can use the `xor` operator which performs addition in $\mathbb{Z}_2$ on the corresponding bits of two integers. Examples: `xor(0,0)` and `xor(1,1)` return 0, while `xor(0,1)` and `xor(1,0)` return 1; `xor(10,7)` returns 13, because the binary representations (with 4 bits) of 10, 7 and 13, are `1010`, `0111`, respectively `1101`.

In the following, we present the Python coding of the solution given in the proof of Theorem 1.

```
from operator import xor
#Spike and Jerry pre-establish this function
def f(binary_code):
    ones_positions=[i for i,b in enumerate(binary_code) if b!=0]
    output=0
    for k in ones_positions:
        output=xor(output,k)
    return output+1
```

Let's test the function, by executing Spike and Jerry's plan on some data produced by Tom.

Tom's part:

```
from random import choices
n=128
# Tom generates the random binary code
rnd_code=choices([0,1],k=n)
# Tom writes the secret number
k=113
```

Spike's part:

```
# Spike uses Tom's code and secret number,
# to find the bit i that he has to flip
code=rnd_code.copy(); code[k-1]=xor(code[k-1],1)
i=f(code) # Spike calls the pre-established function
# Spike flips the i-th bit
rnd_code[i-1]=xor(rnd_code[i-1],1)
```

Jerry's part:

```
# Jerry just calls the pre-established function
print("The secret number is %d." %f(rnd_code))
```

The code can be easily tested in, e.g., a Jupyter notebook [6].

REFERENCES

[1] Berry, N., *Impossible Escape?*, http://datagenetics.com/blog/december12014/index.html, 2014.
[2] Parker, M., *The almost impossible chessboard puzzle*, https://www.youtube.com/watch?v=as7Gkm7Y7h4, 2020.
[3] Rahman, C.L., *Impossible Chessboard Escape Puzzle*, https://github.com/CoryLR/impossible-chessboard-escape-puzzle, 2021.
[4] Sanderson, G., *The impossible chessboard puzzle*, https://www.3blue1brown.com/lessons/chessboard-puzzle, 2020.
[5] Srivastava, K., *Generalised 'Almost Impossible' Chessboard Problem*, https://karansrivastava.com/files/Chess.pdf, University of Wisconsin-Madison, 2020.
[6] https://jupyter.org/try-jupyter/.

Faculty of Mathematics and Computer Science, Babeș-Bolyai University, Cluj-Napoca
e-mail: mihai.iancu@ubbcluj.ro