MASUD 2022-2023

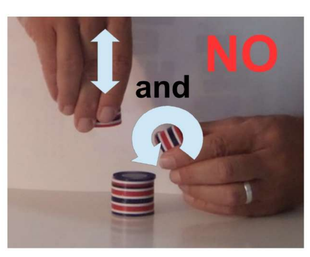
# TURNING OVER COINS

BEATRICE TEOC, IRINA PAUL, EMMA VASII, SARA CRIȘAN, BRIANA FILIMON, IRINA TEREC, MARIA BREAZ, BIANKA TAVASZI, GLORIA DOBRA, ALEXIA BUCȘA, CLARA BOB, RIANNA TANCZEL, CARLOS POP

COLEGIUL NAȚIONAL „EMIL RACOVIȚĂ” CLUJ-NAPOCA

## INTRODUCTION

We have a certain number of stacked coins. A stack of coins can be turned over partly, but only by starting from the top of the stack. How many times do we have to repeat this operation to get all the coins on the front side? How should we proceed in general?
Restrain: we cannot take out coins from the middle of the stack and turn them over.


## METHODS AND RESULTS

### FIRST STEPS INTO THE PROBLEM

Each of us tried to visualise the problem, so we made some coins out of various materials such as plastic and foam.

PLASTIC (3D PRINTED)

FOAM

### TOTAL NUMBER OF POSSIBLE COMBINATIONS

We tried out cases on paper and then created an Excel file containing the possible combinations of n coins. We realised that the next formula describes the total number of possible combinations of n coins:

$$\sum_{k=0}^{n} C_n^k = 2^n$$

### TOTAL NUMBER OF POSSIBLE SWITCHES

We can find out the numbers of switches needed simply by counting the number of ‘changes’ in the stack of coins. A ‘change’ consists in the alternations between 1 and 0, where 1 represents the black side of the coin and 0, the red side of the coin.
For example, the string 100101 has 5 ‘changes’: 1 00 1 0 1. Therefore, if a stack of coins has n ‘changes’, we would have to make (x-1) switches if the bottom coin is 1 and x switches if the bottom coin is 0 (because we have to make one final switch after our stack is all 0s) to reach our desired goal, which is to have all coins on the front side.

**x** changes

- **(x-1)** switches if bottom coin is 1
- **x** switches if bottom coin is 0

### ALGORITHM

Our algorithm checks whether or not the top coin is the same as the second one. If they are different, the algorithm switches the top coin and goes on. If not, it goes on and checks if the second and third coin are different, in which case it switches the first two and then moves on.

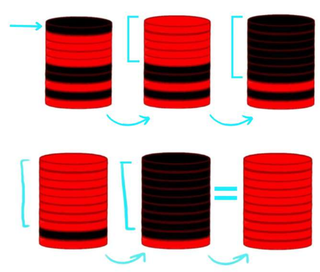
For example, x=5:

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| | 1 | 2 | 3 | 4 | 5 |

**CASE ONE**
The bottom coin is 0, so we will make 5 switches

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| | 1 | 2 | 3 | 4 |

**CASE TWO**
The bottom coin is 1, so we will make 4 switches



### TOTAL NUMBER OF SWITCHES

We discovered a formula which describes the average number of switches, based on the number of coins n.

the average number of switches = n/2

For example, n=3.
The average number of switches is 1.5 :

- 1 case with 0 switches
- 3 cases with 1 switch
- 3 cases with 2 switches
- 1 case with 3 switches

CASE 1 => number of switches = 0

| | | |
|---|---|---|
| 1 | | |
| 1 | | |
| 1 | | |

CASE 2 => number of switches = 1

| | |
|---|---|
| 0 | 1 |
| 0 | 1 |
| 0 | 1 |

CASE 3 => number of switches = 1

| | |
|---|---|
| 0 | 1 |
| 1 | 1 |
| 1 | 1 |

CASE 4 => number of switches =1

| | |
|---|---|
| 0 | 1 |
| 0 | 1 |
| 1 | 1 |

CASE 5 => number of switches = 2

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |

CASE 6 => number of switches = 2

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 1 |

CASE 7 => number of switches = 2

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 0 | 1 |
| 0 | 0 | 1 |

CASE 8 => number of switches = 3

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 |

### PROGRAMS

Each team wrote a program in C++ or Python which takes as input the number of coins and their order in the stack. The programs solve the stack, show the steps and the total number of moves. You can check them out scanning the following QR code:

### APP

We programmed a web application using JavaScript, HTML and CSS to make it easier to visualize solving a stack of coins. The app shows each step and the total number of moves required to solve the stack.

Scan the QR code to access the app or go to ***homebase.ro/game***.