THE MACHINE TO PLAY THE STICKS GAME

Colegiul National “Emil Racovita” Cluj-Napoca, Romania
Petra Hedesiu, Matei Bojan, Stefan Oprean, Matei Coldea, Petru Saveanu, George Iorga

Problem statement

The game of sticks consists of two players with n sticks, each in turn removing one or two sticks, the one who takes the last stick wins.

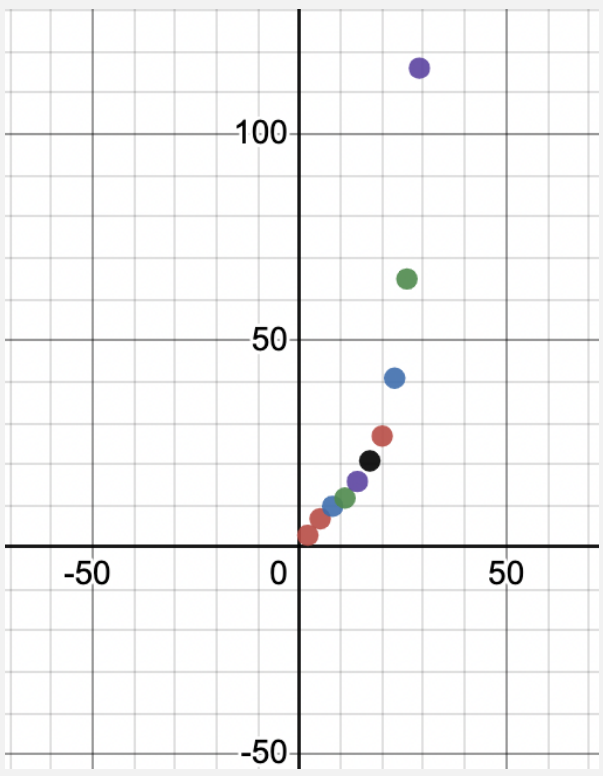
Introduction

The objective of this research is to determine the average number of games after which the machine always defeats the human player. The angle chosen to carry out our research was first developing a C++ programme which simulates the game. After finding a dependence between the number of sticks in the game and the number of games played, we focused on obtaining multiple charts for a visual representation of our data.

Simulating the machine

The machine can’t learn from a player with a random strategy, so we found the optimal strategy for the player: j[] = {1, 2, 0, 1, 2, 0, 1, 2, ...}. In order for the machine to always win, each cup it falls into needs to have only one type of balls in it. If the number of cups is a multiple of 3, the machine can never always win, because it starts from a cup where the player’s strategy is j[n] = 0 and wins against the machine.

Visual representation of our data



Alpha-Beta pruning and Minimax algorithm

Alpha-Beta pruning is a method for optimizing the minimax algorithm by greatly reducing the computation time. This optimization allows for faster search and the ability to delve deeper into the game tree.
It eliminates branches that do not need to be searched because a better move has already been identified. The technique is named for the two parameters it adds to the minimax function: alpha and beta.
Alpha represents the highest value that the maximizer can currently ensure at that level or higher, while beta represents the lowest value that the minimizer can currently ensure at that level or lower.

Calculating the probability using recurrences

We will consider 3 arrays of numbers which represent the probability that after n rounds there will be 0, 1 or 2 wrong balls. We can calculate these arrays using previous members. We will notate N0, N1 and N2 (there are still 2 bad balls). Using probability theory we can obtain:

```
N0[n] = N1[n-1] / (n + 1)
N1[n] = N1[n-1] * (n) / (n + 1) + N2[n-1] * 2 / (n + 3)
N2[n] = N2[n-1] * (n + 1) / (n + 3)
```

The Monte Carlo algorithm

The Monte Carlo algorithm is a method for solving problems using simulations. The basic idea behind the algorithm is to simulate many possible scenarios, track the outcomes, and then use the collected data to make predictions or decisions.
In the case of the game of sticks, the Monte Carlo algorithm can be used to determine the average number of games it takes for the machine to always win.
The algorithm would involve the following steps:

1. Initialise the machine
2. Run a large number of simulations
3. Track the machine’s performance
4. Analize the results

Probabilistic approach: Poisson

μ - mean of “bad balls" (Monte Carlo algorithm)

$$P(X) = \frac{\lambda^x e^{-\lambda}}{X!}$$

$P(x; \mu) = (e\text{-}\mu * \mu x) / x!$

$P_n(1) \longrightarrow P_{n+1}(0)$, $P_n(1) \longrightarrow P_{n+1}(1)$, $P_n(2) \longrightarrow P_{n+1}(1)$, $P_n(2) \longrightarrow P_{n+1}(2)$

$\text{average} = \Sigma\ (i\ *\ P_i(0))$

The probability of having 0 bad balls after x games: $P(0; \mu) = (e\text{-}\mu * \mu 0) / 0!$

Obs:
"bad ball" = balls that according to the array of optimal moves calculated with the minimax algorithm, do not represent the optimal move.

Spline interpolation

In the mathematical field of numerical analysis, spline interpolation is a form of interpolation where the interpolant is a special type of piecewise polynomial called a spline.

The curvature of any y = y(x) is defined as:

$$\kappa = \frac{|y''|}{\left(1 + y'^2\right)^{3/2}}$$

$$\begin{cases} q_i(x_i) = q_{i+1}(x_i) = y_i \\ q_i'(x_i) = q_{i+1}'(x_i) \\ q_i''(x_i) = q_{i+1}''(x_i) \end{cases} \quad 1 \le i \le n-1.$$

The use of a matrix

The aforementioned graph contains a few points (xi, yi)
x = the number n
y = the average number of games played before the machine always wins
We will try to create a function that characterises each of these cases, as before, but with a friendlier method.
We used matrices for t equations:

$$yi = C1 \cdot xi^{t-1} + C2 \cdot xi^{t-2} + \ldots + C_{t-1} \cdot xi + Ct$$

We placed the xi^h on each column (h=0,1,...t-1) and calculated the determinant of the matrix , then switched each column with yi and calculated those determinants to i.
So for Ci, the expression is:

$Ci = \Delta(i - 1)/\Delta$, for i = 1,2..t.