Master Mind

Arthur

May 19, 2024

1 Introduction

Mastermind: A Classic Game of Logic and Deduction Mastermind is a renowned two-player board game that challenges players with the task of code-breaking under a veil of secrecy. Invented in 1970 by Mordecai Meirowitz, an Israeli telecommunications expert, Mastermind has since become a beloved puzzle game worldwide1. The game is elegantly simple in its components but complex in its strategic depth. It consists of a decoding board, a variety of colored pegs, and key pegs used for providing feedback. One player, the "codemaker," creates a secret code using the colored pegs, which the other player, the "codebreaker," must deduce within a limited number of turns. The code is composed of a sequence of colored pegs hidden from the codebreaker's view. With each guess, the codemaker provides feedback with key pegs—black or white—indicating correct colors in the correct or incorrect positions, respectively. The codebreaker uses this feedback to refine subsequent guesses. Mastermind is not just a game but a battle of wits, requiring logical deduction, pattern recognition, and strategic planning. It's a testament to the enduring appeal of cerebral challenges and the joy of unraveling mysteries—one peg at a time.

2 Complexity of Move Search

The complexity of the move search in the game of Mastermind can vary depending on the stage of the game and the strategies employed. Here's a general overview of the game:

2.1 Early Game

At the beginning, with four pegs and six colors (denoted as MM(6; 4)), there are

$6^4 = 1296$

possible codes. The search space is at its largest, and without any information, the complexity is high. Strategies like simulated annealing (SA) or maximum expected reduction in consistency (MERC) can be used to navigate the search space

2.2 Mid Game

As the game progresses and feedback is received, the search space reduces. Algorithms like the one developed by Donald Knuth, which minimizes the maximum query partition sets, can solve the pattern in five moves or fewer by progressively reducing the number of possible patterns2. This approach requires an average of 4.478 guesses

2.3 End Game

In the end game of Mastermind, the complexity of the move search can be significantly reduced compared to the early and mid-game stages. As the game progresses and the codebreaker receives more feedback, the number of possible codes diminishes, making it easier to deduce the correct code.The end game complexity is often determined by the number of remaining possible codes and the strategies employed to solve them. For instance, once the codebreaker has narrowed down the colors and positions of some pegs, they can use a process of elimination and logical deduction to finalize the positions of the remaining pegs. This might involve swapping the positions of known pegs to find the correct

configuration, which can be done in a few moves. According to one strategy shared on a puzzle forum, once you know what pegs go in the front and back, there is a pattern you can follow to settle the final positions1. For example, if you receive feedback of 2 black and 0 white pegs for a guess of PRGY, you can then try PRYG; if you get 4 black pegs, the game is won. If not, you continue with the pattern until the correct sequence is found.

3 Quantum Complexity

3.1 Non-Adaptive Complexity

$$\Theta\left(\frac{n\log(k)}{max\left(\log\left(\frac{n}{k}\right),1\right)}\right)$$

3.2 Classical Complexity

$$\Theta\left(n\cdot\frac{log(k)}{log(n)+\frac{k}{n}}\right)$$

4 How to write Mathematics

In the context of the Mastermind game, a system of qubits can be used to represent the various states of the game. Here's a conceptual framework for such a system: Qubits System: Each peg in Mastermind can be represented by a qubit. Since there are typically six colors to choose from, we can use a 3-qubit system to represent each peg (since $2^3 = 8$, which is sufficient to encode six colors). For a standard game with four pegs, this would require a 12-qubit system to represent the entire code. Game States: The game states in Mastermind can be defined by the possible arrangements of pegs. Each game state is a unique combination of peg colors. In quantum terms, each game state corresponds to a specific state vector in the Hilbert space spanned by the basis vectors representing the color combinations. State Vectors: A state vector in this system would be a tensor product of the state vectors of individual qubits. For example, if we assign a specific vector to each color, the state vector for a particular arrangement of pegs would be the tensor product of the vectors corresponding to the colors of those pegs.

5 Qubit System for Mastermind

Pegs: Let's consider a standard Mastermind game with (n) pegs. Colors: Assume there are (m) different colors available.

5.1 Qubits per Peg

Since we have (m) colors, we need ($\lceil \log_2(m) \rceil$)qubits to represent each color. For simplicity, let's assume (m) is a power of 2, so

$$(\log_2(m))$$

is an integer.

5.2 Total Qubits

The total number of qubits required to represent the game state is

$$(n \times \log_2(m))$$

6 Game States and State Vectors

6.1 Basis States

Each color is represented by a unique basis state in the computational basis. For example, if (m = 8), we can represent the colors by the states

$$(|000\rangle), (|001\rangle), (|010\rangle), \ldots, (|111\rangle)$$

Game State: A game state is a specific configuration of colors on the pegs. It can be represented by a tensor product of the basis states of individual qubits. State Vector: The state vector for a game state is a vector in a Hilbert space spanned by the basis vectors. It is a direct product of the state vectors representing each peg's color. For example, if we have a 4-peg game with 8 colors, we would need 3 qubits per peg, resulting in a 12-qubit system. A game state where the first peg is color 1 (represented by

$$(|000\rangle)),$$

the second peg is color 2 (represented by

$$(|001\rangle)),$$

the third peg is color 3 (represented by

$$(|010\rangle)),$$

and the fourth peg is color 4 (represented by

$$(|011\rangle)),$$

would have the state vector:

$$000>001>010>011>$$

This state vector represents the entire game state in the quantum Mastermind game. Each guess and feedback in the game would correspond to a transformation of this state vector within the Hilbert space, following the rules of quantum mechanics. Feedback and Measurement: Feedback States: Feedback from the codemaker can also be represented using qubits. For example, a black peg indicating a correct color in the correct position, and a white peg indicating a correct color in the wrong position, can be represented by different qubit states. Measurement: After each guess, a measurement is performed on the feedback qubits to obtain the classical feedback, which then informs the next guess.

7 Quantum Gates

7.1 Classical to Quantum Mapping

Classical comparison and evaluation can be mapped to quantum operations that compare qubit states and apply transformations based on those comparisons.

7.2 Quantum Gate Equivalents

NOT Gate: Represented by the Pauli-X gate, (X), which flips the state of a qubit.

$$\mathrm{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

AND Gate: Implemented using the Toffoli gate, (CCNOT),which flips the target qubit if both control qubits are in the state ($|1\rangle$).

$$\mathrm{CCNOT} = \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix}$$

OR Gate: No direct quantum equivalent, but can be constructed using Toffoli gates and ancillary qubits.

7.3 Feedback Mechanism

Controlled gates can be used to evaluate feedback. For example, a controlled-Z gate, (CZ), might represent a correct color in the correct position.

$$\mathrm{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

7.4 State Preparation

The initial state is prepared by setting qubits to represent the first guess. This can be done using a series of (X) gates applied to the$|0\rangle$state to achieve the desired initial state.

7.5 Sequential Application

The quantum gates are applied in sequence to simulate the steps of the classical algorithm. This involves using the gate matrices to transform the state vector representing the current game state.

7.6 Measurement

After the gates are applied, the qubits are measured, which can be represented mathematically by projecting the state vector onto the measurement basis.

8 Application

To apply the designed quantum gates to superposition states for the parallelization of the move search in a game like Mastermind, we'll mathematically construct a quantum circuit. This circuit will utilize superposition to explore all possible game states simultaneously and entanglement to correlate the feedback with the game states. Let's assume we have a Mastermind game with (n) pegs and (m) colors. We'll need ($\lceil \log_2(m) \rceil$)qubits to represent each color, and thus ($n \times \lceil \log_2(m) \rceil$) qubits to represent the entire game state.

8.1 Initial Superposition

Apply the Hadamard gate (H) to each qubit to create a superposition of all possible states.

$$\mathrm{H} = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The initial state ($|0\rangle^{\otimes n \times \lceil \log_2(m) \rceil}$) becomes a superposition of all possible game states. Gate Application: Apply the quantum gates designed to simulate the Mastermind algorithm. For example, the Toffoli gate (CCNOT) for an AND operation:

$$\text{CCNOT} = \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix}$$

Here, (I) is the identity matrix, and (X) is the Pauli-X gate. Entanglement Use controlled gates to create entanglement between the game state qubits and the feedback qubits. This correlates the possible game states with the feedback received.

8.2 Parallel Computation

The superposition and entanglement allow for the evaluation of all possible moves in parallel when the quantum gates are applied.

8.3 Measurement

Measure the qubits to collapse the superposition to a specific state, which gives the result of the move search. Mathematically, the state of the system after applying the Hadamard gates to each qubit is given by:

$$|\Psi\rangle = H^{\otimes n \times \lceil \log_2(m) \rceil} |0\rangle^{\otimes n \times \lceil \log_2(m) \rceil}$$

After applying the quantum gates designed for the Mastermind algorithm, the state becomes:

$$|\Phi\rangle = U|\Psi\rangle$$

Where (U) represents the unitary transformation corresponding to the sequence of quantum gates applied. Finally, the measurement collapses the state ($|\Phi\rangle$) to one of the basis states, which corresponds to the outcome of the move search. This mathematical framework outlines how quantum gates can be applied to superposition states to achieve parallelization in the move search of the Mastermind game. The actual quantum gates and their sequence would depend on the specific rules and logic of the Mastermind algorithm being simulated. The goal is to use quantum computing's inherent parallelism to solve the game more efficiently than classical methods.