**Comprehensive Analysis of Bulls and Cows Game:**

**A Dual-Mode Algorithmic Approach**

*Swetha Gendlur Nagarajan*

**Code Video Link:** [**https://drive.google.com/file/d/1pg9IDG8kTYSWut-GFM25e0g1f1hmn5OG/view?usp=sharing**](https://drive.google.com/file/d/1pg9IDG8kTYSWut-GFM25e0g1f1hmn5OG/view?usp=sharing)

**Code Link:** [**https://drive.google.com/file/d/1kIoKOaCpc0ldSCvOD1whWasOw673YrN9/view?usp=sharing**](https://drive.google.com/file/d/1kIoKOaCpc0ldSCvOD1whWasOw673YrN9/view?usp=sharing)

**GitHub Link:**[**https://github.com/swetha-gn/Bulls\_Cows\_Game\_An\_Entropy\_Based**](https://github.com/swetha-gn/Bulls_Cows_Game_An_Entropy_Based)

**Drive link:** [**https://drive.google.com/drive/folders/1Ml2LSCPutAInCEp3PPh0K711a\_XANNDM?usp=sharing**](https://drive.google.com/drive/folders/1Ml2LSCPutAInCEp3PPh0K711a_XANNDM?usp=sharing)

**Presentation Video link:**[**https://drive.google.com/file/d/1LzK8sdqCyajeSF\_Yuub2cdqhBrOh9bck/view?usp=sharing**](https://drive.google.com/file/d/1LzK8sdqCyajeSF_Yuub2cdqhBrOh9bck/view?usp=sharing)

**PPTlink:**[**https://docs.google.com/presentation/d/1z6x1JUL8DMhzNwWKdx0eSbLjL4HeY9Km/edit?usp=sharing&ouid=100201292702921013375&rtpof=true&sd=true**](https://docs.google.com/presentation/d/1z6x1JUL8DMhzNwWKdx0eSbLjL4HeY9Km/edit?usp=sharing&ouid=100201292702921013375&rtpof=true&sd=true)

**1. Introduction**

The Bulls and Cows game implementation presents a unique computational approach to a classic number guessing game, offering two distinct gameplay modes that showcase different algorithmic strategies and computational thinking. In this process, you will attempt to guess the computer's secret number. The computer will display the number of "bulls" and "cows" related to your guesses, providing information about the current entropy and the current possibility set. You can analyze this data to inform your next move. Based on your subsequent guess, the computer will provide additional options, continuing this process until you identify the optimal move.

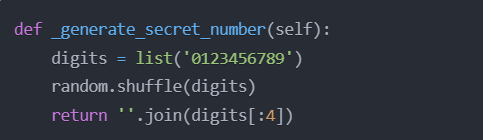
This method is designed to help you find the most optimal solution while minimizing the number of guesses required. The algorithm significantly reduces the number of guesses needed.

**2. Player Guessing Mode: Detailed Analysis**

**2.1 Core Mechanism**

The `PlayerGuessingMode` class implements a scenario where the player attempts to guess a computer-generated secret number.

**2.1.1 Secret Number Generation**

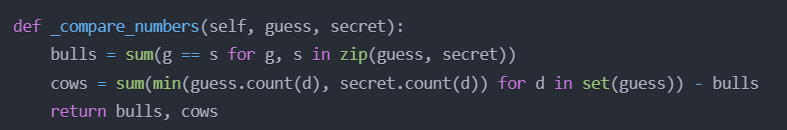


**Key characteristics:**

- Guarantees 4 unique digits

- Total possible combinations: 5,040 (10 P 4)

**2.2 Guess Evaluation Algorithm**



**Algorithmic Breakdown:**

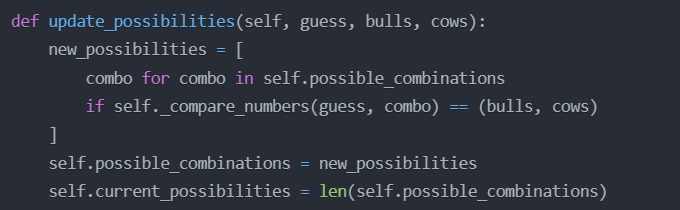
The cows variable in the \_compare\_numbers() function calculates the number of correct digits in the wrong position between the guess and the secret number.

Here's how the calculation works:

1. guess.count(d) counts the number of occurrences of each digit d in the guess.
2. secret.count(d) counts the number of occurrences of each digit d in the secret number.
3. min(guess.count(d), secret.count(d)) finds the minimum of those two counts for each digit d.
4. Summing those minimum counts for all digits d in the set(guess) (the unique digits in the guess) gives the total number of correct digits that are in the wrong position, which is the number of "cows".
5. Subtracting the number of "bulls" (correct digits in the correct position) gives the final "cows" count.

So in our example, if the guess is "1234" and the secret number is some other 4-digit number, the "cows" value would represent the number of digits from the guess that are present in the secret number, but in the wrong positions.

**2.3 Possibility Reduction Strategy**



**Optimization Techniques:**

This function updates the list of possible combinations based on the feedback (bulls and cows) received for the previous guess. It narrows down the search space by only keeping the combinations that match the provided feedback, making the guessing process more efficient.

If we have:

* Secret: "1234"
* Guess:3456
* Feedback: 0 bull, 2 cows
* Current possibilities: ["1234", "1325", "4231", "5678"]

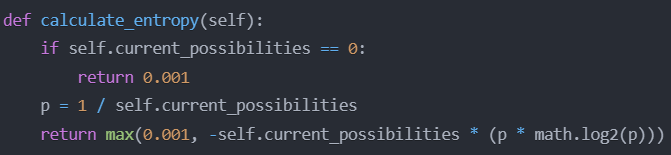
The method would:

After this filtering, self.possible\_combinations will be significantly reduced. It will only contain numbers that:

* Have exactly two of the digits 3, 4, 5, or 6
* Do not have any of these digits in the same position as in 4356

This algorithmic approach ensures that all remaining possibilities are consistent with all previous guesses and their feedback, making it a critical component of the game's logic system.

**2.4 Entropy Tracking**



* The entropy calculation is based on the Shannon entropy formula, which is a fundamental concept in information theory. It measures the uncertainty or information content of a probability distribution.
* In this case, the entropy is calculated based on the current number of possible combinations (self.current\_possibilities) and the probability p of any single combination being correct.
* The higher the entropy, the more uncertainty there is in the current state of the game. Conversely, lower entropy indicates that the game state is more certain and the computer/player is closer to finding the correct number.
* By tracking the entropy, the game can make informed decisions about the best guessing strategy to minimize the number of attempts required to find the secret number.

**Note: If the number of possibilities is 0, the function returns 0.001 instead of 0. This is likely a safeguard to prevent division by 0 errors later in the calculation.**

**2.5 Get Answer**

* Provides an option to reveal the secret number, ending the game.
* This implementation allows players to make guesses, receive feedback, and see how the space of possibilities narrows down with each guess, mirroring the computer's guessing process in a player-controlled environment**.**

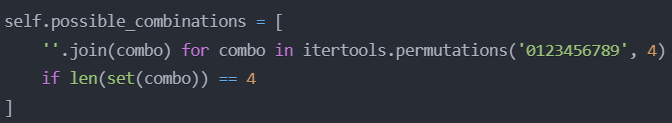
**3. Computer Guessing Mode in Bulls and Cows**

**3.1 Introduction**

The Bulls and Cows game implementation features a "Computer Guessing Mode" where the computer attempts to guess the player's secret number. This mode showcases a sophisticated algorithmic approach to systematically narrowing down the search space and converging on the correct solution.

**3.2 Core Functionality**

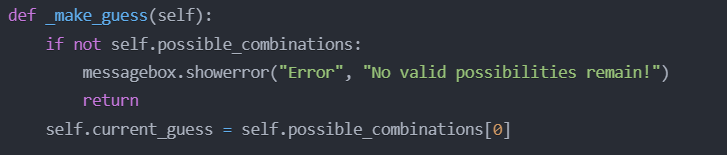
**3.2.1 Generating the Initial Possibility Space**



The game starts by generating all possible 4-digit combinations with unique digits. This is accomplished using the `itertools.permutations()` function, which generates all permutations of the digits 0-9 taken 4 at a time. The list comprehension then filters out any combinations with repeated digits, ensuring that each 4-digit number is unique.

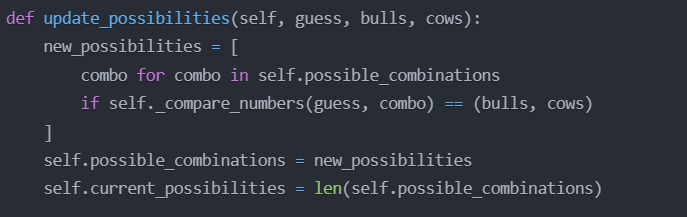
The initial size of the possibility space is 5,040 (10P4), representing the total number of unique 4-digit combinations.

**3.2.2 Making the First Guess**



To make the first guess, the computer simply **selects the first combination** from the `self.possible\_combinations` list. This represents a strategic decision to start with the most "general" or "typical" guess, as the first guess can significantly impact the subsequent reduction of the search space.

**3.2.3 Updating the Possibility Space**

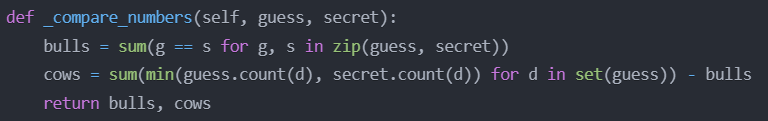


The key to the computer's guessing strategy lies in the `update\_possibilities()` method. This function takes the previous guess, the number of bulls (correct digits in the correct position), and the number of cows (correct digits in the wrong position), and then updates the list of possible combinations accordingly.

The method uses a list comprehension to filter the `self.possible\_combinations` list, keeping only the combinations that match the provided bulls and cows feedback for the previous guess. This narrows down the search space to only those combinations that are consistent with the player's response.

By continuously updating the possibility space, the computer can systematically eliminate invalid combinations and focus on the most promising candidates, ultimately converging on the correct solution.

**3.2.4 Comparing Numbers**



The `\_compare\_numbers()` method is a crucial component in the guessing process. It takes a guess and a secret number, and calculates the number of bulls (correct digits in the correct position) and cows (correct digits in the wrong position).

**The algorithm works as follows:**

1. The number of bulls is calculated by iterating through the digits of the guess and secret number, and counting the number of exact matches.

2. The number of cows is calculated by first counting the occurrences of each unique digit in both the guess and the secret number, and then taking the minimum of those counts for each digit. The sum of these minima represents the number of correct digits in the wrong position, which is the number of cows. Finally, the number of bulls is subtracted to avoid double-counting.

This comparison mechanism is used both in the `update\_possibilities()` method to filter the possibility space, and in the `\_evaluate\_guess()` method (not shown in the provided code) to provide feedback to the player.

**3.3 Algorithmic Analysis**

3.3.1 Implicit Minimax Strategy

The computer's guessing strategy can be viewed as an implicit implementation of a minimax algorithm. By systematically reducing the search space and making the "best" guess (the first remaining possibility), the computer is effectively minimizing the maximum number of guesses required to find the secret number.

This approach maximizes the information gained from each guess, ensuring that the computer converges on the solution as quickly as possible.

**4. Advantages**

1. Efficient Search Space Reduction: The ability to dynamically update the possibility space based on feedback allows the computer to quickly narrow down the search and focus on the most promising combinations.

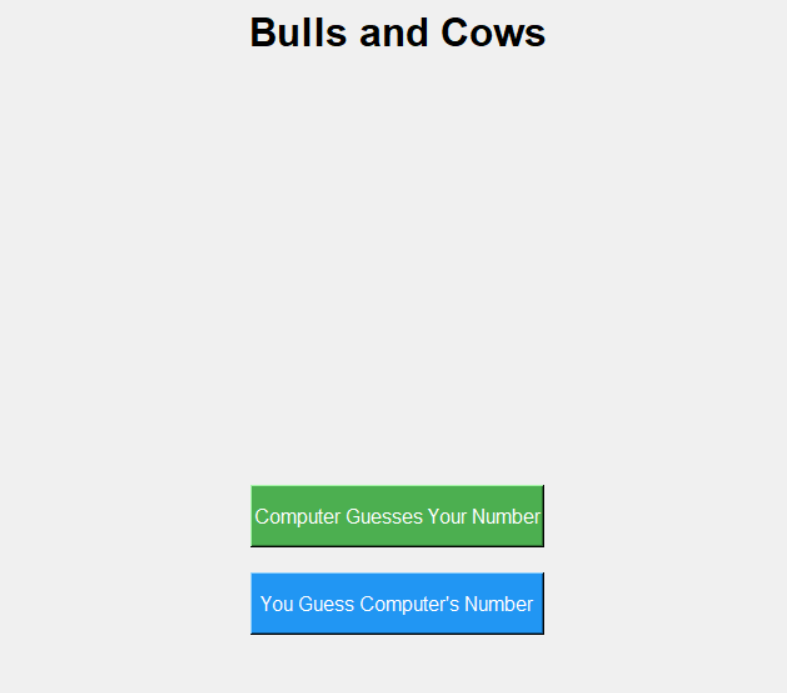
2. Computational Simplicity: The core algorithms, such as number comparison and entropy calculation, are relatively straightforward and efficient, making the implementation feasible even for modest computational resources.

3. Probabilistic Foundations: The guessing strategy is built on principles of information theory and probability, providing a strong theoretical foundation for the decision-making process.

**5. Conclusion**

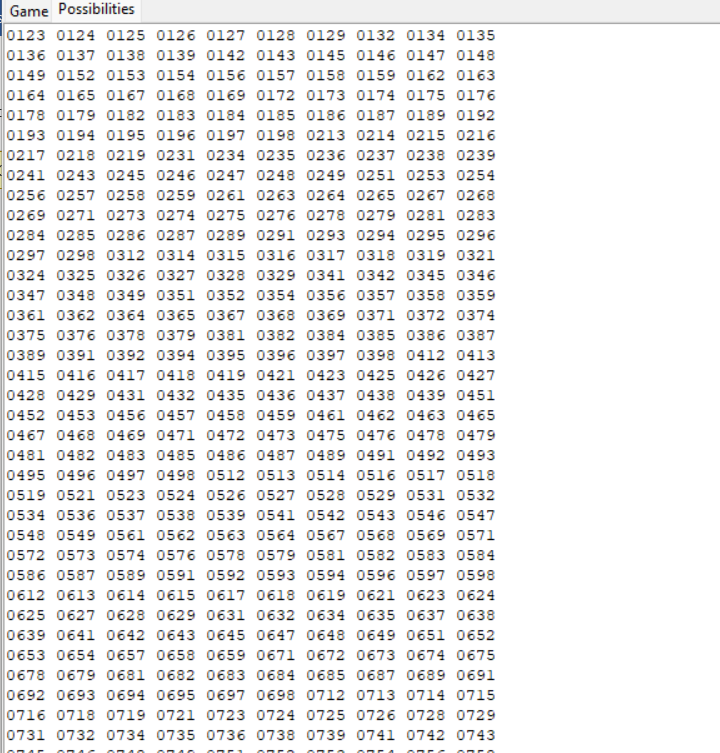
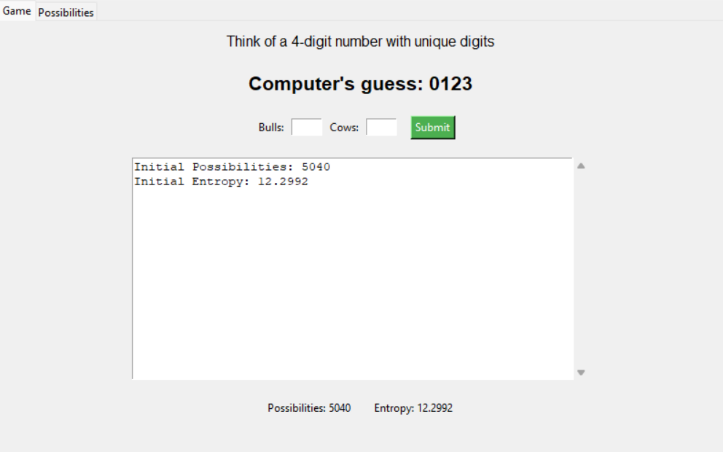
The Bulls and Cows implementation represents a sophisticated intersection of combinatorics, information theory, and interactive computational modeling. By providing two distinct gameplay modes, the application demonstrates the versatility of algorithmic thinking and computational problem-solving strategies.

1. **Code Output:**

****

**Fig 6.1: Tkinter library is used for GUI**

 You will encounter two options. In the first option, the computer will try to guess your number, and in the second, you will attempt to guess the number the computer has chosen.



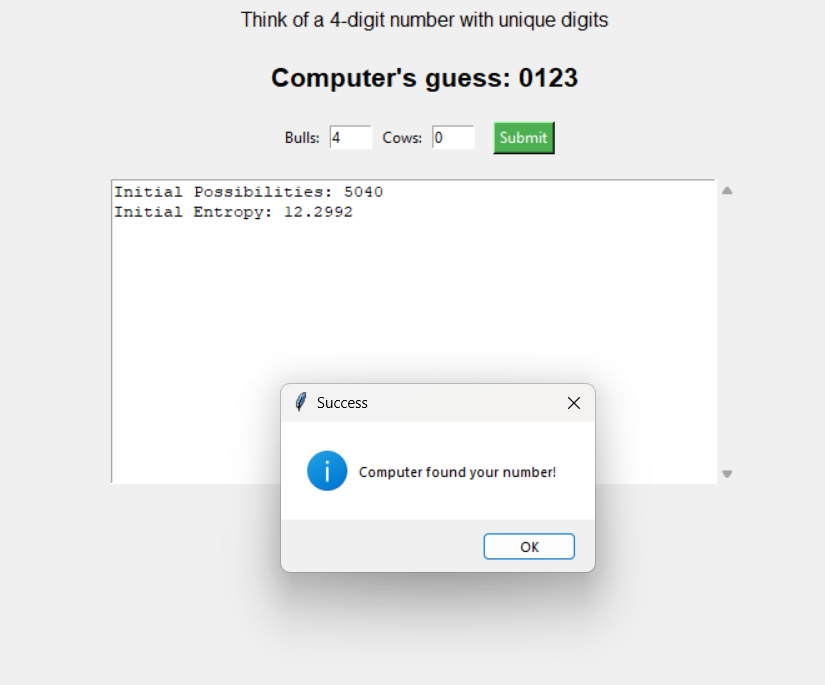
**Fig 6.2 Computer Guess Mode**

* The "Game" tab displays the current guess, input fields for bulls and cows, and a log of previous guesses with statistics.
* The "Possibilities" tab shows all remaining possible combinations.

****

**Fig 6.3 Edge cases**

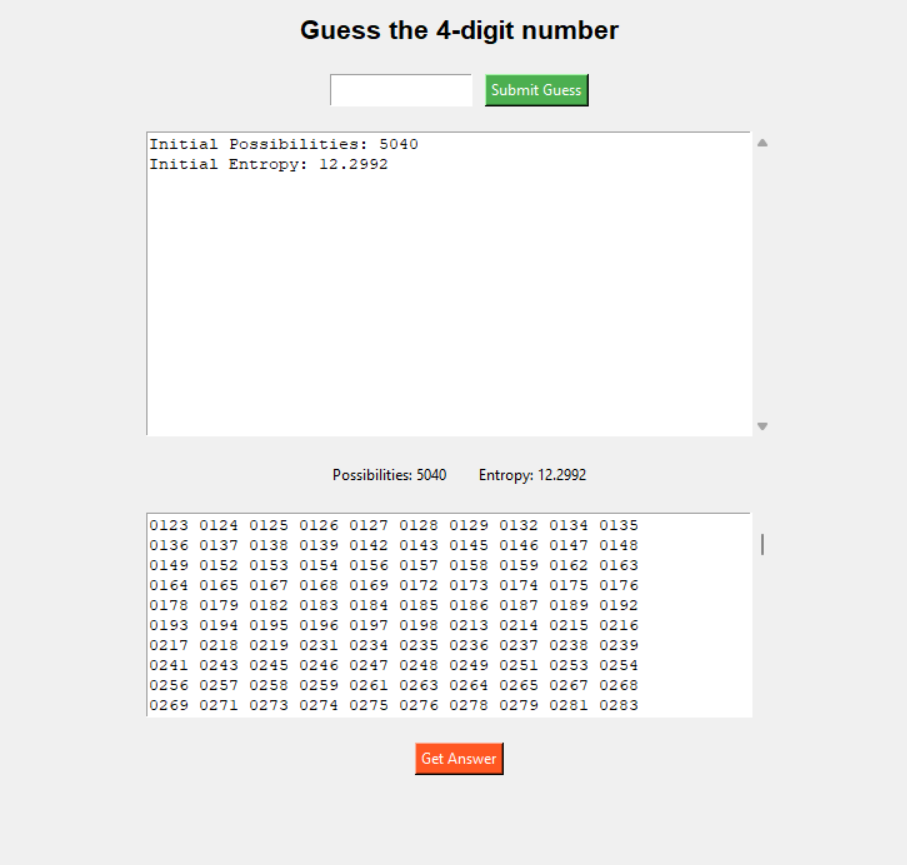
* If the Bulls and Cows count are more than 4, the prompt appears

****

**Fig 6.4 Output Prompt**

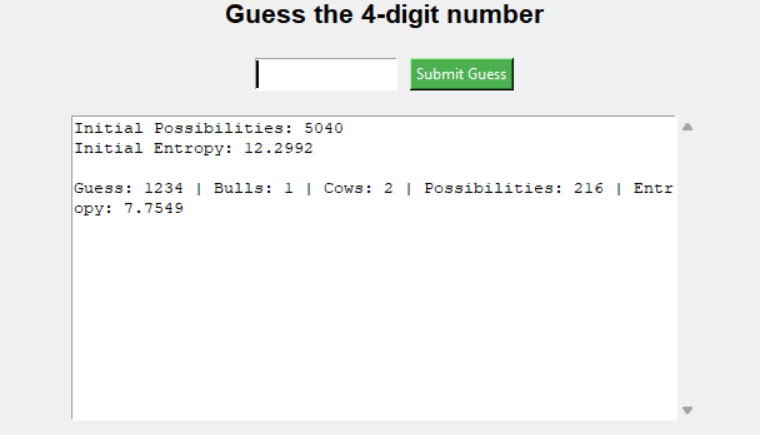
* If the Bulls are 4 and Cows are 0, then the game ends.

**Player Guessing Mode:**

****

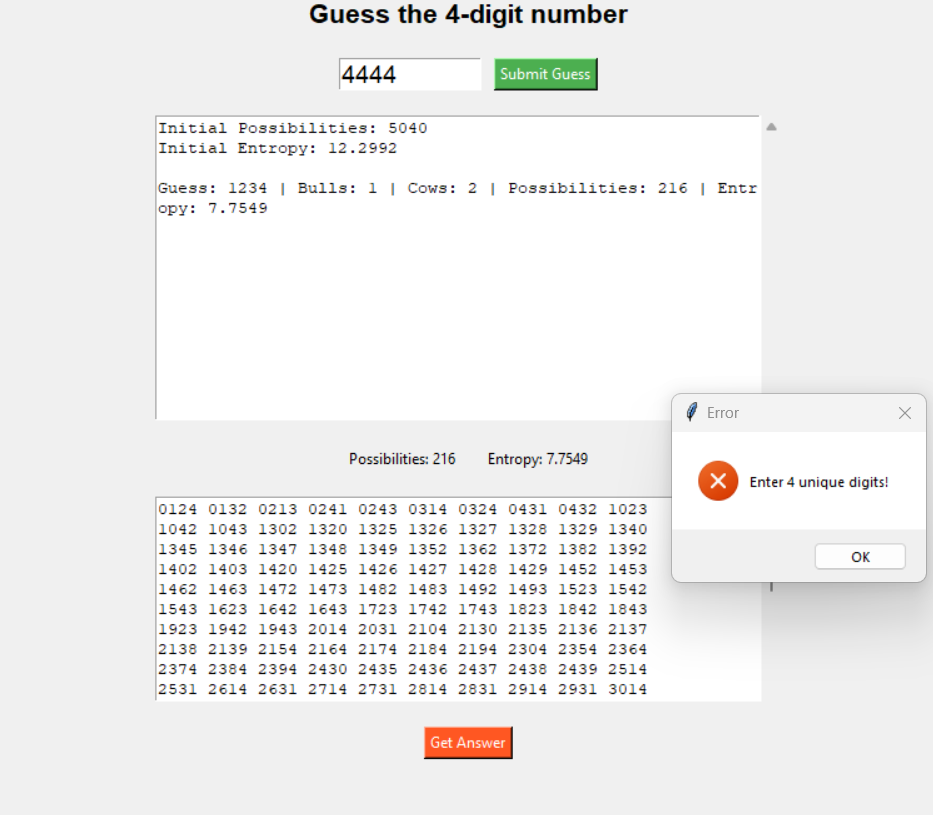
**Fig 6.5 GUI for Player guessing mode**

* Contains Possibilities set and a get answer button to know the answer

****

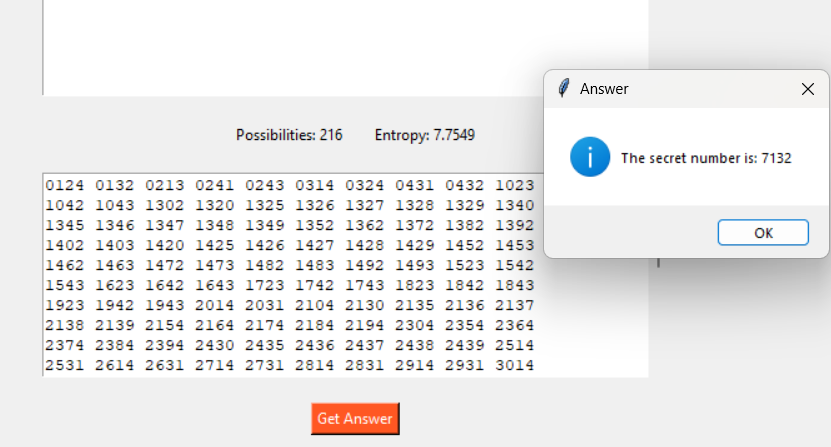
**Fig 6.6 Each step Calculation**

* Calculates and displays the entropy for each user guess

****

**Fig 6.7 Invalid Input**

* If the user enters repeated numbers then a dialog box appears, asks the suer to enter again

****

**Fig 6.8 Get Answer Functionality**