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

Code Link: https://drive.google.com/file/d/1kIoKOaCpc0ldSCvOD1whWasOw673YrN9/view?usp=sharing

GitHub Link:https://github.com/swetha-gn/Bulls_Cows_Game_An_Entropy_Based

Drive link: https://drive.google.com/drive/folders/1Ml2LSCPutAInCEp3PPh0K711a_XANNDM?usp=sharing

Presentation Video link: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

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

```python
def _generate_secret_number(self):
    digits = list('0123456789')
    random.shuffle(digits)
    return ''.join(digits[:4])
```

Key characteristics:

- Guarantees 4 unique digits

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

2.2 Guess Evaluation Algorithm

```python
def _compare_numbers(self, guess, secret):
    bulls = sum(g == s for g, s in zip(guess, secret))
    cows = sum(min(guess.count(d), secret.count(d)) for d in set(guess)) - bulls
    return bulls, cows
```

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

```python
def update_possibilities(self, guess, bulls, cows):
    new_possibilities = [
        combo for combo in self.possible_combinations
        if self._compare_numbers(guess, combo) == (bulls, cows)
    ]
    self.possible_combinations = new_possibilities
    self.current_possibilities = len(self.possible_combinations)
```

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"]

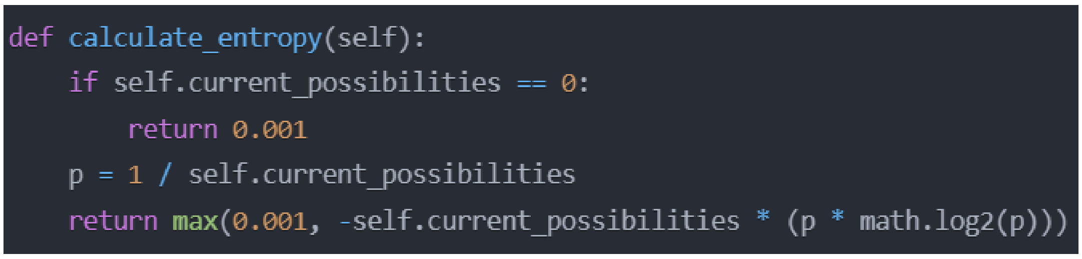
The method would:

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



```python
def calculate_entropy(self):
    if self.current_possibilities == 0:
        return 0.001
    p = 1 / self.current_possibilities
    return max(0.001, -self.current_possibilities * (p * math.log2(p)))
```

- 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.

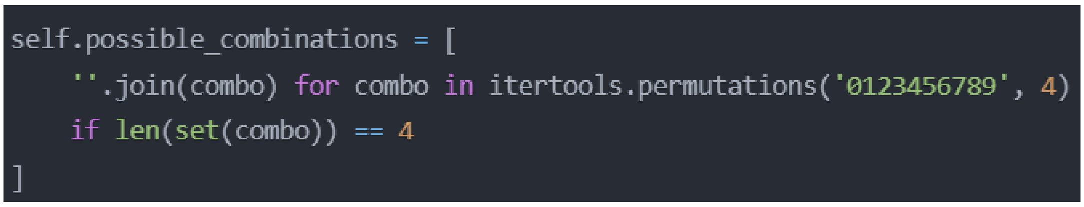
3. Computer Guessing Mode in Bulls and Cows

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



```python
self.possible_combinations = [
    ''.join(combo) for combo in itertools.permutations('0123456789', 4)
    if len(set(combo)) == 4
]
```

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

```python
def _make_guess(self):
    if not self.possible_combinations:
        messagebox.showerror("Error", "No valid possibilities remain!")
        return
    self.current_guess = self.possible_combinations[0]
```

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

```python
def update_possibilities(self, guess, bulls, cows):
    new_possibilities = [
        combo for combo in self.possible_combinations
        if self._compare_numbers(guess, combo) == (bulls, cows)
    ]
    self.possible_combinations = new_possibilities
    self.current_possibilities = len(self.possible_combinations)
```

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

```python
def _compare_numbers(self, guess, secret):
    bulls = sum(g == s for g, s in zip(guess, secret))
    cows = sum(min(guess.count(d), secret.count(d)) for d in set(guess)) - bulls
    return bulls, cows
```

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.

6. Code Output:

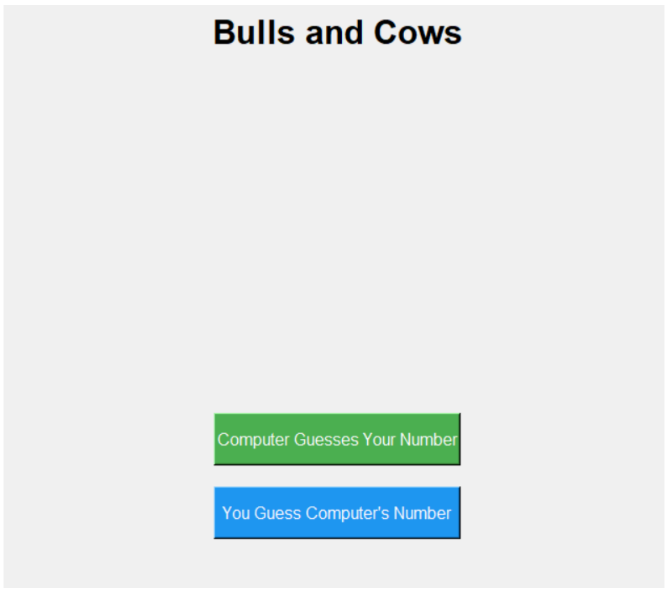
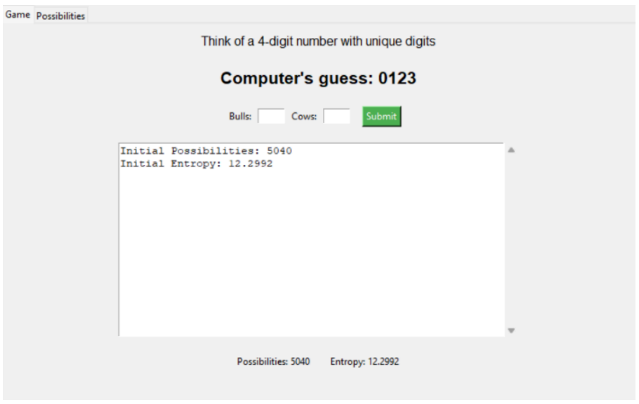


Fig 6.1: Tkinter library is used for GUI

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.



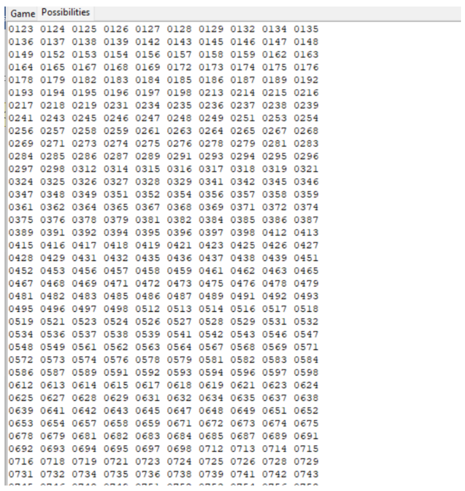
Think of a 4-digit number with unique digits

Computer's guess: 0123

Bulls: Cows:

```
Initial Possibilities: 5040
Initial Entropy: 12.2992
```

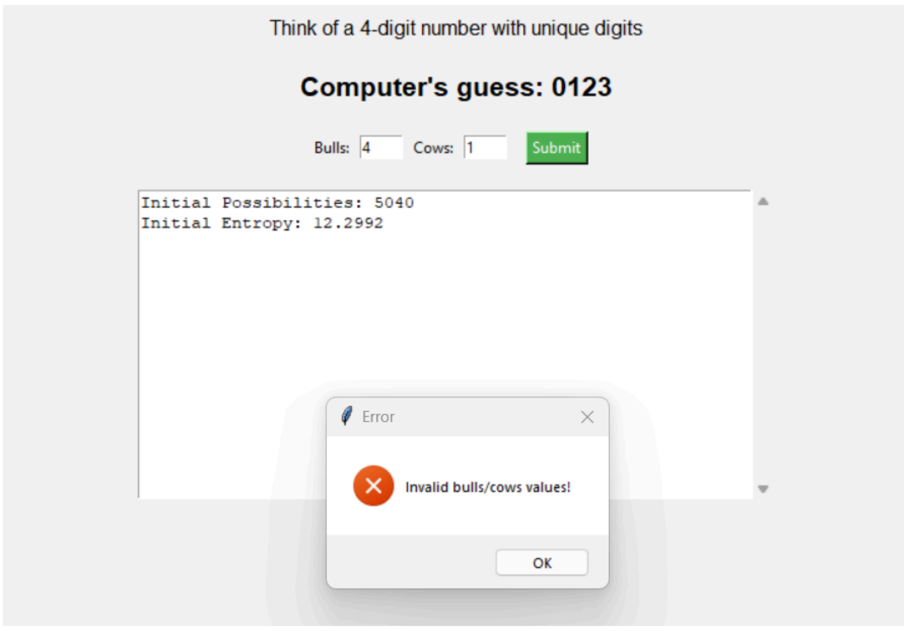
Possibilities: 5040   Entropy: 12.2992



```
0123 0124 0125 0126 0127 0128 0129 0132 0134 0135
0136 0137 0138 0139 0142 0143 0145 0146 0147 0148
0149 0152 0153 0154 0156 0157 0158 0159 0162 0163
0164 0165 0167 0168 0169 0172 0173 0174 0175 0176
0178 0179 0182 0183 0184 0185 0186 0187 0189 0192
0193 0194 0195 0196 0197 0198 0213 0214 0215 0216
0217 0218 0219 0231 0234 0235 0236 0237 0238 0239
0241 0243 0245 0246 0247 0248 0249 0251 0253 0254
0256 0257 0258 0259 0261 0263 0264 0265 0267 0268
0269 0271 0273 0274 0275 0276 0278 0279 0281 0283
0284 0285 0286 0287 0289 0291 0293 0294 0295 0296
0297 0298 0312 0314 0315 0316 0317 0318 0319 0321
0324 0325 0326 0327 0328 0329 0341 0342 0345 0346
0347 0348 0349 0351 0352 0354 0356 0357 0358 0359
0361 0362 0364 0365 0367 0368 0369 0371 0372 0374
0375 0376 0378 0379 0381 0382 0384 0385 0386 0387
0389 0391 0392 0394 0395 0396 0397 0398 0412 0413
0415 0416 0417 0418 0419 0421 0423 0425 0426 0427
0428 0429 0431 0432 0435 0436 0437 0438 0439 0451
0452 0453 0456 0457 0458 0459 0461 0462 0463 0465
0467 0468 0469 0471 0472 0473 0475 0476 0478 0479
0481 0482 0483 0485 0486 0487 0489 0491 0492 0493
0495 0496 0497 0498 0512 0513 0514 0516 0517 0518
0519 0521 0523 0524 0526 0527 0528 0529 0531 0532
0534 0536 0537 0538 0539 0541 0542 0543 0546 0547
0548 0549 0561 0562 0563 0564 0567 0568 0569 0571
0572 0573 0574 0576 0578 0579 0581 0582 0583 0584
0586 0587 0589 0591 0592 0593 0594 0596 0597 0598
0612 0613 0614 0615 0617 0618 0619 0621 0623 0624
0625 0627 0628 0629 0631 0632 0634 0635 0637 0638
0639 0641 0642 0643 0645 0647 0648 0649 0651 0652
0653 0654 0657 0658 0659 0671 0672 0673 0674 0675
0678 0679 0681 0682 0683 0684 0685 0687 0689 0691
0692 0693 0694 0695 0697 0698 0712 0713 0714 0715
0716 0718 0719 0721 0723 0724 0725 0726 0728 0729
0731 0732 0734 0735 0736 0738 0739 0741 0742 0743
```

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.


Think of a 4-digit number with unique digits

Computer's guess: 0123

Bulls: 4 Cows: 1 Submit

```
Initial Possibilities: 5040
Initial Entropy: 12.2992
```

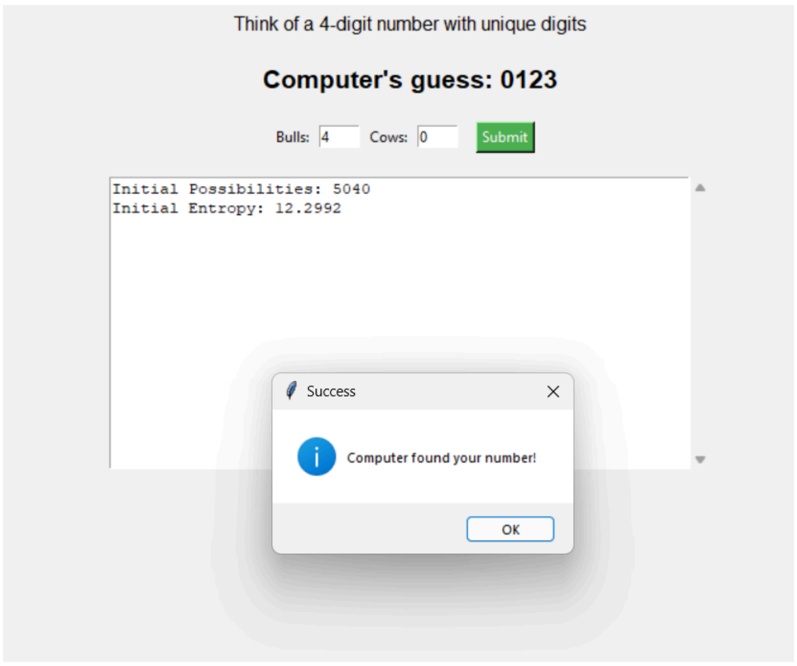
Error

Invalid bulls/cows values!

OK

Fig 6.3 Edge cases

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


Think of a 4-digit number with unique digits

Computer's guess: 0123

Bulls: 4 Cows: 0 Submit

```
Initial Possibilities: 5040
Initial Entropy: 12.2992
```

Success

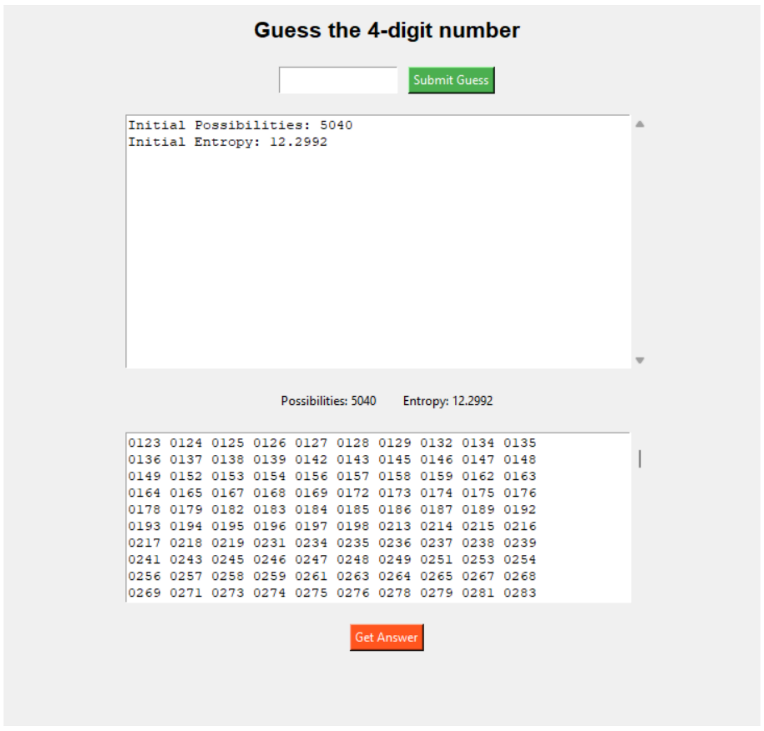
Computer found your number!

OK

Fig 6.4 Output Prompt

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

Player Guessing Mode:



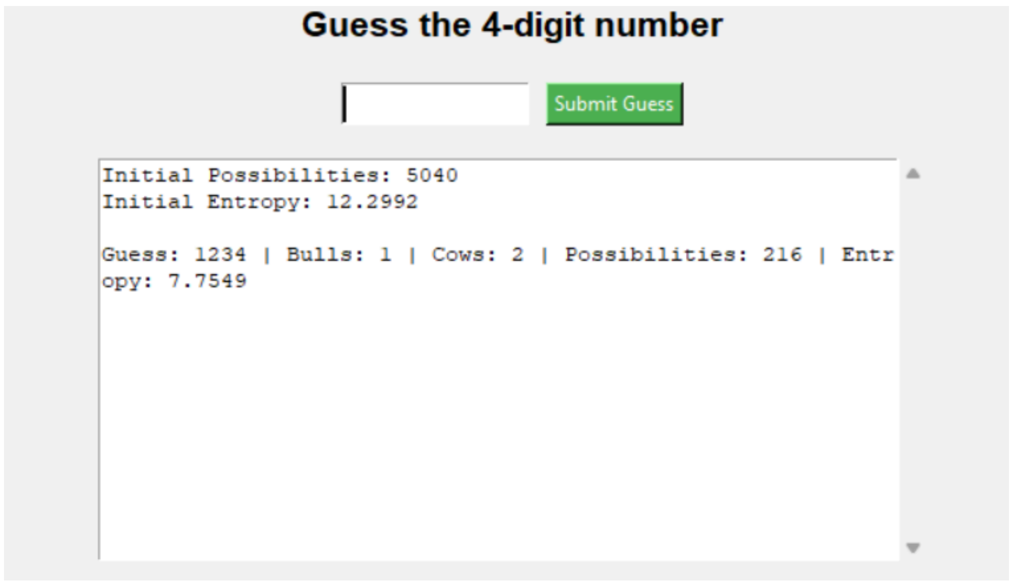
```
Initial Possibilities: 5040
Initial Entropy: 12.2992
```

Possibilities: 5040 Entropy: 12.2992

```
0123 0124 0125 0126 0127 0128 0129 0132 0134 0135
0136 0137 0138 0139 0142 0143 0145 0146 0147 0148
0149 0152 0153 0154 0156 0157 0158 0159 0162 0163
0164 0165 0167 0168 0169 0172 0173 0174 0175 0176
0178 0179 0182 0183 0184 0185 0186 0187 0189 0192
0193 0194 0195 0196 0197 0198 0213 0214 0215 0216
0217 0218 0219 0231 0234 0235 0236 0237 0238 0239
0241 0243 0245 0246 0247 0248 0249 0251 0253 0254
0256 0257 0258 0259 0261 0263 0264 0265 0267 0268
0269 0271 0273 0274 0275 0276 0278 0279 0281 0283
```

Fig 6.5 GUI for Player guessing mode

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

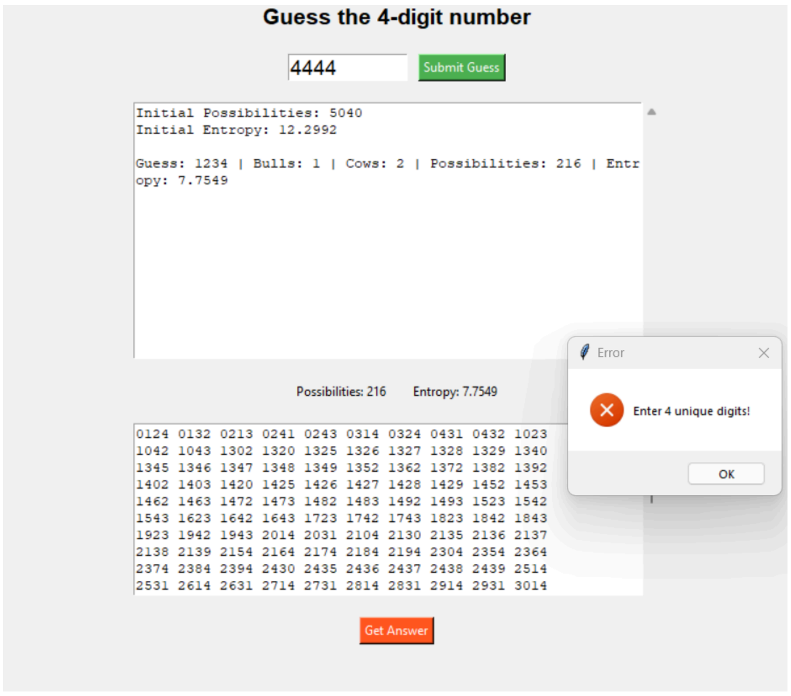


```
Initial Possibilities: 5040
Initial Entropy: 12.2992

Guess: 1234 | Bulls: 1 | Cows: 2 | Possibilities: 216 | Entr
opy: 7.7549
```

Fig 6.6 Each step Calculation

- Calculates and displays the entropy for each user guess



```
Initial Possibilities: 5040
Initial Entropy: 12.2992

Guess: 1234 | Bulls: 1 | Cows: 2 | Possibilities: 216 | Entr
opy: 7.7549
```

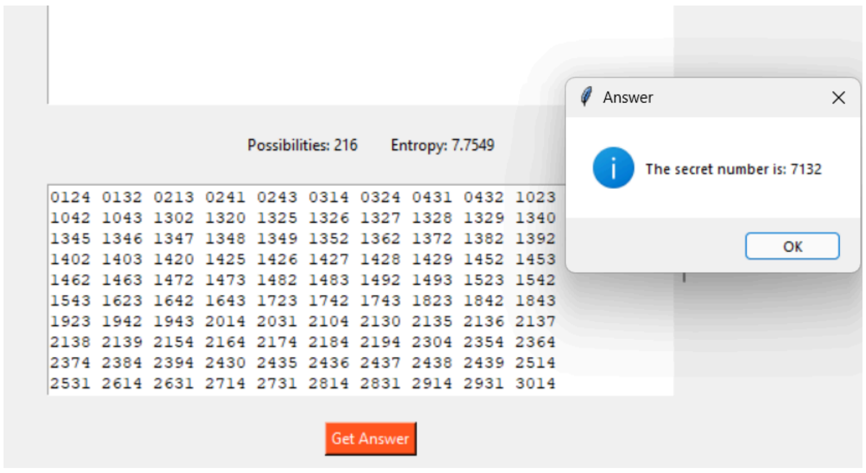
Enter 4 unique digits!

Possibilities: 216   Entropy: 7.7549

```
0124 0132 0213 0241 0243 0314 0324 0431 0432 1023
1042 1043 1302 1320 1325 1326 1327 1328 1329 1340
1345 1346 1347 1348 1349 1352 1362 1372 1382 1392
1402 1403 1420 1425 1426 1427 1428 1429 1452 1453
1462 1463 1472 1473 1482 1483 1492 1493 1523 1542
1543 1623 1642 1643 1723 1742 1743 1823 1842 1843
1923 1942 1943 2014 2031 2104 2130 2135 2136 2137
2138 2139 2154 2164 2174 2184 2194 2304 2354 2364
2374 2384 2394 2430 2435 2436 2437 2438 2439 2514
2531 2614 2631 2714 2731 2814 2831 2914 2931 3014
```

Fig 6.7 Invalid Input

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



Possibilities: 216   Entropy: 7.7549

The secret number is: 7132

```
0124 0132 0213 0241 0243 0314 0324 0431 0432 1023
1042 1043 1302 1320 1325 1326 1327 1328 1329 1340
1345 1346 1347 1348 1349 1352 1362 1372 1382 1392
1402 1403 1420 1425 1426 1427 1428 1429 1452 1453
1462 1463 1472 1473 1482 1483 1492 1493 1523 1542
1543 1623 1642 1643 1723 1742 1743 1823 1842 1843
1923 1942 1943 2014 2031 2104 2130 2135 2136 2137
2138 2139 2154 2164 2174 2184 2194 2304 2354 2364
2374 2384 2394 2430 2435 2436 2437 2438 2439 2514
2531 2614 2631 2714 2731 2814 2831 2914 2931 3014
```

Fig 6.8 Get Answer Functionality