z

AI BASED NUMBER GUESSING GAME

Name	Roll No.	Branch
Vinay Rathore	202401100300280 (2428CSEAI1779)	CSE(AI)

KIET GROUP OF INSTITUTIONS, GHAZIABAD, UTTAR PRADESH

INTRODUCTION

In this project, we are designing an **AI-powered number guessing game** where an intelligent algorithm efficiently finds a secret number chosen by the user. Instead of randomly guessing numbers, the AI applies a **logical and structured approach** to minimize the number of guesses.

This problem is an excellent way to understand how AI can **make decisions, eliminate possibilities, and reach a conclusion efficiently**—a concept widely used in **search algorithms, machine learning, and AI-driven decision-making systems**.

UNDERSTANDING THE PROBLEM

The game consists of two participants:

- **The User:** Thinks of a secret number between 1 and 100 and provides feedback after each guess.
- **The AI:** Tries to guess the number as quickly as possible based on the user's feedback.

The AI follows a structured process:

1. **Make an initial guess.**
2. **Receive feedback** from the user on whether the guess is:
    - **Correct.**
    - **Too High** (i.e., the guessed number is greater than the secret number).

    - **Too Low** (i.e., the guessed number is smaller than the secret number).
3. **Adjust the range** based on the feedback.
4. **Repeat the process** until the correct number is found.

The goal is to **guess the correct number in the least number of attempts** by using a systematic approach rather than random guesses.

CHALLENGES IN THE PROBLEM

A **random guessing approach** would involve the AI picking numbers arbitrarily between 1 and 100 until it finds the correct one. This is **highly inefficient** because:

- The AI might take a long time to guess correctly.
- In the worst case, the AI could require **up to 100 guesses**.

For example, if the user's secret number is **75**, and the AI guesses randomly, it could take **dozens of attempts** before reaching the correct number.

Instead, we need a more **optimized and intelligent approach** that significantly reduces the number of guesses required.

METHADOLOGY

To achieve an efficient guessing strategy, the AI utilizes the **Binary Search Algorithm**, which follows a structured decision-making process. This ensures that the number is found in the fewest possible attempts.

STEPS

1. **Define the Problem:**
    - **The user selects a secret number within a given range (e.g., 1 to 100).**
    - **The AI must guess this number based on user feedback.**
2. **Choosing an Efficient Algorithm:**
    - **A random guessing approach is inefficient as it may take up to 100 attempts.**
    - **Instead, Binary Search is used, which significantly reduces the number of guesses by half at each step.**
3. **Binary Search Implementation:**
    - **The AI starts with a range (low = 1, high = 100).**
    - **It selects the midpoint as the first guess:**
    - **Based on user feedback:**
        - **If too low, adjust low = guess + 1.**
        - **If too high, adjust high = guess - 1.**
        - **If correct, the game ends.**

- **This process repeats until the AI finds the number.**

OPTIMISATION AND EDGE CASES

- **Handling incorrect inputs:** Ensures users provide valid feedback.
- **Expanding range flexibility:** Allows users to define custom ranges.
- **Tracking past guesses:** Prevents redundant calculations.

CODE

```
def play_game():
    """Efficient AI-based number guessing game with user input."""

    # Step 1: Ask the user to enter a secret number
    while True:
        try:
            secret_number = int(input("Enter a secret number between 1 and 100: "))
            if 1 <= secret_number <= 100:
                break
            else:
                print("Please enter a valid number between 1 and 100.")
        except ValueError:
            print("Invalid input! Please enter an integer.")

    print("\nThe AI will now try to guess your number!")

    # Step 2: Initialize search bounds and attempts counter
    lower_bound, upper_bound = 1, 100
    attempts = 0

    # Step 3: AI starts guessing using binary search logic
    while lower_bound <= upper_bound:
```

```
        guess = (lower_bound + upper_bound) // 2  # AI picks the midpoint
        attempts += 1
        remaining_options = upper_bound - lower_bound + 1  # Possible remaining numbers

        # Print the guess and remaining possible numbers
        print(f"AI guesses: {guess} (Remaining possible numbers: {remaining_options})", end=" ")

        if guess == secret_number:  # AI guessed correctly
            print("✅ Correct!")
            print(f"AI guessed your number {secret_number} in {attempts} attempts! 🎉")
            return  # End the game

        elif guess < secret_number:  # AI guessed too low
            print("-> Too low")
            lower_bound = guess + 1  # Adjust lower bound

        else:  # AI guessed too high
            print("-> Too high")
            upper_bound = guess - 1  # Adjust upper bound

# Run the game
```

play_game()

OUTPUT

```
Enter a secret number between 1 and 100: 64

The AI will now try to guess your number!
AI guesses: 50 (Remaining possible numbers: 100) -> Too low
AI guesses: 75 (Remaining possible numbers: 50) -> Too high
AI guesses: 62 (Remaining possible numbers: 24) -> Too low
AI guesses: 68 (Remaining possible numbers: 12) -> Too high
AI guesses: 65 (Remaining possible numbers: 5) -> Too high
AI guesses: 63 (Remaining possible numbers: 2) -> Too low
AI guesses: 64 (Remaining possible numbers: 1) ✅ Correct!
AI guessed your number 64 in 7 attempts! 🎉
```

```
Enter a secret number between 1 and 100: 91

The AI will now try to guess your number!
AI guesses: 50 (Remaining possible numbers: 100) -> Too low
AI guesses: 75 (Remaining possible numbers: 50) -> Too low
AI guesses: 88 (Remaining possible numbers: 25) -> Too low
AI guesses: 94 (Remaining possible numbers: 12) -> Too high
AI guesses: 91 (Remaining possible numbers: 5) ✅ Correct!
AI guessed your number 91 in 5 attempts! 🎉
```

CREDITS

ALGORITHM & INSPIRATION:

→Official Python Website: https://www.python.org/

→**Binary Search Algorithm** for efficient guessing

TOOLS AND TECHNOLOGIES USED:

→ **Programming Language:** Python

→ **Development Environment:** GOOGLE Colab notebook