Juego del 15

Trabajo Corte 1 - Computación 2

Nombre: Mariana Rodríguez Pérez

Fecha: Febrero 2025

El juego del quince es un rompecabezas deslizante que consiste en una cuadrícula de 4x4 donde se encuentran 15 fichas numeradas y una casilla vacía. El objetivo es mover las fichas, deslizando aquellas que se encuentran en la misma fila o columna que la casilla vacía, hasta ordenar los números de forma secuencial (de izquierda a derecha y de arriba hacia abajo), dejando la casilla vacía en la última posición.

El código se realizó en Python utilizando conceptos básicos de programación orientada a objetos. Se creó una clase llamada **Game** que:

- Inicializa el tablero con una matriz 4x4.
- Define métodos para visualizar el tablero usando **matplotlib**.
- Implementa funciones para encontrar la casilla vacía y validar si los movimientos (arriba, abajo, izquierda o derecha) son permitidos.
- Permite realizar los movimientos actualizando el estado del tablero y registra el historial de movimientos.

Esta estructura modular facilita la extensión y mejora del código, permitiendo una gestión clara de la lógica del juego y su visualización gráfica.

```
import random
import numpy as np
import matplotlib.pyplot as plt

class Game:
    """Clase principal para el juego del rompecabezas de 15."""

    GOAL = [[1, 2, 3, 4],
            [5, 6, 7, 8],
            [9, 10, 11, 12],
            [13, 14, 15, None]]

    def __init__(self, initial_board):
        self.board = initial_board
        self.move_stack = []  # Lista para historial de movimientos

    def show(self):
        """Mostrar visualización gráfica del tablero."""
        _, ax = plt.subplots()
        plt.imshow(np.array([[0 if x is None else x for x in fila] for
```

```
fila in self.board]),
                    cmap="YlGn", interpolation="nearest", vmin=0,
vmax=255)
        ax.set_xticks(np.arange(-0.5, 4, 1), minor=True)
        ax.set_yticks(np.arange(-0.5, 4, 1), minor=True)
        ax.grid(which="minor", color="black", linestyle="-",
linewidth=2)
        ax.set_xticks([])
        ax.set_yticks([])
        for i in range(4):
            for j in range(4):
                value = self.board[i][j]
                text = str(value) if value is not None else " "
                ax.text(j, i, text, ha='center', va='center',
color='black',
                        fontsize=16, fontweight='bold')
        plt.show()

    def is_game_win(self):
        """Verificar si se ha alcanzado el estado objetivo del
tablero."""
        return self.board == Game.GOAL

    def _find_empty(self):
        """Encontrar la posición del espacio vacío en el tablero."""
        for i in range(4):
            for j in range(4):
                if self.board[i][j] is None:
                    return i, j
        raise ValueError("No empty space found")

    def is_allowed_move_up(self):
        """Verificar si es posible mover el espacio vacío hacia
arriba."""
        i, j = self._find_empty()
        return i > 0

    def is_allowed_move_down(self):
        """Verificar si es posible mover el espacio vacío hacia
abajo."""
        i, j = self._find_empty()
        return i < 3

    def is_allowed_move_left(self):
        """Verificar si es posible mover el espacio vacío hacia la
izquierda."""
        i, j = self._find_empty()
        return j > 0

    def is_allowed_move_right(self):
```

```
        """Verificar si es posible mover el espacio vacío hacia la
derecha."""
        i, j = self._find_empty()
        return j < 3

    def move_up(self):
        """Realizar movimiento hacia arriba si es permitido."""
        if self.is_allowed_move_up():
            i, j = self._find_empty()
            self.board[i][j], self.board[i-1][j] = self.board[i-1][j],
self.board[i][j]
            self.move_stack.append((i-1, j))  # Registrar movimiento
en la pila

    def move_down(self):
        """Realizar movimiento hacia abajo si es permitido."""
        if self.is_allowed_move_down():
            i, j = self._find_empty()
            self.board[i][j], self.board[i+1][j] = self.board[i+1][j],
self.board[i][j]
            self.move_stack.append((i+1, j))  # Registrar movimiento
en la pila

    def move_left(self):
        """Realizar movimiento hacia la izquierda si es permitido."""
        if self.is_allowed_move_left():
            i, j = self._find_empty()
            self.board[i][j], self.board[i][j-1] = self.board[i][j-1],
self.board[i][j]
            self.move_stack.append((i, j-1))  # Registrar movimiento
en la pila

    def move_right(self):
        """Realizar movimiento hacia la derecha si es permitido."""
        if self.is_allowed_move_right():
            i, j = self._find_empty()
            self.board[i][j], self.board[i][j+1] = self.board[i][j+1],
self.board[i][j]
            self.move_stack.append((i, j+1))  # Registrar movimiento
en la pila

    def undo_last_move(self):
        """Deshacer el último movimiento realizado."""
        if self.move_stack:
            last_move = self.move_stack.pop()
            i, j = self._find_empty()
            self.board[i][j], self.board[last_move[0]][last_move[1]] =
self.board[last_move[0]][last_move[1]], self.board[i][j]

    def next_allowed_moves(self):
```

```
        """Obtener los movimientos permitidos y almacenarlos en una
lista."""
        moves = []

        empty_i, empty_j = self._find_empty()

        # Simular movimiento hacia arriba
        if self.is_allowed_move_up():
            new_board = [fila[:] for fila in self.board]
            new_board[empty_i][empty_j], new_board[empty_i-1][empty_j]
= new_board[empty_i-1][empty_j], new_board[empty_i][empty_j]
            moves.append(new_board)

        # Simular movimiento hacia abajo
        if self.is_allowed_move_down():
            new_board = [fila[:] for fila in self.board]
            new_board[empty_i][empty_j], new_board[empty_i+1][empty_j]
= new_board[empty_i+1][empty_j], new_board[empty_i][empty_j]
            moves.append(new_board)

        # Simular movimiento hacia la izquierda
        if self.is_allowed_move_left():
            new_board = [fila[:] for fila in self.board]
            new_board[empty_i][empty_j], new_board[empty_i][empty_j-1]
= new_board[empty_i][empty_j-1], new_board[empty_i][empty_j]
            moves.append(new_board)

        # Simular movimiento hacia la derecha
        if self.is_allowed_move_right():
            new_board = [fila[:] for fila in self.board]
            new_board[empty_i][empty_j], new_board[empty_i][empty_j+1]
= new_board[empty_i][empty_j+1], new_board[empty_i][empty_j]
            moves.append(new_board)

        return moves

# Tablero inicial del juego
initial_board = [
    [8, 13, 9, 15],
    [None, 14, 6, 1],
    [11, 2, 5, 4],
    [3, 7, 10, 12]
]

# Crear instancia del juego
game = Game(initial_board)

# Mostrar el tablero inicial
game.show()
```

8	13	9	15
	14	6	1
11	2	5	4
3	7	10	12