Reporte de Divide y Vencerás

TECNICAS ALGORITMICAS

JOSE ALEXANDER HERNANDEZ HERNANDEZ
230300992

11/26/2024

Emmanuel Morales Saavedra

Recorre la fila en la posición row (posición de la fila) del tablero. Tiene como fin comprobar si el número num ya está presente en la fila. Si lo encuentra, retorna False (no es válido colocar el número en esa celda).

Recorre la columna en la posición col (posición de la columna) del tablero. De igual manera comprueba si el número num ya está presente en la columna. Si hay, retorna False.

Calcula la esquina superior izquierda del subcuadro 3x3 que contiene la celda (row, col):

start_row = 3 * (row // 3): Encuentra la fila inicial del subcuadro.

start_col = 3 * (col // 3): Encuentra la columna inicial del subcuadro.

Recorre las 9 celdas del subcuadro 3x3 usando dos bucles anidados (i y j).

Verifica si el número num ya está en alguna de esas celdas.

Propósito: Garantiza que num no se repita dentro del subcuadro 3x3 al que pertenece la celda (row, col). Si lo encuentra, retorna False.

```python
def solve_sudoku(board, row=0, col=0):
    #ROW(POSICION DE LA FILA)
    #COL(COLUMNA DE LA FILA)
    """
    Resolicon del Sudoku usando la tecnica de Divide y Vencerás.
    row, col: Indican la celda actual a resolver.
    """
```

Se define la función para resolver el Sudoku.

board: Es el tablero de Sudoku, una lista de listas de 9x9.

```python
    # ultima celda, el tablero está completo
    if row == 9:    Trailing whitespace
        return True

    # Continuar a la siguiente fila si llega al final de una columna
    if col == 9:    Trailing whitespace
        return solve_sudoku(board, row + 1, 0)

    # Si ya está completa la celda, sigue a la otra celda que continúa
    if board[row][col] != 0:    Trailing whitespace
        return solve_sudoku(board, row, col + 1)
```

1. Verifica si se alcanzó el final del tablero. El índice row indica la fila actual, y el tablero tiene 9 filas (índices de 0 a 8).
2. Sí row == 9, significa que el algoritmo ya revisó todas las celdas del tablero, por lo que devuelve True, indicando que el Sudoku está completo y resuelto. En este caso:

➔ Incrementa row en 1 para pasar a la siguiente fila.
➔ Reinicia col a 0 para comenzar en la primera columna de la nueva fila.
➔ Llama recursivamente a solve_sudoku para continuar resolviendo.
➔ Saltar celdas ya llenas. Verifica si la celda actual (board[row][col]) ya contiene un número (es decir, no es 0).

- Si la celda no está vacía:
- No intenta resolverla.
- Avanza a la siguiente celda de la misma fila incrementando col en 1.
- Llama recursivamente a solve_sudoku para resolver desde la nueva posición.

```python
# Prueba con números del 1 al 9
for num in range(1, 10):
    if is_valid(board, row, col, num):
        board[row][col] = num  # Coloca el número provisionalmente
        print(f"Colocando {num} en la posición ({row}, {col}):")    Trailing whitespace
        print_board(board)

        # Resuelve la celda siguiente
        if solve_sudoku(board, row, col + 1):
            return True

        # Retrocede si no es válido
        board[row][col] = 0
        print(f"Retrocediendo en la posición ({row}, {col}):")    Trailing whitespace
        print_board(board)

return False  # No se pudo resolver (no hay solución)
```

1. El ciclo recorre los números del 1 al 9 para intentar colocar un número válido en la celda actual.
2. Si el número es válido, se coloca en la celda y se intenta resolver la siguiente celda.
3. Si una celda no lleva a una solución, se retrocede eliminando el número y probando con otro.
4. Si todas las celdas se completan correctamente, la función devuelve True y el Sudoku está resuelto.
5. Si no se puede completar el tablero, el algoritmo devuelve False y retrocede hasta encontrar una solución alternativa.

```python
def print_board(board):
    """Imprime el tablero de Sudoku."""
    for i in range(9):
        if i % 3 == 0 and i != 0:
            print("-" * 21)
        for j in range(9):
            if j % 3 == 0 and j != 0:
                print("| ", end="")
            print(board[i][j] if board[i][j] != 0 else ".", end=" ")
        print()
```

1. La función recorre las filas (i) y las columnas (j) del tablero de Sudoku.
2. Cada vez que se llega a una posición de columna que es múltiplo de 3, se imprime una barra (|) para separar los subcuadros de 3x3.
3. Imprime el número de cada celda o un punto (".") si la celda está vacía (es decir, tiene un valor de 0).
4. Después de cada fila, se imprime una línea separadora después de cada 3 filas, para mejorar la visualización del tablero.
5. El resultado final es un tablero de Sudoku bien formateado que se muestra de forma legible.

EJEMPLO DE SALIDA

```python
# Tablero (0 = celdas vacías)
sudoku_board = [
    [5, 3, 0, 0, 7, 0, 0, 0, 0],
    [6, 0, 0, 1, 9, 5, 0, 0, 0],
    [0, 9, 8, 0, 0, 0, 0, 6, 0],
    [8, 0, 0, 0, 6, 0, 0, 0, 3],
    [4, 0, 0, 8, 0, 3, 0, 0, 1],
    [7, 0, 0, 0, 2, 0, 0, 0, 6],
    [0, 6, 0, 0, 0, 0, 2, 8, 0],
    [0, 0, 0, 4, 1, 9, 0, 0, 5],
    [0, 0, 0, 0, 8, 0, 0, 7, 9]
]
```

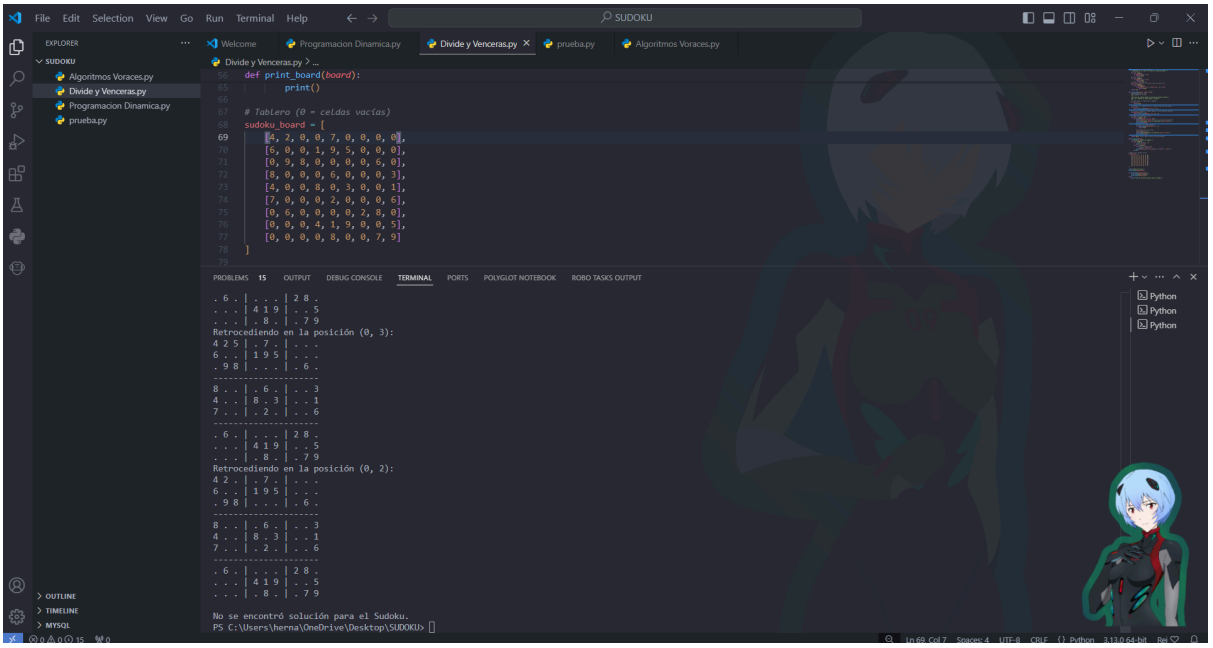
De esta manera quedaría la tabla al ejecutarlo

```
5 3 . | . 7 . | . . .
6 . . | 1 9 5 | . . .
. 9 8 | . . . | . 6 .
---------------------
8 . . | 8 . . | . . .
4 . . | 4 1 9 | . . 5
7 . . | . . . | . 3 .
---------------------
. 6 . | . . . | 2 8 .
. . . | 8 . . | . 7 9
. . . | . 6 . | . . .
```

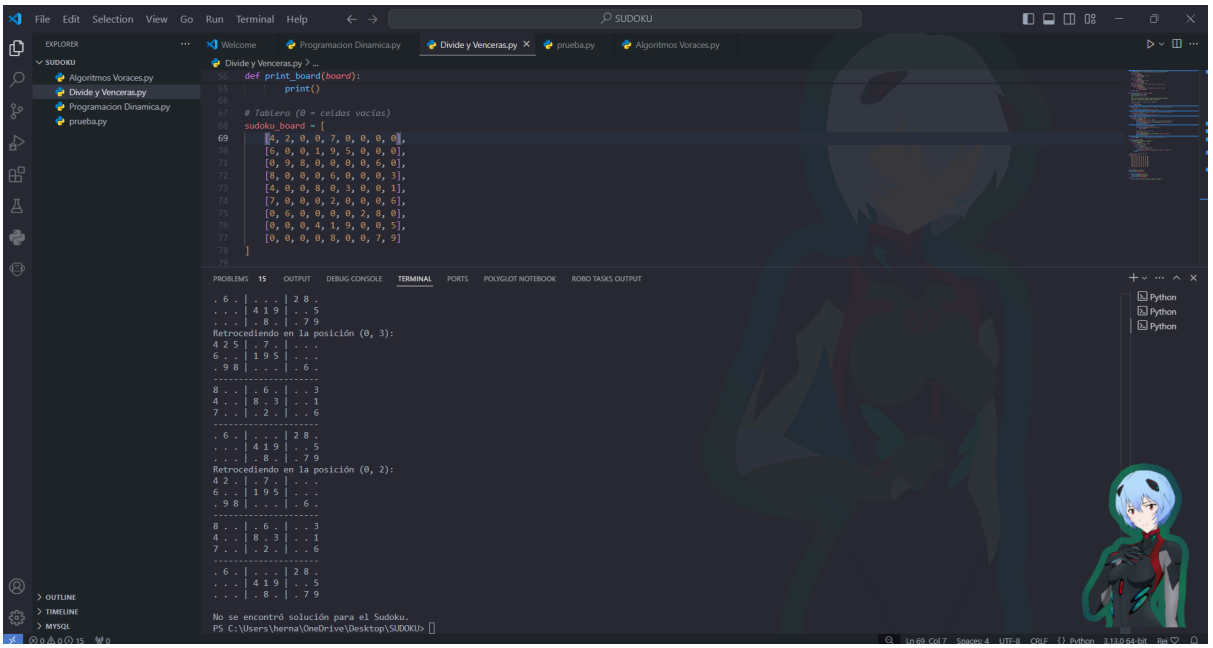
```
print("Tablero inicial:")
print_board(sudoku_board)

if solve_sudoku(sudoku_board):
    print("\n¡Sudoku resuelto!")
    print_board(sudoku_board)
else:
    print("\nNo se encontró solución para el Sudoku.")
```

1. Se imprime el tablero de Sudoku inicial.
2. Se intenta resolver el tablero llamando a solve_sudoku.
    - Si el Sudoku se resuelve correctamente, se muestra el mensaje "¡Sudoku resuelto!" y el tablero resuelto.
    - Si no se puede resolver, se muestra el mensaje "No se encontró solución para el Sudoku.".

EL SUDOKU RESUELTO



SIN SOLUCIÓN



CÓDIGO DE DIVIDE Y VENCERÁS

```
def is_valid(board, row, col, num):
    """Verifica si un número es válido en la posición dada."""
    # revisa fila
    for i in range(9):
        if board[row][i] == num:
            return False
    # revisa columna
    for i in range(9):
        if board[i][col] == num:
            return False
    # revisa el subcuadro 3x3
    start_row, start_col = 3 * (row // 3), 3 * (col // 3)
    for i in range(3):
        for j in range(3):
            if board[start_row + i][start_col + j] == num:
                return False
    return True


def solve_sudoku(board, row=0, col=0):
    #ROW(POSICION DE LA FILA)
    #COL(COLUMNA DE LA FILA)
    """
    Resolicon del Sudoku usando la tecnica de Divide y Vencerás.
    row, col: Indican la celda actual a resolver.
    """
    # ultima celda, el tablero está completo
```

```
    if row == 9:
        return True

    # Continuar a la siguiente fila si llega al final de una columna
    if col == 9:
        return solve_sudoku(board, row + 1, 0)

    # Si ya está completa la celda, sigue a la otra celda que continúa
    if board[row][col] != 0:
        return solve_sudoku(board, row, col + 1)

    # Prueba con números del 1 al 9
    for num in range(1, 10):
        if is_valid(board, row, col, num):
            board[row][col] = num  # Coloca el número provisionalmente
            print(f"Colocando {num} en la posición ({row}, {col}):")
            print_board(board)

            # Resuelve la celda siguiente
            if solve_sudoku(board, row, col + 1):
                return True

            # Retrocede si no es válido
            board[row][col] = 0
            print(f"Retrocediendo en la posición ({row}, {col}):")
            print_board(board)
```

```
    return False  # No se pudo resolver (no hay solución)


def print_board(board):
    """Imprime el tablero de Sudoku."""
    for i in range(9):
        if i % 3 == 0 and i != 0:
            print("-" * 21)
        for j in range(9):
            if j % 3 == 0 and j != 0:
                print("| ", end="")
            print(board[i][j] if board[i][j] != 0 else ".", end=" ")
        print()


# Tablero (0 = celdas vacías)
sudoku_board = [
    [4, 2, 0, 0, 7, 0, 0, 0, 0],
    [6, 0, 0, 1, 9, 5, 0, 0, 0],
    [0, 9, 8, 0, 0, 0, 0, 6, 0],
    [8, 0, 0, 0, 6, 0, 0, 0, 3],
    [4, 0, 0, 8, 0, 3, 0, 0, 1],
    [7, 0, 0, 0, 2, 0, 0, 0, 6],
    [0, 6, 0, 0, 0, 0, 2, 8, 0],
    [0, 0, 0, 4, 1, 9, 0, 0, 5],
    [0, 0, 0, 0, 8, 0, 0, 7, 9]
]


print("Tablero inicial:")
```

```
print_board(sudoku_board)



if solve_sudoku(sudoku_board):

    print("\n¡Sudoku resuelto!")

    print_board(sudoku_board)

else:

    print("\nNo se encontró solución para el Sudoku.")
```