Resolución de Sudoku Usando Búsqueda en Anchura y Profundidad

César Omar Martín Rodríguez
Universidad Panamericana
Aguascalientes, México
0244161@up.edu.mx

***Abstract*—Este artículo presenta la implementación y comparación de dos algoritmos clásicos de búsqueda —Búsqueda en Anchura (BFS) y Búsqueda en Profundidad (DFS)— aplicados a la resolución automática del juego de Sudoku. La solución incluye una interfaz gráfica con `Tkinter`, lectura de datos desde archivos CSV y visualización de resultados en tiempo real. Se discuten los resultados obtenidos, diferencias de rendimiento y posibles extensiones del trabajo.**

***Index Terms*—Sudoku, BFS, DFS, Python, Tkinter, Visualización, Grafos, Búsqueda**

I. Introducción

El Sudoku es un juego lógico que requiere rellenar una cuadrícula 9x9 con dígitos del 1 al 9, sin repetir valores en filas, columnas ni subcuadrantes de 3x3. Resolver un Sudoku puede modelarse como un problema de búsqueda de soluciones en un grafo implícito, donde cada estado representa un tablero parcialmente lleno.

II. Objetivo

Implementar algoritmos de búsqueda BFS y DFS para resolver automáticamente un Sudoku, comparando su rendimiento y efectividad, e integrando una interfaz visual interactiva.

III. Metodología

A. Entrada y Representación

El tablero de Sudoku se representa como una matriz de 9x9, donde el valor 0 indica una celda vacía. Los datos se leen desde un archivo CSV llamado `sudoku_input.csv`.

B. Algoritmos Implementados

DFS (Depth-First Search) explora una rama del árbol de búsqueda hasta encontrar una solución o un callejón sin salida. Se implementó de forma recursiva con backtracking.

BFS (Breadth-First Search) utiliza una cola FIFO para explorar todas las posibilidades por nivel. Aunque menos eficiente en este caso, sirve como contraste interesante.

C. Interfaz Gráfica

Ambas versiones del programa muestran el Sudoku en una ventana de `Tkinter`, permitiendo al usuario visualizar el tablero y presionar un botón para iniciar la resolución.

IV. Resultados

Se evaluó el desempeño de ambos algoritmos resolviendo el mismo tablero.

A. Tablero de Entrada

```
[7,0,0,8,0,3,0,0,5]
[0,0,5,0,7,0,3,0,0]
[0,4,0,0,6,0,0,2,0]
...
```

B. Tiempos de Ejecución

Algoritmo	Tiempo (s)
DFS	0.06
BFS	0.09

TABLE I
Comparación de tiempos de ejecución

C. Observaciones

- DFS fue significativamente más rápido y consume menos memoria.
- BFS exploró muchos más nodos intermedios, lo que afecta el rendimiento.
- Ambos algoritmos lograron encontrar una solución válida.

V. Conclusiones

DFS es más eficiente para resolver Sudokus debido a su menor consumo de recursos y ejecución más directa. BFS, aunque funcional, no es ideal en este tipo de problemas por su naturaleza exhaustiva. Se cumplieron todos los objetivos: lectura desde archivo, interfaz gráfica, implementación de algoritmos y análisis comparativo.

VI. Trabajo Futuro

Como extensiones futuras se propone:

- Agregar exportación a archivo de salida.
- Incluir otras técnicas como A* o heurísticas.
- Resolver Sudokus de distintas dificultades.
- Medición del número de nodos generados.

Agradecimientos

Gracias al profesor por su guía y paciencia, y al café por su incansable apoyo moral.

REPOSITORIO

El proyecto completo está organizado con la siguiente estructura:

```
/ProyectoSudoku/
  |- sudoku_input.csv
  |- SudokuBFS.py
  |- SudokuDFS.py
  |- README.pdf
  |- informe.pdf
```