Sudoku Project: Validation, Resolution, and Generation Algorithms

Hever Andre Alfonso Jimenez
Juan Andrés Young Hoyos
Mateo Villada Higuita
Juan Camilo Ríos Rodríguez
Universidad EAFIT

September 30, 2024

1 Algoritmo para Validar un n-Sudoku

1.1 Descripción del Algoritmo

Este algoritmo valida un n-Sudoku verificando que las filas, columnas y subcuadrículas no tengan duplicados y que todos los valores estén dentro del rango permitido.

1.2 Complejidad del Algoritmo

El algoritmo de validación de Sudoku tiene una complejidad temporal de $O(n^2)$, ya que cada una de las verificaciones de filas, columnas y cajas se realiza en un tiempo lineal respecto al tamaño de la cuadrícula n^2.

1.3 Heurística Utilizada

Para simplificar la verificación de la validez del Sudoku, se utiliza una división modular de las subcuadrículas y verificaciones lineales para evitar duplicados. Esta heurística se basa en el aprovechamiento de la estructura jerárquica de las subcuadrículas del Sudoku.

Algorithm 1 Validador de Sudoku

Inicializar la cuadrícula del Sudoku *grid*
Calcular n como el tamaño de la cuadrícula ($n \times n$)
Calcular *box_size* como la raíz cuadrada de n
if $box\_size^2 \neq n$ **then**
    Lanzar un error: "El tamaño de la cuadrícula no es válido. n debe ser un cuadrado perfecto."
end if
Definir *numbers* como el conjunto de números del 1 al n
Verificar si el Sudoku es válido comprobando filas, columnas y subcuadrículas
Función *check_rows*():
for cada fila i en *grid* **do**
    Obtener todos los números no vacíos de la fila
    if hay números duplicados **then**
        Imprimir "Error en la fila $i + 1$"
        Retornar FALSO
    end if
end for
Retornar VERDADERO
Función *check_columns*():
for cada columna j en *grid* **do**
    Obtener todos los números no vacíos de la columna
    if hay números duplicados **then**
        Imprimir "Error en la columna $j + 1$"
        Retornar FALSO
    end if
end for
Retornar VERDADERO
Función *check_boxes*():
for cada subcuadrícula comenzando en (box_row, box_col) **do**
    Obtener todos los números no vacíos de la subcuadrícula
    if hay números duplicados **then**
        Imprimir "Error en la caja que comienza en (box_row+1, box_col+1)"
        Retornar FALSO
    end if
end for
Retornar VERDADERO

2 Algoritmo para Resolver un n-Sudoku

2.1 Descripción del Algoritmo

Este algoritmo utiliza una combinación de heurísticas y backtracking para resolver un n-Sudoku. Primero llena celdas con candidatos únicos y utiliza backtracking para explorar posibles soluciones.

2.2 Pasos del Algoritmo

- **Leer Entrada:** Procesa un n-Sudoku desde la entrada estándar y lo convierte en una estructura con valores y notas.
- **Actualizar Notas:** Para cada celda vacía, actualiza los candidatos posibles considerando filas, columnas y subcuadrículas.
- **Aplicar Heurísticas:** Identifica y asigna valores únicos usando solitarios ocultos y candidatos únicos.
- **Backtracking:** Explora configuraciones posibles si las heurísticas no llenan todas las celdas.

2.3 Complejidad del Algoritmo

El algoritmo tiene una complejidad de $O((n^2)!)$ en el peor caso debido al uso de backtracking. Sin embargo, las heurísticas reducen el tiempo de ejecución en la mayoría de los casos prácticos.

2.4 Heurística Utilizada

- **Candidatos Únicos:** Si una celda tiene un único candidato posible, se asigna automáticamente.
- **Solitarios Ocultos:** Se identifica el único lugar en una fila, columna o subcuadrícula donde un número puede colocarse.

2.5 Implementación del Algoritmo

Algorithm 2 Resolución de un n-Sudoku con Backtracking

```
Leer el tablero de Sudoku desde la entrada
Inicializar candidatos para cada celda vacía
while existan celdas vacías do
    Actualizar notas para todas las celdas
    Aplicar heurísticas para rellenar candidatos únicos y solitarios ocultos
    if no se puedan rellenar más celdas then
        Usar backtracking para explorar posibles soluciones
    end if
end while
Retornar el tablero resuelto si tiene solución
```

3 Algoritmo para Crear un n-Sudoku

3.1 Descripción del Algoritmo

Este algoritmo genera un tablero completo de Sudoku y elimina números aleatoriamente para dejar un porcentaje deseado de pistas, creando un rompecabezas válido.

3.2 Pasos del Algoritmo

- **Generar Tablero Completo:** Se construye un Sudoku completo utilizando un patrón predefinido y se mezclan filas, columnas y números aleatoriamente.
- **Eliminar Números:** Se eliminan números de las celdas de forma aleatoria hasta alcanzar la cantidad deseada de pistas.
- **Mantener Simetría:** Los números se eliminan en pares simétricos para mantener la estética del Sudoku.

3.3 Complejidad del Algoritmo

El algoritmo tiene una complejidad de $O(n^2)$, ya que los pasos principales (generación, mezcla y eliminación) operan sobre cada celda una sola vez.

3.4 Heurística Utilizada

- **Simetría:** Los números se eliminan de forma simétrica para mantener la apariencia clásica del Sudoku.
- **Aleatoriedad Controlada:** Las celdas se seleccionan de manera aleatoria para garantizar la variabilidad en los rompecabezas generados.

3.5 Implementación del Algoritmo

Algorithm 3 Generación de un n-Sudoku

Generar una cuadrícula base para el Sudoku
Mezclar filas, columnas y bandas de manera aleatoria
Definir el número de pistas deseadas
while queden celdas por eliminar **do**
    Seleccionar una celda (i, j) aleatoriamente
    if la celda no está vacía **then**
        Eliminar el número de la celda y su par simétrico
    end if
end while
Retornar la cuadrícula generada