**Trabajo Práctico: Resolución de un Sudoku**

**Objetivo:**

El objetivo es implementar y comparar diferentes enfoques algorítmicos para resolver el problema de la **resolución de un Sudoku**, utilizando las técnicas de **Backtracking** y **Branch & Bound (B&B)**. El Sudoku debe ser resuelto completando las celdas vacías con números del 1 al 9, respetando las restricciones del juego.

Como complemento a este problema, debe resolverse el método para crear un tablero. Éste método es parte del entregable.

---

**Descripción del Problema:**

Dado un tablero de Sudoku parcialmente lleno de tamaño 9×9, el objetivo es completar las celdas vacías de manera que cada fila, columna y subcuadro 3×3 contenga los números del 1 al 9 sin repeticiones.

---

**Instrucciones Generales:**

- El trabajo deberá ser realizado en el lenguaje de programación **Python** o **Java**.
- Se debe implementar la solución del Sudoku utilizando **Backtracking** y **Branch & Bound (B&B)**.
- Cada solución deberá ser analizada en cuanto a su **eficiencia** y el **número de nodos explorados** en cada técnica.
- Probar las implementaciones con tableros de Sudoku de diferentes niveles de dificultad (fácil, medio, difícil).

---

**Parte 1: Backtracking para la Resolución del Sudoku**

**Descripción:**

Implementa el algoritmo de **Backtracking** para resolver el Sudoku. Este algoritmo debe explorar todas las posibilidades de colocar números en las celdas vacías hasta encontrar una solución válida.

**Tareas:**

1. **Backtracking básico**:

- Implementa la solución del Sudoku utilizando backtracking. Para cada celda vacía, intenta colocar un número del 1 al 9.
- Si colocar un número no es válido (viola las reglas del Sudoku), retrocede y prueba con otro número.
- La solución debe funcionar para tableros de Sudoku de diferentes niveles de dificultad.

2. **Análisis**:
   - Mide el **número de nodos explorados** (intentos de colocar un número) y el **tiempo de ejecución** para tableros de distintos niveles de dificultad.
   - Reflexiona sobre la complejidad temporal de este algoritmo.

---

**Parte 2: Optimización con Branch & Bound (B&B)**

**Descripción:**

Optimiza la búsqueda utilizando **Branch & Bound (B&B)** para reducir el número de posibilidades que exploras. Utiliza una heurística que priorice colocar números en **las celdas más restringidas** (las que tienen menos opciones disponibles).

**Tareas:**

1. **Branch & Bound**:
   - Modifica la solución de backtracking para incorporar B&B. Utiliza una heurística que priorice colocar números en celdas donde haya menos opciones (variables más restringidas).
   - Implementa el cálculo de **cotas** (bounds) que permitan podar caminos que no conduzcan a una solución válida.
2. **Análisis**:
   - Mide el número de nodos explorados y el tiempo de ejecución, comparándolo con el algoritmo de **backtracking puro**.
   - Reflexiona sobre cómo B&B afecta el rendimiento del algoritmo en tableros de dificultad creciente.

---

**Parte 3: Comparación Experimental**

**Tareas:**

- Realiza pruebas con tableros de Sudoku de diferentes niveles de dificultad
    - Fácil entre 35 y 50 números en el tablero inicial
    - Medio entre 22 y 34 números en el tablero inicial
    - Difícil entre 10 y 21 números en el tablero inicial
- Compara los dos enfoques (Backtracking puro y Branch & Bound) en cuanto a:
    - **Número de nodos explorados**: ¿Cuántos intentos se realizaron antes de llegar a la solución?
    - **Tiempo de ejecución**: ¿Qué tan rápido es cada enfoque?
    - **Facilidad de implementación**: Reflexiona sobre cuál de las técnicas fue más fácil de implementar y ajustar para este problema.

---

**Parte 4: Informe Final**

**Incluir:**

1. **Explicación detallada de cada implementación**:
    - Explica cómo implementaste cada técnica (**Backtracking**, **B&B**) para resolver el Sudoku.
2. **Comparación de resultados**:
    - Incluye gráficos y tablas que muestren el número de nodos explorados y el tiempo de ejecución de cada técnica para tableros de diferentes niveles de dificultad.
3. **Reflexiones**:
    - Explica por qué ciertas técnicas funcionaron mejor que otras.
    - Reflexiona sobre la aplicabilidad de cada una en problemas de optimización similares.

---

**Entregables:**

- **Código fuente** de las implementaciones.

- **Informe en formato PDF** que incluya el análisis y las comparaciones mencionadas anteriormente.
- **Pruebas realizadas** en instancias de tableros de Sudoku de diferentes niveles de dificultad.

---

**Criterios a evaluar:**

- **Correctitud del algoritmo**: El algoritmo debe resolver correctamente cualquier instancia válida de Sudoku.
- **Optimización**: Minimización de nodos explorados y tiempo de ejecución gracias a la poda mediante **B&B**.
- **Heurística empleada**: Evaluar cómo se guía la búsqueda y la poda de soluciones no prometedoras.
- **Comparación de resultados**: Se valorará la presentación de los resultados obtenidos y la comparación entre las diferentes técnicas utilizadas.

---

**Presentación:**

- La presentación se hará sobre una PC que muestre los resultados obtenidos. Estos deben incluir una representación visual del Sudoku resuelto.
- Cada uno de los participantes tendrá un tiempo igual para dar su parte de la presentación.
- Se realizarán preguntas generales sobre el método de selección de nodos, backtracking y cualquier componente del trabajo, tanto de código como de la estrategia de resolución.
- La **aprobación será individual**.