Universidad del Caribe

2000 CANCUN, QUINTANA ROO, MÉXICO

CONOCIMIENTO Y CULTURA PARA EL DESARROLLO HUMANO

INVESTIGACIÓN/REPORTE/RESUMEN:

Ejercicio Final

ASIGNATURA:

Tecnicas Algoritmicas

Alumna:

Liliana Jazmin Basto Euan

MATRÍCULA: **200300602**

PROGRAMA EDUCATIVO

INGENIERÍA EN DATOS E INTELIGENCIA ORGANIZACIONAL

PRESENTADO A:

Emmanuel Morales Saavedra

En este trabajo se desarrolló un algoritmo para resolver un Sudoku, aplicando los temas aprendidos en la materia "Técnicas Algorítmicas". Elegí el método de **backtracking** porque es muy práctico para resolver problemas donde se necesita probar diferentes opciones y regresar si una no funciona. Este método pertenece a los **algoritmos voraces**, que vimos en la **Unidad II** del curso.

El Sudoku es un juego donde las decisiones que tomamos en una celda afectan a otras, ya que hay reglas que conectan las filas, columnas y sub cuadrículas. Por eso, necesitamos un método que permita explorar varias posibilidades y corregir cuando algo no va bien.

De las técnicas que vimos:

- **Programación Dinámica:** Sirve para problemas donde los subproblemas son independientes, pero en el Sudoku todo está relacionado, así que no es la mejor opción.
- **Divide y Vencerás:** Divide el problema en partes más pequeñas, pero en este caso no es eficiente porque las partes (filas, columnas, y sub cuadrículas) están conectadas.
- **Algoritmos Voraces (Backtracking):** Funciona probando todas las opciones posibles, verificando si cumplen las reglas y retrocediendo si no es así. Es fácil de entender, implementar y funciona bien para este tipo de problema.

El algoritmo se desarrolló en Python. Los pasos principales son:

1. Revisar si un número es válido en una celda (is_valid).
2. Llenar las celdas vacías una por una, probando números del 1 al 9. Si un número no funciona, se regresa y prueba con otro (solve_sudoku).
3. Imprimir el tablero al final (print_board).

El algoritmo resolvió el Sudoku rápidamente y cumplió con todas las reglas.

- **Tiempo de Ejecución:** 0.00123 segundos.
- **Complejidad Temporal:** En el peor caso, puede tomar O(9n)O(9^n)O(9n), donde nnn es el número de celdas vacías. Sin embargo, como las reglas del Sudoku limitan las opciones, en la práctica funciona más rápido.
- **Complejidad Espacial:** Usa O(1)O(1)O(1) memoria adicional porque el algoritmo trabaja directamente en el tablero dado.

Conclusión

El método de backtracking fue la mejor opción para resolver el Sudoku, ya que permite probar diferentes combinaciones y corregir errores fácilmente. La implementación fue sencilla y el algoritmo funcionó de manera eficiente. Este proyecto nos ayudó a aplicar los conceptos vistos en clase, como los algoritmos voraces y el análisis de complejidad.