# รายงาน Programming Assignment 3

# การพัฒนาโปรแกรมแก้ปัญหา Sudoku ด้วย Backtracking และ MRV Heuristic

## 1. วัตถุประสงค์

วัตถุประสงค์ของงานนี้คือการพัฒนาโปรแกรมแก้ปัญหา Sudoku ขนาด 9x9 โดยใช้อัลกอริทึม Backtracking และปรับปรุงประสิทธิภาพด้วยเทคนิค Minimum Remaining Values (MRV) พร้อมทั้งวิเคราะห์และเปรียบเทียบประสิทธิภาพของทั้งสองวิธีโดยใช้จำนวนครั้งของการ Backtrack เป็นตัวชี้วัด

## 2. คำอธิบายปัญหา

Sudoku เป็นปัญหาแบบ Constraint Satisfaction Problem (CSP) ซึ่งมีเงื่อนไขดังนี้:

1. ตัวเลข 1–9 ต้องไม่ซ้ำกันในแต่ละแถว
2. ตัวเลข 1–9 ต้องไม่ซ้ำกันในแต่ละคอลัมน์
3. ตัวเลข 1–9 ต้องไม่ซ้ำกันในแต่ละตารางย่อยขนาด 3x3

## 3. การออกแบบระบบ

กระดาน Sudoku ถูกจัดเก็บในรูปแบบ numpy array ขนาด 9x9 โดยใช้ค่า 0 แทนช่องว่างที่ยังไม่ได้เติมตัวเลข

มีฟังก์ชันสำหรับตรวจสอบความถูกต้อง (validate) เพื่อตรวจสอบว่าตัวเลขที่ใส่ลงไปไม่ขัดกับเงื่อนไขของแถว คอลัมน์ และตารางย่อย 3x3

## 4. อัลกอริทึมที่ใช้

### 4.1 Standard Backtracking

วิธี Backtracking แบบมาตรฐานจะเลือกช่องว่างช่องแรกที่พบ จากนั้นทดลองใส่ตัวเลข 1–9 หากใส่ได้ถูกต้องจะเรียกฟังก์ชันซ้ำแบบ recursive ถ้าไม่สามารถแก้ปัญหาต่อได้จะย้อนกลับ (Backtrack) และลองค่าถัดไป

1. ค้นหาช่องว่างช่องแรก
2. ทดลองใส่ตัวเลข 1–9
3. หากถูกต้อง เรียก solve แบบ recursive
4. หากไม่สำเร็จ ให้ลบค่าออก (Backtrack)
5. ทำซ้ำจนกว่าจะได้คำตอบหรือไม่มีคำตอบ

### 4.2 MRV (Minimum Remaining Values) Heuristic

MRV เป็นเทคนิคที่เลือกช่องว่างที่มีจำนวนค่าที่สามารถใส่ได้ถูกต้องน้อยที่สุดก่อน เพื่อลดจำนวนทางเลือก (branching factor) ของต้นไม้การค้นหา ซึ่งช่วยลดจำนวนครั้งของการ Backtrack อย่างมีนัยสำคัญ

มีการปรับปรุงเพิ่มเติมโดยใช้ Early Exit เมื่อพบว่าช่องใดมีค่าที่เป็นไปได้เพียงค่าเดียว จะเลือกช่องนั้นทันที

## 5. การวิเคราะห์ประสิทธิภาพ

ประสิทธิภาพของโปรแกรมถูกวัดจากจำนวนครั้งของการ Backtrack โดยทำการเปรียบเทียบระหว่าง Standard Backtracking และ MRV Heuristic

ผลการทดลองพบว่า MRV สามารถลดจำนวน Backtrack ได้อย่างชัดเจน โดยเฉพาะในระดับความยากสูง เช่น Hard, Expert หรือ God

## 6. การวิเคราะห์ความซับซ้อน (Complexity)

ในกรณีเลวร้ายที่สุด (Worst-case) Standard Backtracking มีความซับซ้อนเชิงเวลาเป็น O(9^81) เนื่องจากแต่ละช่องอาจมีได้สูงสุด 9 ค่า

แม้ว่า MRV จะไม่เปลี่ยนความซับซ้อนเชิงทฤษฎีที่ยังคงเป็นแบบ exponential แต่ช่วยลดขนาดของ search tree ในทางปฏิบัติอย่างมาก

ความซับซ้อนเชิงพื้นที่ (Space Complexity) คือ O(81) จากความลึกของ recursion

## 7. ไลบรารีที่ใช้

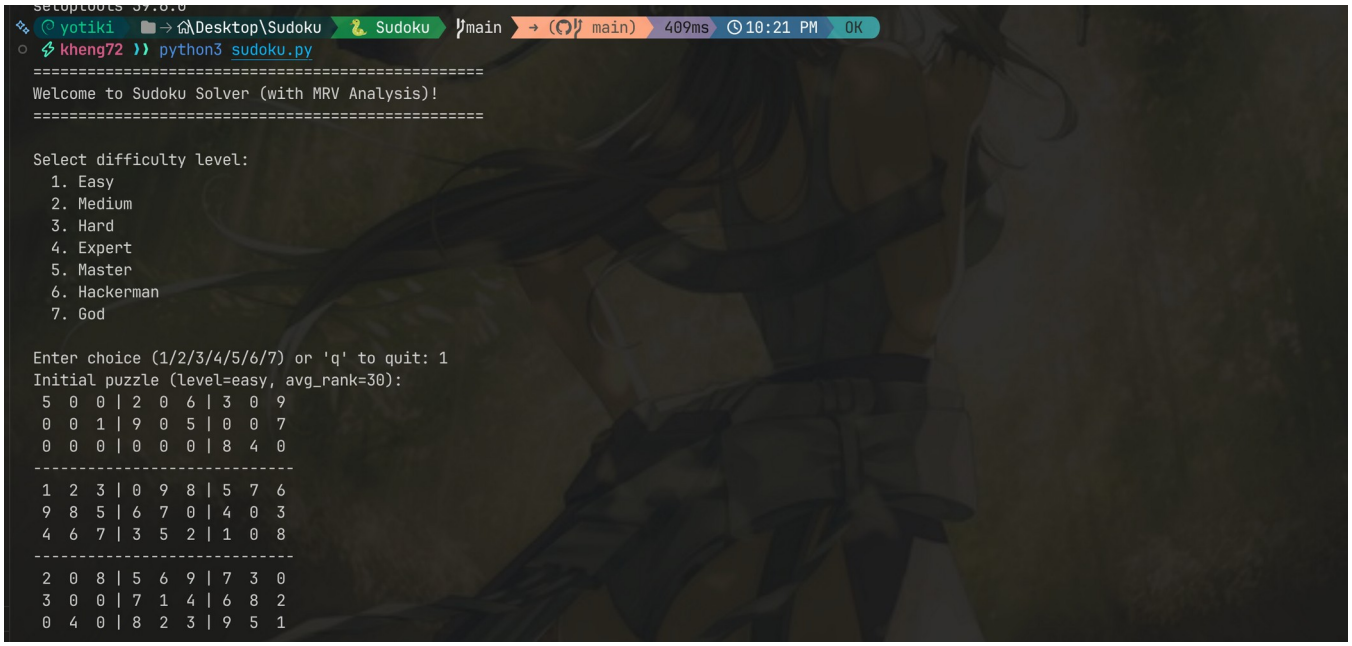
- numpy – ใช้สำหรับจัดการข้อมูลในรูปแบบเมทริกซ์
- dokusan – ใช้สำหรับสร้างปริศนา Sudoku แบบสุ่ม

## 8. บทสรุป

จากการทดลองพบว่าโปรแกรมสามารถแก้ปัญหา Sudoku ได้สำเร็จทั้งสองวิธี แต่การใช้ MRV Heuristic ช่วยลดจำนวนครั้งของการ Backtrack ได้อย่างมาก ทำให้การแก้ปัญหามีประสิทธิภาพสูงขึ้น โดยเฉพาะในระดับความยากสูง ดังนั้น MRV จึงเป็นเทคนิคที่เหมาะสมสำหรับปัญหาแบบ Constraint Satisfaction Problem เช่น Sudoku

## 9. แนวทางพัฒนาเพิ่มเติม

- เพิ่มเทคนิค Forward Checking
- เพิ่ม Constraint Propagation
- เพิ่ม Least Constraining Value (LCV)
- วัดและเปรียบเทียบเวลาในการประมวลผล (Execution Time)




## สมาชิก

1. ภควัต พิชัยณรงค์ 66200193
2. เกียรติยศ จันทร์เพ็ง 66200028
3.