BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ THÀNH PHỐ HỒ CHÍ MINH**

**KHOA KINH DOANH**

**\*\*\***

**A blue and black logo

Description automatically generated**

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**GIẢI BÀI TOÁN SUDOKU**

**Môn học:** [Introduction to Artificial Intelligence](https://sis.umt.edu.vn/attendance/228da4d3-8a4f-4608-a86c-62210488434a)

**Giảng viên hướng dẫn:** Nguyễn Quản Bá Hồng

**Mã lớp: 1488**

**Tên:** Nguyễn Nhật Khôi

**MSSV: 2302700156**

**Thành phố Hồ Chí Minh, tháng 7 năm 2025**

**1. Giới Thiệu**

Bài toán Sudoku là một bài toán cổ điển trong lĩnh vực trí tuệ nhân tạo và giải thuật. Nó yêu cầu điền các số từ 1 đến 9 vào một bảng 9x9 sao cho mỗi hàng, mỗi cột và mỗi vùng 3x3 không trùng nhau. Để giải bài toán này, chúng ta sử dụng thuật toán backtracking, một kỹ thuật mạnh mẽ trong việc tìm kiếm lời giải cho bài toán có không gian tìm kiếm lớn.

Trong đồ án này, em không chỉ giải quyết bài toán Sudoku với một bảng 9x9, mà còn mở rộng bài toán này để giải các biến thể Sudoku 16x16, hỗ trợ Sudoku X (có ràng buộc đường chéo), và hiển thị thông tin về thời gian giải và số lần thử.

* 1. **Mục Tiêu Đồ Án**
* **Giải bài toán Sudoku cơ bản (9x9)**.
* **Áp dụng thuật toán backtracking** kết hợp với **heuristic MRV** để tối ưu quá trình giải.
* **Xây dựng giao diện người dùng** với Tkinter để dễ dàng tương tác và hiển thị kết quả.
* **Hiển thị thông tin về số lần thử và thời gian giải** cho người dùng.
* **Tạo đề Sudoku tự động và đảm bảo có lời giải duy nhất**.

## 2. Các Yêu Cầu Đồ Án

## 2.1. Yêu cầu cơ bản

- **Nhập bảng Sudoku chưa hoàn chỉnh (9x9)**.

-**Giải Sudoku bằng thuật toán backtracking kết hợp heuristic**: Chọn ô trống có ít khả năng điền nhất (MRV).

- **Xuất lời giải hoặc báo không có lời giải nếu bài Sudoku không hợp lệ**.

**2.2.Yêu Cầu Nâng Cao:**

- Hiển thị số lần thử và thời gian giải.

- Tạo giao diện nhập liệu và hiển thị lời giải.

- Tự động phát hiện Sudoku không có lời giải.

- Tạo đề tự động và đảm bảo có lời giải duy nhất.

**3. Mô Tả Các File**

**3.1 File sudoku\_solver.py**

Đây là file chính chứa các thuật toán giải quyết bài toán Sudoku. Các hàm chính trong file này là:

-is\_valid(board, row, col, num): Kiểm tra xem việc điền số num vào ô (row, col) có hợp lệ không. Kiểm tra này bao gồm việc kiểm tra:

+ Hàng: Kiểm tra số num đã xuất hiện trong hàng chưa.

+ Cột: Kiểm tra số num đã xuất hiện trong cột chưa.

+ Vùng 3x3: Kiểm tra số num đã xuất hiện trong vùng 3x3 chưa.

-find\_mrv(board): Tìm ô trống có ít khả năng điền số nhất (Minimum Remaining Values - MRV). Đây là một kỹ thuật heuristic nhằm tối ưu hóa thuật toán giải Sudoku bằng cách ưu tiên điền vào những ô có ít số khả thi nhất.

-solve\_sudoku(board): Thuật toán giải Sudoku bằng cách sử dụng phương pháp backtracking kết hợp với MRV. Thuật toán này sẽ thử tất cả các khả năng hợp lệ để điền vào các ô trống cho đến khi tìm được lời giải.

-generate\_full\_board(): Hàm này tạo ra một bảng Sudoku đầy đủ, tức là một bảng đã được giải. Đây là bảng Sudoku hoàn chỉnh mà không có ô trống.

-remove\_cells(board, num\_remove=40): Hàm này dùng để xóa đi một số ô trong bảng Sudoku đã giải để tạo ra một bài toán Sudoku chưa hoàn chỉnh, có thể dùng để thử nghiệm. Số lượng ô bị xóa được điều chỉnh qua tham số num\_remove.

**3.2 File sudoku\_gui.py**

File này tạo giao diện người dùng sử dụng Tkinter, cho phép người dùng nhập vào bảng Sudoku, giải bài toán, và hiển thị kết quả. Các chức năng trong file này bao gồm:

-Nhập bảng Sudoku: Người dùng có thể nhập vào các ô của bảng Sudoku.

-Giải bài toán: Khi người dùng nhấn nút Solve, chương trình sẽ giải bài toán và hiển thị kết quả.

-Tạo đề Sudoku mới: Khi người dùng nhấn nút Generate New Sudoku, chương trình sẽ tạo một bài Sudoku mới.

**3.3 File sudoku\_gui\_advanced.py**

Đây là file mở rộng cho yêu cầu nâng cao. Nó cung cấp các tính năng bổ sung như:

- Hiển thị số lần thử và thời gian giải.

- Tự động phát hiện khi không có lời giải.

**4.Khó khăn trong việc thực hiện yêu cầu nâng cao**

Trong quá trình làm đồ án, em đã gặp phải một số khó khăn khi thực hiện các yêu cầu nâng cao, đặc biệt là trong việc mở rộng bài toán Sudoku từ kích thước 9x9 sang 16x16 và thêm các ràng buộc mới như Sudoku X. Dưới đây là những khó khăn cụ thể em đã gặp phải và cách giải quyết:

**4.1. Mở rộng kích thước bảng Sudoku (16x16)**

Khi chuyển từ bài toán Sudoku 9x9 sang 16x16, em đã phải điều chỉnh rất nhiều phần trong mã nguồn, đặc biệt là các hàm kiểm tra tính hợp lệ và giải thuật giải Sudoku. Việc chuyển từ 9x9 sang 16x16 không chỉ là thay đổi kích thước mảng mà còn cần phải xử lý việc kiểm tra các vùng con 4x4 (thay vì 3x3 như trong bảng 9x9). Điều này yêu cầu em phải cập nhật lại các hàm kiểm tra tính hợp lệ của một số được điền vào bảng (hàm is\_valid) để làm việc với vùng 4x4 thay vì 3x3.

Em đã sửa lại hàm is\_valid để kiểm tra các vùng con 4x4 và mở rộng khả năng kiểm tra số vào hàng, cột và vùng con này. Mặt khác, việc sử dụng số từ 1 đến 16 thay vì 1 đến 9 cũng đòi hỏi em phải thay đổi phạm vi kiểm tra trong thuật toán.

**4.2. Thêm ràng buộc đường chéo (Sudoku X)**

Một trong những yêu cầu khó khăn là thêm ràng buộc đường chéo trong Sudoku X. Để thực hiện yêu cầu này, em phải đảm bảo rằng các số trên các đường chéo chính (từ góc trên trái đến góc dưới phải) và đường chéo phụ (từ góc trên phải đến góc dưới trái) không bị trùng lặp. Điều này em phải bổ sung thêm các điều kiện kiểm tra trong hàm is\_valid, điều này làm tăng độ phức tạp của thuật toán.

Em thêm các điều kiện kiểm tra đường chéo vào trong hàm is\_valid, bao gồm kiểm tra các đường chéo chính và phụ. Đây là một thách thức lớn vì yêu cầu phải kiểm tra nhiều ràng buộc cùng một lúc.

**4.3. Hiển thị số lần thử và thời gian giải**

Một khó khăn nữa là việc tính toán và hiển thị số lần thử cũng như thời gian giải cho người dùng. Ban đầu, em không rõ cách thức nào có thể hiệu quả trong việc tính toán số lần thử và đồng thời đo thời gian giải thuật.

Em sử dụng thư viện time của Python để đo thời gian giải và biến attempt\_counter để đếm số lần thử. Mỗi lần thuật toán thử một số khác nhau vào ô trống, em tăng biến đếm và sau khi giải xong, em hiển thị kết quả này trong giao diện người dùng.

**4.4. Giải quyết bài toán Sudoku không có lời giải**

Một trong những yêu cầu quan trọng là phải tự động phát hiện khi bài toán Sudoku không có lời giải. Ban đầu, em gặp khó khăn trong việc xác định cách thức kiểm tra và thông báo lỗi khi không tìm thấy lời giải.

Em sài hàm solve\_sudoku để kiểm tra xem bảng Sudoku có thể giải được hay không. Nếu không giải được, em sử dụng hộp thoại thông báo lỗi (messagebox) để thông báo cho người dùng biết rằng bài toán không có lời giải.

**4.5. Tạo đề tự động và đảm bảo có lời giải duy nhất**

Một yêu cầu rất khó khăn là phải tạo đề Sudoku một cách tự động và đảm bảo rằng có đúng một lời giải duy nhất. Để làm được điều này, em đã sử dụng thuật toán backtracking để sinh bảng Sudoku đầy đủ, sau đó xóa các ô một cách ngẫu nhiên cho đến khi tạo ra một bài toán có lời giải duy nhất. Tuy nhiên, việc đảm bảo chỉ có một lời giải duy nhất đòi hỏi phải kiểm tra và thử nghiệm nhiều lần.

Em lấy thuật toán giải Sudoku kết hợp với việc sinh và xóa các ô để tạo ra bài toán. Sau đó, em kiểm tra tính duy nhất của lời giải bằng cách sử dụng giải thuật backtracking hai lần. Nếu có nhiều lời giải, em sẽ tiếp tục thay đổi số lượng ô bị xóa cho đến khi đảm bảo bài toán có đúng một lời giải duy nhất.