BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM

BÀI TẬP LỚN

TÊN HỌC PHẦN: TRÍ TUỆ NHÂN TẠO

Giáo viên hướng dẫn: ThS. Lê Trung Hiếu

Hà Nội, năm 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM

BÀI TẬP LỚN
TÊN HỌC PHẦN: TRÍ TUỆ NHÂN TẠO
ĐỀ TÀI: GAME SUDOKU

STT	Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
				Bằng Số	Bằng Chữ
1	1771020565	Trịnh Minh Quân	14/09/2005		
2	1771020267	Nguyễn Minh Hiếu	14/09/2005		
3	1771020518	Đỗ Trọng Nguyên	13/02/2005		

CÁN BỘ CHẤM THI

Hà Nội, năm 2025

LỜI CẢM ƠN

Trong quá trình thực hiện báo cáo môn học, chúng em đã nhận được sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình từ thầy cô, gia đình và bạn bè. Chúng em xin gửi lời cảm ơn chân thành đến giảng viên Bộ môn Trí tuệ Nhân tạo - Trường Đại học Đại Nam. Thầy đã tận tình hướng dẫn, chỉ bảo cho em trong suốt quá trình học tập và nghiên cứu. Em cũng xin chân thành cảm ơn các thầy cô giáo trong Trường Đại học Đại Nam nói chung, đặc biệt là các thầy cô trong Khoa Công nghệ Thông tin đã tận tâm giảng dạy, trang bị cho em những kiến thức cơ bản và chuyên ngành vững chắc. Nhờ đó, em có được hành trang lý thuyết vững vàng và tạo điều kiện thuận lợi để em hoàn thành tốt quá trình học tập. Một lần nữa, em xin chân thành cảm ơn!

Hà Nội, tháng 1 năm 2025

Mục lục

1 CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN

1.1 Giới thiệu về trí tuệ nhân tạo

1.1.1 Trí tuệ nhân tạo là gì?

Để hiểu trí tuệ nhân tạo (Artificial Intelligence) là gì chúng ta bắt đầu với khái niệm sự bay nhân tạo (flying machines), tức là cái máy bay. Đã từ lâu, loài người mong muốn làm ra một cái máy mà có thể di chuyển được trên không trung mà không phụ thuộc vào địa hình ở dưới mặt đất, hay nói cách khác là máy có thể bay được. Không có gì ngạc nhiên khi những ý tưởng đầu tiên làm máy bay là từ nghiên cứu cách con chim bay. Những chiếc máy biết bay được thiết kế theo nguyên lý "vỗ cánh" như con chim chỉ có thể bay được quãng đường rất ngắn và lịch sử hàng không thực sự sang một trang mới kể từ anh em nhà Wright thiết kế máy bay dựa trên các nguyên lý của khí động lực học (aerodynamics).

Các máy bay hiện nay, như đã thấy, có sức trở rất lớn và bay được quãng đường có thể vòng quanh thế giới. Nó không nhất thiết phải có nguyên lý bay của con chim nhưng vẫn bay được như chim (dáng vẻ), và còn tốt hơn chim. Quay lại câu hỏi Trí tuệ nhân tạo là gì. Trí tuệ nhân tạo là trí thông minh của máy do con người tạo ra. Ngay từ khi chiếc máy tính điện tử đầu tiên ra đời, các nhà khoa học máy tính đã hướng đến phát hiện hệ thống máy tính (gồm cả phần cứng và phần mềm) sao cho nó có khả năng thông minh như loài người. Mặc dù cho đến nay, theo quan niệm của người viết, ước mơ này vẫn còn xa mới thành hiện thực, tuy vậy những thành tựu đạt được cũng không hề nhỏ: chúng ta đã làm được các hệ thống (phần mềm chơi cờ vua chạy trên siêu máy tính Gene Blue) có thể thắng được vua cờ thế giới; chúng ta đã làm được các phần mềm có thể chứng minh được các bài toán hình học; v.v. Hay nói cách khác, trong một số lĩnh vực, máy tính có thể thực hiện tốt hơn hoặc tương đương con người (tất nhiên không phải tất cả các lĩnh vực). Đó chính là các hệ thống thông minh.

Có nhiều cách tiếp cận để làm ra trí thông minh của máy (hay là trí tuệ nhân tạo), chẳng hạn là nghiên cứu cách bộ não người sản sinh ra trí thông minh của loài người như thế nào rồi ta bắt chước nguyên lý đó, nhưng cũng có những cách khác sử dụng nguyên lý hoàn toàn khác với cách sản sinh ra trí thông minh của loài người mà vẫn làm ra cái máy thông minh như hoặc hơn người; cũng giống như máy bay hiện nay bay tốt hơn con chim do nó có cơ chế bay không phải là giống như cơ chế bay của con chim.

Như vậy, trí tuệ nhân tạo ở đây là nói đến khả năng của máy khi thực hiện các công việc mà con người thường phải xử lý; và khi dáng vẻ ứng xử hoặc kết quả thực hiện của máy là tốt hơn hoặc tương đương với con người thì ta gọi đó là máy thông minh hay máy đó có trí thông minh. Hay nói cách khác, đánh giá sự thông minh của máy không phải dựa trên nguyên lý nó thực hiện nhiệm vụ đó có giống cách con người thực hiện hay không mà dựa trên kết quả hoặc dáng vẻ ứng xử bên ngoài của nó có giống với kết quả hoặc dáng vẻ ứng xử của con người hay không. Các nhiệm vụ của con người thường xuyên phải thực hiện là: giải bài toán (tìm kiếm, chứng minh, lập luận), học, giao tiếp, thể hiện cảm xúc, thích nghi với môi trường xung quanh, v.v., và dựa trên kết quả thực hiện các nhiệm vụ đó để kết luận rằng một ai đó có là thông minh hay không. Môn học Trí tuệ nhân tạo nhằm cung cấp các phương pháp luận để làm ra hệ thống có khả năng thực hiện các nhiệm vụ đó: giải toán, học, giao tiếp, v.v. bất kể cách nó làm có như con người hay không mà là kết quả đạt được hoặc dáng vẻ bên ngoài như con người.

Khi máy móc ngày càng tăng khả năng, các nhiệm vụ được coi là cần "trí thông minh" thường bị loại bỏ khỏi định nghĩa về AI, một hiện tượng được gọi là hiệu ứng AI.

Một câu châm ngôn trong Định lý của Tesler nói rằng "AI là bất cứ điều gì chưa được thực hiện." Ví dụ, nhận dạng ký tự quang học thường bị loại trừ khỏi những thứ được coi là AI, đã trở thành một công nghệ thông thường. Khả năng máy hiện đại thường được phân loại như AI bao gồm thành công hiểu lời nói của con người, cạnh tranh ở mức cao nhất trong trò chơi chiến lược (chẳng hạn như cờ vua và Go), xe hoạt động độc lập, định tuyến thông minh trong mạng phân phối nội dung, và mô phỏng quân sự.

Trí tuệ nhân tạo có thể được phân thành ba loại hệ thống khác nhau: trí tuệ nhân tạo phân tích, lấy cảm hứng từ con người và nhân tạo. AI phân tích chỉ có các đặc điểm phù hợp với trí tuệ nhận thức; tạo ra một đại diện nhận thức về thế giới và sử dụng học tập dựa trên kinh nghiệm trong quá khứ để thông báo các quyết định trong tương lai. AI lấy cảm hứng từ con người có các yếu tố từ trí tuệ nhận thức và cảm xúc; hiểu cảm xúc của con người, ngoài các yếu tố nhận thức và xem xét chúng trong việc ra quyết định. AI nhân cách hóa cho thấy các đặc điểm của tất cả các loại năng lực (nghĩa là trí tuệ nhận thức, cảm xúc và xã hội), có khả năng tự ý thức và tự nhận thức được trong các tương tác.

Trí tuệ nhân tạo được thành lập như một môn học thuật vào năm 1956, và trong những năm sau đó đã trải qua nhiều làn sóng lạc quan, sau đó là thất vọng và mất kinh phí (được gọi là "mùa đông AI"), tiếp theo là cách tiếp cận mới, thành công và tài trợ mới. Trong phần lớn lịch sử của mình, nghiên cứu AI đã được chia thành các trường con thường không liên lạc được với nhau. Các trường con này dựa trên các cân nhắc kỹ thuật, chẳng hạn như các mục tiêu cụ thể (ví dụ: "robot học" hoặc "học máy"), việc sử dụng các công cụ cụ thể ("logic" hoặc mạng lưới thần kinh nhân tạo) hoặc sự khác biệt triết học sâu sắc. Các ngành con cũng được dựa trên các yếu tố xã hội (các tổ chức cụ thể hoặc công việc của các nhà nghiên cứu cụ thể).

Lĩnh vực này được thành lập dựa trên tuyên bố rằng trí thông minh của con người "có thể được mô tả chính xác đến mức một cỗ máy có thể được chế tạo để mô phỏng nó". Điều này làm dấy lên những tranh luận triết học về bản chất của tâm trí và đạo đức khi tạo ra những sinh vật nhân tạo có trí thông minh giống con người, đó là những vấn đề đã được thần thoại, viễn tưởng và triết học từ thời cổ đại đề cập tới. Một số người cũng coi AI là mối nguy hiểm cho nhân loại nếu tiến triển của nó không suy giảm. Những người khác tin rằng AI, không giống như các cuộc cách mạng công nghệ trước đây, sẽ tạo ra nguy cơ thất nghiệp hàng loạt. Trong thế kỷ 21, các kỹ thuật AI đã trải qua sự hồi sinh sau những tiến bộ đồng thời về sức mạnh máy tính, dữ liệu lớn và hiểu biết lý thuyết; và kỹ thuật AI đã trở thành một phần thiết yếu của ngành công nghệ, giúp giải quyết nhiều vấn đề thách thức trong học máy, công nghệ phần mềm và nghiên cứu vận hành. Trong môn học này, chúng ta sẽ tìm hiểu các phương pháp để làm cho máy tính biết cách giải bài toán, biết cách lập luận, biết học, v.v.

1.1.2 Lĩnh vực AI

Lập luận, suy diễn tự động: khái niệm lập luận (reasoning), và suy diễn (inference) được sử dụng rất phổ biến trong lĩnh vực AI. Lập luận là suy diễn logic, dùng để chỉ một tiến trình rút ra kết luận (tri thức mới) từ những giả thiết đã cho (được biểu diễn dưới dạng cơ sở tri thức). Như vậy, để thực hiện lập luận người ta cần có các phương pháp lưu trữ cơ sở tri thức và các thủ tục lập luận trên cơ sở tri thức đó.

Biểu diễn tri thức: Muốn máy tính có thể lưu trữ và xử lý tri thức thì cần có các phương pháp biểu diễn tri thức. Các phương pháp biểu diễn tri thức ở đây bao gồm các ngôn ngữ biểu diễn và các kỹ thuật xử lý tri thức. Một ngôn ngữ biểu diễn tri thức được đánh giá là “tốt” nếu nó có tính biểu đạt cao và các tính hiệu quả của thuật toán lập

luận trên ngôn ngữ đó. Tính biểu đạt của ngôn ngữ thể hiện khả năng biểu diễn một phạm vi rộng lớn các thông tin trong một miền ứng dụng. Tính hiệu quả của các thuật toán lập luận thể hiện chi phí về thời gian và không gian dành cho việc lập luận. Tuy nhiên, hai yếu tố này dường như đối nghịch nhau, tức là nếu ngôn ngữ có tính biểu đạt cao thì thuật toán lập luận trên đó sẽ có độ phức tạp lớn (tính hiệu quả thấp) và ngược lại (ngôn ngữ đơn giản, có tính biểu đạt thấp thì thuật toán lập luận trên đó sẽ có hiệu quả cao). Do đó, một thách thức lớn trong lĩnh vực AI là xây dựng các ngôn ngữ biểu diễn tri thức mà có thể cân bằng hai yếu tố này, tức là ngôn ngữ có tính biểu đạt đủ tốt (tùy theo từng ứng dụng) và có thể lập luận hiệu quả.

Lập kế hoạch: khả năng suy ra các mục đích cần đạt được đối với các nhiệm vụ đưa ra, và xác định dãy các hành động cần thực hiện để đạt được mục đích đó. Học máy: là một lĩnh vực nghiên cứu của AI đang được phát triển mạnh mẽ và có nhiều ứng dụng trong các lĩnh vực khác nhau như khai phá dữ liệu, khám phá tri thức...

Xử lý ngôn ngữ tự nhiên: là một nhánh của AI, tập trung vào các ứng dụng trên ngôn ngữ của con người. Các ứng dụng trong nhận dạng tiếng nói, nhận dạng chữ viết, dịch tự động, tìm kiếm thông tin...

Hệ chuyên gia: cung cấp các hệ thống có khả năng suy luận để đưa ra những kết luận. Các hệ chuyên gia có khả năng xử lý lượng thông tin lớn và cung cấp các kết luận dựa trên những thông tin đó. Có rất nhiều hệ chuyên gia nổi tiếng như các hệ chuyên gia y học MYCIN, đoán nhận cấu trúc phân tử từ công thức hóa học DENDRAL...

1.1.3 Tác động

Sau khi nhà vật lý học Stephen Hawking và Elon Musk cảnh báo về mối đe dọa tiềm ẩn của trí tuệ nhân tạo, nhiều người cho rằng họ đã quá lo xa trong khi AI đang giúp ích rất nhiều cho cuộc sống của chúng ta. Stephen Hawking khẳng định "Trí tuệ nhân tạo có thể là dấu chấm hết cho nhân loại khi nó phát triển đến mức hoàn thiện nhất".

Tác động đầu tiên của trí tuệ nhân tạo mà chúng ta có thể dễ dàng nhận thấy chính là tỷ lệ thất nghiệp tăng cao. Nếu AI phát triển hoàn thiện, nó có khả năng thay thế con người trong các công việc trí tuệ như chăm sóc sức khỏe, phục vụ, sản xuất theo dây chuyền tự động, công việc văn phòng.... Hoặc cũng có thể vấn đề thất nghiệp sẽ được AI giải quyết một cách mà chúng ta không thể hình dung được. Theo Bill Joy, người đồng sáng lập và Giám đốc khoa học của Sun Microsystems: "Có một vấn đề rất lớn đối với xã hội loài người khi AI trở nên phổ biến, đó là chúng ta sẽ bị lệ thuộc. Khi AI trở nên hoàn thiện và thông minh hơn, chúng ta sẽ cho phép mình nghe theo những quyết định của máy móc, vì đơn giản là các cỗ máy luôn đưa ra quyết định chính xác hơn con người."

Theo Andrew Maynard, nhà vật lý và là người giám đốc Trung tâm nghiên cứu rủi ro khoa học tại đại học Michigan: "Khi AI kết hợp với công nghệ nano có thể là bước tiến đột phá của khoa học, nhưng cũng có thể là mối đe dọa lớn nhất đối với con người. Trong khi Bộ quốc phòng Mỹ đang nghiên cứu dự án Autonomous Tactical Robot (EATR), trong đó các robot sẽ sử dụng công nghệ nano để hấp thụ năng lượng bằng những chất hữu cơ có thể là cơ thể con người. Đó thực sự là mối đe dọa lớn nhất, khi các robot nano tự tạo ra năng lượng bằng cách ăn các chất hữu cơ từ cây cối và động vật, có thể là cả con người. Nghe có vẻ giống như trong các bộ phim viễn tưởng, nhưng đó là điều hoàn toàn có thể xảy ra. Có lẽ chúng ta nên bắt đầu cẩn thận ngay từ bây giờ."

1.1.4 Thuật toán quay lui (Backtracking)

Quay lui hay Backtracking là một kỹ thuật thiết kế giải thuật dựa trên đệ quy. Ý tưởng của quay lui là tìm lời giải từng bước, mỗi bước chọn một trong số các lựa chọn khả dĩ

và đệ quy. Người đầu tiên đề ra thuật ngữ này (backtrack) là nhà toán học người Mỹ D. H. Lehmer vào những năm 1950.

Quay lui dùng để giải bài toán liệt kê các cấu hình. Mỗi cấu hình được xây dựng bằng từng phần tử. Mỗi phần tử lại được chọn bằng cách thử tất cả các khả năng.

Dùng để giải bài toán liệt kê các cấu hình. Mỗi cấu hình được xây dựng bằng từng phần tử. Mỗi phần tử lại được chọn bằng cách thử tất cả các khả năng. Các bước trong việc liệt kê cấu hình dạng X[1..n]:

- Xét tất cả các giá trị X[1] có thể nhận, thử X[1] nhận các giá trị đó. Với mỗi giá trị của X[1] ta sẽ:
- Xét tất cả giá trị X[2] có thể nhận, lại thử X[2] cho các giá trị đó. Với mỗi giá trị X[2] lại xét khả năng giá trị của X[3]...tiếp tục như vậy cho tới bước:
- Xét tất cả giá trị X[n] có thể nhận, thử cho X[n] nhận lần lượt giá trị đó.
- Thông báo cấu hình tìm được.

Bản chất của quay lui là một quá trình tìm kiếm theo chiều sâu (Depth-First Search).

Đoạn code cho thuật toán quay lui: (Lưu ý: Đoạn mã gốc không được cung cấp đầy đủ, vì vậy tôi để trống, bạn có thể bổ sung nếu có.)

Ưu điểm: Việc quay lui là thử tất cả các tổ hợp để tìm được một lời giải. Thế mạnh của phương pháp này là nhiều cài đặt tránh được việc phải thử nhiều trường hợp chưa hoàn chỉnh, nhờ đó giảm thời gian chạy. Nhược điểm: Trong trường hợp xấu nhất độ phức tạp của quay lui vẫn là cấp số mũ. Vì nó mắc phải các nhược điểm sau:

- Rơi vào tình trạng "thrashing": quá trình tìm kiếm cứ gặp phải bế tắc với cùng một nguyên nhân.
- Thực hiện các công việc dư thừa: Mỗi lần chúng ta quay lui, chúng ta cần phải đánh giá lại lời giải trong khi đôi lúc điều đó không cần thiết.
- Không sớm phát hiện được các khả năng bị bế tắc trong tương lai. Quay lui chuẩn, không có cơ chế nhìn về tương lai để nhận biết được nhánh tìm kiếm sẽ đi vào bế tắc.

Sudoku có tên gọi tiếng Anh là Number Place, là một trò chơi câu đố sắp xếp chữ số dựa trên logic theo tổ hợp. Sudoku sẽ được cho sẵn một vài con số và nằm ở những vị trí bất kỳ. Nhiệm vụ của người chơi là điền các chữ số vào một lưới 9×9 sao cho mỗi hàng, mỗi cột, và cả mỗi phần trong số chín lưới con 3×3 cấu tạo nên lưới chính đều chứa tất cả các chữ số từ 1 tới 9.

Sudoku xuất hiện đầu tiên ở Mỹ với tên gọi là “Đặt vị trí số” - Number Place. Sau đó nó được du nhập vào Nhật Bản và được nhà xuất bản Nikoli đổi tên thành Sudoku có nghĩa là duy nhất bởi mỗi ô chỉ có một con số duy nhất. Trải qua thời gian, Sudoku đã trở thành trò chơi trí tuệ được yêu thích tại nhiều quốc gia.

Sudoku có rất nhiều biến thể nhưng chỉ thay đổi về kích thước và số lượng ô trong trò chơi còn lối chơi cơ bản vẫn giữ nguyên. Ở phiên bản chuẩn (bản gốc) chỉ có kích thước là 9x9 (ô nhỏ) và được chia thành 9 vùng, mỗi vùng có kích thước 3x3. Các vùng này được nhóm lại và phân tách với nhau bằng một viền đen đậm hơn so với các ô nhỏ. Luật chơi của Sudoku là điền kín những ô còn lại với điều kiện:

- Các hàng ngang: Phải có đủ các số từ 1 đến 9, không trùng số và không cần đúng thứ tự.
- Các hàng dọc: Đảm bảo có đủ các số từ 1-9, không trùng số, không cần theo thứ tự.
- Mỗi vùng 3 x 3: Phải có đủ các số từ 1-9 và không trùng số nào trong cùng 1 vùng 3x3.

Chương trình giải dựa trên thuật toán quay lui. Bằng việc liệt kê các tình huống, thử các khả năng có thể cho đến khi tìm thấy một lời giải đúng, thuật toán quay lui chia nhỏ bài toán, lời giải của bài toán lớn sẽ là kết quả của việc tìm kiếm theo chiều sâu của tập hợp các bài toán phần tử. Trong suốt quá trình tìm kiếm nếu gặp phải một hướng nào đó mà biết chắc không thể tìm thấy đáp án thì quay lại bước trước đó và tìm hướng khác kế tiếp hướng vừa tìm kiếm đó. Trong trường hợp không còn một hướng nào khác nữa thì thuật toán kết thúc.

2 CHƯƠNG 2: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT BÀI TOÁN

2.1 2.1. Phát biểu bài toán

Đầu vào: Đề Sudoku đọc bảng từ file, hoặc do người dùng nhập. Đầu ra (Kết quả): Đáp án có thể là các bước giải Sudoku của chương trình hoặc đáp án toàn bộ chương trình (có thể ra nhiều đáp án nếu đầu vào có nhiều đáp án).

2.2 2.2. Vấn đề

Chương trình giải dựa trên thuật giải quay lui. Tư tưởng của thuật giải này là chia nhỏ bài toán lớn thành các bài toán phần tử, giải bài toán phần tử đó, ứng với mỗi trường hợp giải được của bài toán phần tử đó, ta đi tìm lời giải cho bài toán phần tử tiếp theo cho đến khi bài toán lớn trở nên đầy đủ.

Ta nhận xét rằng: Khi đang xét vị trí thứ i, ta đưa ra các phương án để tiếp tục xét vị trí i+1, có những phương án sẽ làm cho vị trí i+1, i+2... có thể tìm tiếp phương án và dẫn đến quả cuối cùng (Gọi là phương án khả thi) và có những phương án sẽ không có kết quả (gọi là phương án bất khả thi). Vậy thì để chương trình chạy nhanh, ta cần phải loại bỏ các phương án bất khả thi càng nhiều càng tốt. Thuật toán được biểu diễn bằng đệ quy nhưng nếu ta cài đặt bằng đệ quy thì sẽ không có lợi, phải sử dụng bộ nhớ stack, gọi hàm đệ quy nhiều lần, điều đó làm chương trình sẽ chạy rất chậm. Vì vậy ta có thể cài đặt bằng không đệ quy nhưng mà tinh thần giải thì vẫn dựa trên phương pháp đệ quy. Để làm được điều này, cần phải có cách sao cho có thể di chuyển và thử các khả năng trên các ô được dễ dàng.

2.3 2.3. Cấu trúc dữ liệu

2.3.1 2.3.1. Cấu trúc dữ liệu chính

- `bang`: Một mảng hai chiều 9×9 đại diện cho bảng Sudoku hiện tại. Giá trị 0 đại diện cho các ô trống. Các số từ 1-9 đại diện cho các giá trị đã được điền.
- `bang_goc`: Sao chép của bảng ban đầu sau khi tạo đề bài. Dùng để kiểm tra nếu người chơi thay đổi giá trị không được phép.
- `o_nhap`: Một mảng hai chiều lưu trữ các đối tượng Entry (ô nhập liệu trong giao diện). Mỗi ô này tương ứng với một ô trong bảng Sudoku.

2.3.2 2.3.2. Các hàm chức năng chính

- **Sinh bảng Sudoku và tạo đề bài**
  - `sinh_bang_day_du()`: Sử dụng thuật toán đệ quy và backtracking để tạo bảng Sudoku đầy đủ (bảng đã có lời giải). Kiểm tra tính hợp lệ của các số được điền bằng hàm con `la_hop_le`.
  - `xoa_so(bang_day_du, muc_do)`: Tạo đề bài bằng cách xóa một số ô ngẫu nhiên khỏi bảng đầy đủ. Số ô bị xóa tùy thuộc vào mức độ khó: "Dễ", "Trung bình", "Khó".
- **Cập nhật và kiểm tra bảng**

  - `cap_nhat_bang()`: Lấy giá trị người dùng nhập từ giao diện và cập nhật vào mảng `bang`.
  - `kiem_tra_dap_an()`: Kiểm tra tính hợp lệ của bảng Sudoku hiện tại: Không trùng giá trị trong hàng, cột, và lưới 3×3. Đổi màu ô nhập liệu dựa trên tính đúng/sai: Màu xanh lá: Đúng. Màu đỏ: Sai.

- **Tương tác giao diện**
  - `dat_lai_bang()`: Tạo bảng Sudoku mới với mức độ khó được chọn. Làm mới giao diện hiển thị.
  - `to_sang_lien_quan(event)`: Tô sáng hàng, cột, và lưới con 3×3 liên quan đến ô đang được chọn.

2.3.3 2.3.3. Giao diện người dùng

- **Giao diện chính (`root`)**: Khung lớn chứa các thành phần giao diện. Tùy chỉnh màu sắc, kích thước, và bố cục.
- **Bảng Sudoku**: 9×9 ô nhập liệu, chia thành 9 lưới con (3×3) để dễ nhìn. Ô nhập liệu có sự kiện liên quan đến tương tác, như tô sáng khi người dùng chọn ô.
- **Nút chức năng**:
  - Kiểm tra: Gọi `kiem_tra_dap_an` để kiểm tra lời giải.
  - Làm mới: Gọi `dat_lai_bang` để khởi tạo bảng Sudoku mới.
  - Thoát: Thoát chương trình.
- **Menu chọn mức độ khó**: Người dùng có thể chọn mức độ khó: "Dễ", "Trung bình", hoặc "Khó". Giá trị được lưu trong biến `difficulty`.

2.3.4 2.3.4. Thuật toán chính

- **Backtracking**: Dùng để sinh bảng Sudoku đầy đủ. Thử các giá trị từ 1-9 cho từng ô trống, kiểm tra tính hợp lệ, và quay lui nếu cần.
- "Kiểm tra hợp lệ":
  - Hàng (Rows): Không được có số trùng lặp.
  - Cột (Columns): Không được có số trùng lặp.
  - Lưới con 3×3 (Boxes): Không được có số trùng lặp.

2.4 2.4. Thuật toán quay lui

Thuật toán quay lui (backtracking) như tên gọi của nó, là một quá trình tìm kiếm mà trong quá trình tìm kiếm, nếu ta gặp một hướng lựa chọn không thỏa mãn, ta quay lui về điểm lựa chọn nơi có các hướng khác và thử hướng lựa chọn tiếp theo. Quá trình tìm kiếm thất bại khi không còn điểm lựa chọn nào nữa.

Độ phức tạp: Trong trường hợp xấu nhất độ phức tạp của quay lui vẫn là cấp số mũ. Vì nó mắc phải các nhược điểm sau:

- Rơi vào tình trạng "thrashing": quá trình tìm kiếm gặp phải bế tắc với cùng một nguyên nhân.
- Thực hiện các công việc dư thừa: Mỗi lần chúng ta quay lui, chúng ta cần phải đánh giá lại lời giải trong khi đôi lúc điều đó không cần thiết.
- Không sớm phát hiện được các khả năng bị bế tắc trong tương lai. Quay lui chuẩn, không có cơ chế nhìn về tương lai để nhận biết được nhánh tìm kiếm sẽ đi vào bế tắc.

3 CHƯƠNG 3: CÀI ĐẶT THUẬT TOÁN VÀ ĐÁNH GIÁ THỬ NGHIỆM

3.1 3.1. Code bài toán Sudoku

```python
import tkinter as tk
from tkinter import messagebox
import random
import time

# Ham sinh bang Sudoku day du bang thuat toan backtracking
def sinh_bang_day_du():
    bang = [[0] * 9 for _ in range(9)]

    def la_hop_le(so, hang, cot):
        for i in range(9):
            if bang[hang][i] == so or bang[i][cot] == so:
                return False
        sub_hang, sub_cot = 3 * (hang // 3), 3 * (cot // 3)
        for i in range(sub_hang, sub_hang + 3):
            for j in range(sub_cot, sub_cot + 3):
                if bang[i][j] == so:
                    return False
        return True

    def giai():
        for i in range(9):
            for j in range(9):
                if bang[i][j] == 0:
                    so_ngau_nhien = list(range(1, 10))
                    random.shuffle(so_ngau_nhien)
                    for so in so_ngau_nhien:
                        if la_hop_le(so, i, j):
                            bang[i][j] = so
                            if giai():
                                return True
                            bang[i][j] = 0
                    return False
        return True

    giai()
    return bang

# Ham xoa so ngau nhien de tao de bai
def xoa_so(bang_day_du, muc_do):
    bang = [row[:] for row in bang_day_du]
    so_can_xoa = {"De": 30, "Trung binh": 40, "Kho": 50}[muc_do]
    dem = 0
    while dem < so_can_xoa:
        hang, cot = random.randint(0, 8), random.randint(0, 8)
        if bang[hang][cot] != 0:
            bang[hang][cot] = 0
            dem += 1
    return bang

# Ham dat lai bang voi muc do kho
```

```
def dat_lai_bang():
    global bang, bang_goc, start_time
    muc_do = difficulty.get()
    bang_day_du = sinh_bang_day_du()
    bang = xoa_so(bang_day_du, muc_do)
    bang_goc = [row[:] for row in bang]
    start_time = time.time()

    for i in range(9):
        for j in range(9):
            o_nhap[i][j].delete(0, tk.END)
            if bang[i][j] != 0:
                o_nhap[i][j].insert(0, str(bang[i][j]))
                o_nhap[i][j].config(fg="black", bg="white")
            else:
                o_nhap[i][j].config(fg="blue", bg="white")

# Ham cap nhat du lieu tu cac o nhap lieu vao bang `bang`
def cap_nhat_bang():
    for i in range(9):
        for j in range(9):
            try:
                gia_tri = int(o_nhap[i][j].get())
                bang[i][j] = gia_tri
            except ValueError:
                bang[i][j] = 0

# Ham kiem tra dung/sai
def kiem_tra_dap_an():
    cap_nhat_bang()
    dung = True
    for i in range(9):
        for j in range(9):
            if bang[i][j] != 0:
                hang_hop_le = bang[i].count(bang[i][j]) == 1
                cot_hop_le = [bang[x][j] for x in range(9)].count(bang[
    i][j]) == 1
                sub_hang, sub_cot = i // 3 * 3, j // 3 * 3
                luoi_con = [bang[x][y] for x in range(sub_hang,
    sub_hang + 3) for y in range(sub_cot, sub_cot + 3)]
                luoi_hop_le = luoi_con.count(bang[i][j]) == 1

                if hang_hop_le and cot_hop_le and luoi_hop_le:
                    o_nhap[i][j].config(bg="lightgreen")
                else:
                    o_nhap[i][j].config(bg="red")
                    dung = False

# Su kien to sang hang, cot va luoi con khi nhan vao o
def to_sang_lien_quan(event):
    cap_nhat_bang()
    for i in range(9):
        for j in range(9):
            o_nhap[i][j].config(bg="white")
    widget = event.widget
    index = None
    for i, row in enumerate(o_nhap):
        if widget in row:
```

```
            index = (i, row.index(widget))
            break
    if index:
        hang, cot = index
        for i in range(9):
            o_nhap[hang][i].config(bg="lightyellow")
            o_nhap[i][cot].config(bg="lightyellow")

# Giao dien chinh
root = tk.Tk()
root.title("Thuat giai game Sudoku")
root.geometry("700x800")
root.configure(bg="lightgray")

# Bien toan cuc
start_time = None
difficulty = tk.StringVar(value="Trung binh")

# Tieu de
tieu_de = tk.Label(root, text="Sudoku", font=("Arial", 16), bg="
    lightgray")
tieu_de.pack(pady=10)

# Tao bang Sudoku
khung = tk.Frame(root, bg="lightgray")
khung.pack(pady=20)
o_nhap = []
for sub_hang in range(3):
    for sub_cot in range(3):
        khung_con = tk.Frame(khung, relief="ridge", bd=1, bg="black")
        khung_con.grid(row=sub_hang, column=sub_cot, padx=3, pady=3)
        for i in range(3):
            for j in range(3):
                hang_chinh = sub_hang * 3 + i
                cot_chinh = sub_cot * 3 + j
                o = tk.Entry(khung_con, width=3, font=("Arial", 18),
    justify="center", relief="flat", bd=1)
                o.grid(row=i, column=j, padx=1, pady=1, ipady=5)
                if len(o_nhap) <= hang_chinh:
                    o_nhap.append([])
                o_nhap[hang_chinh].append(o)
                o.bind("<FocusIn>", to_sang_lien_quan)

# Nut chuc nang va chon muc do
khung_nut = tk.Frame(root, bg="lightgray")
khung_nut.pack(pady=10)

nut_kiem_tra = tk.Button(khung_nut, text="Kiem tra", command=
    kiem_tra_dap_an, bg="lightblue", font=("Arial", 14))
nut_kiem_tra.grid(row=0, column=0, padx=10)

nut_dat_lai = tk.Button(khung_nut, text="Lam moi", command=dat_lai_bang
    , bg="lightgreen", font=("Arial", 14))
nut_dat_lai.grid(row=0, column=1, padx=10)

nut_thoat = tk.Button(khung_nut, text="Thoat", command=root.quit, bg="
    red", font=("Arial", 14), fg="white")
nut_thoat.grid(row=0, column=2, padx=10)
```

```
161
162 # Dropdown menu chon muc do
163 menu_chon_muc_do = tk.OptionMenu(khung_nut, difficulty, "De", "Trung
    binh", "Kho")
164 menu_chon_muc_do.config(bg="orange", font=("Arial", 14))
165 menu_chon_muc_do.grid(row=0, column=3, padx=10)
166
167 # Khoi tao bang dau tien
168 dat_lai_bang()
169 root.mainloop()
```

Listing 1: Code bai toan Sudoku

3.2 3.2. Giao diện chính

Bài toán được đưa lên web để dễ tiếp cận mọi người. Dễ thao tác.

Ưu điểm:

- **Tổ chức rõ ràng và trực quan:** Bảng Sudoku được chia thành các khối 3×3 với đường viền rõ ràng, giúp người dùng dễ dàng quan sát và nhập liệu.
- **Các ô nhập liệu có màu sắc phân biệt:**
  - Màu xanh dương: Ô trống do hệ thống tạo.
  - Màu trắng: Ô được điền sẵn trong đề bài.
  - Màu xanh lá nhạt hoặc đỏ: Thông báo trực quan khi kiểm tra đáp án.
- **Tính năng hỗ trợ người dùng:**
  - Thời gian chạy: Có đồng hồ đếm thời gian trực tiếp hiển thị ở đầu giao diện, giúp người dùng theo dõi thời gian giải Sudoku.
  - Tô sáng hàng, cột và khối: Khi người dùng chọn một ô, hàng, cột và khối 3×3 liên quan sẽ được tô màu để tăng khả năng tập trung.
- **Tùy chọn mức độ khó:** Giao diện cho phép người dùng chọn mức độ khó (Dễ, Trung bình, Khó) trước khi khởi tạo bảng Sudoku mới.
- **Giao diện thân thiện:** Giao diện có các nút chức năng được sắp xếp gọn gàng với màu sắc dễ nhìn: Nút Kiểm tra: Màu xanh dương. Nút Làm mới: Màu xanh lá. Nút Thoát: Màu đỏ.

Nhược điểm còn tồn tại:

- Giao diện hơi tĩnh: Mặc dù giao diện đáp ứng đầy đủ tính năng cơ bản, nhưng chưa có yếu tố thẩm mỹ cao hoặc sự tương tác động (animation, hiệu ứng).
- Font chữ và thiết kế tổng thể còn đơn giản, có thể thêm các yếu tố đồ họa để tạo cảm giác hiện đại hơn.
- Hạn chế về kiểm tra lỗi: Khi nhập số không hợp lệ (chẳng hạn, số lớn hơn 9 hoặc ký tự không phải số), giao diện không hiển thị thông báo lỗi rõ ràng.
- Khả năng mở rộng hạn chế: Hiện tại, giao diện chỉ hỗ trợ giải Sudoku và kiểm tra kết quả. Có thể thêm các tính năng bổ sung như gợi ý (hint) hoặc hiển thị lời giải.

3.3 3.3. Giao diện chọn mức độ

Thành phần giao diện:

- **Menu thả xuống (Dropdown):** Menu này cho phép người dùng chọn mức độ khó của bài Sudoku trước khi tạo bảng mới. Các tùy chọn trong menu bao gồm: "Dễ", "Trung bình", "Khó".
- **Vị trí:** Menu thả xuống được đặt trong một khung (frame) cùng với các nút chức năng khác, nằm phía dưới bảng Sudoku, giúp người chơi dễ dàng truy cập.
- **Phong cách:** Màu nền của menu là màu cam. Font chữ sử dụng là Arial, cỡ chữ 14, đảm bảo dễ đọc và dễ thao tác.
- **Cách hoạt động:** Người dùng nhấp vào menu thả xuống để hiển thị danh sách các mức độ khó. Khi chọn một mức độ (ví dụ: "Trung bình"), mức độ này sẽ được lưu trong biến `difficulty`. Khi nhấn nút "Làm mới", chương trình sẽ tạo lại bảng Sudoku mới theo mức độ khó đã chọn.

Mã liên quan:

```
# Dropdown menu chon muc do
menu_chon_muc_do = tk.OptionMenu(khung_nut, difficulty, "De", "Trung
    binh", "Kho")
menu_chon_muc_do.config(bg="orange", font=("Arial", 14))
menu_chon_muc_do.grid(row=0, column=3, padx=10)
```

3.4 3.4. Giao diện kiểm tra

Thành phần giao diện:

- **Nút "Kiểm tra":** Đây là một nút bấm được thiết kế để người chơi kiểm tra đáp án của mình sau khi đã điền các số vào bảng Sudoku.
- **Vị trí:** Nút được đặt trong khung chức năng (frame) cùng với các nút khác như "Làm mới" và "Thoát", ngay dưới bảng Sudoku.
- **Phong cách:** Màu nền: Xanh dương nhạt (lightblue). Font chữ: Arial, cỡ chữ 14. Kích thước nút: Vừa phải, dễ dàng bấm vào trên hầu hết các màn hình.
- **Mã liên quan:**

```
nut_kiem_tra = tk.Button(khung_nut, text="Kiem tra", command=
    kiem_tra_dap_an, bg="lightblue", font=("Arial", 14))
nut_kiem_tra.grid(row=0, column=0, padx=10)

```

Chức năng của nút "Kiểm tra": Khi nhấn vào nút "Kiểm tra", chương trình sẽ thực hiện các thao tác sau:

- Cập nhật bảng Sudoku từ các ô nhập liệu: Gọi hàm $\texttt{cap}_n hat_b ang()Ølygitrtttcccngichi Ønhpvcpnht$ $MiØckimtratheoquytcSudoku : GitrkhngØctrnglptrongcnghng.GitrkhngØctrnglptrongcngct.G$
- Đánh dấu màu sắc cho các ô: Màu xanh lá nhạt (lightgreen): Ô hợp lệ (đúng quy tắc). Màu đỏ (red): Ô không hợp lệ (vi phạm quy tắc Sudoku).

- Kết quả kiểm tra: Nếu tất cả các ô đều hợp lệ, giao diện sẽ đánh dấu toàn bộ bảng bằng màu xanh lá nhạt. Nếu có ô không hợp lệ, các ô đó sẽ được tô màu đỏ và giữ nguyên các ô khác.

Mã liên quan:

```
# Ham kiem tra dung/sai
def kiem_tra_dap_an():
    cap_nhat_bang()
    dung = True
    for i in range(9):
        for j in range(9):
            if bang[i][j] != 0:
                hang_hop_le = bang[i].count(bang[i][j]) == 1
                cot_hop_le = [bang[x][j] for x in range(9)].count(bang[
    i][j]) == 1
                sub_hang, sub_cot = i // 3 * 3, j // 3 * 3
                luoi_con = [bang[x][y] for x in range(sub_hang,
    sub_hang + 3) for y in range(sub_cot, sub_cot + 3)]
                luoi_hop_le = luoi_con.count(bang[i][j]) == 1

                if hang_hop_le and cot_hop_le and luoi_hop_le:
                    o_nhap[i][j].config(bg="lightgreen")
                else:
                    o_nhap[i][j].config(bg="red")
                    dung = False
```

KẾT LUẬN

Học phần này đã giúp chúng ta tiếp cận và làm quen với các thuật toán giải quyết vấn đề, đặc biệt là kỹ thuật lập trình nâng cao như đệ quy, backtracking và các thao tác xử lý giao diện đồ họa với thư viện Tkinter. Đồng thời, nó cũng củng cố kiến thức về cấu trúc dữ liệu mảng hai chiều và cách tổ chức dữ liệu hiệu quả trong lập trình thực tế. Đề tài Sudoku là một bài toán thú vị, mang tính ứng dụng cao, giúp người học vận dụng kiến thức để xây dựng thuật toán sinh bảng hợp lệ, kiểm tra quy tắc, cũng như thiết kế giao diện tương tác với người dùng.

Trong quá trình thực hiện, chúng ta đã xây dựng được thuật toán hoàn chỉnh để sinh bảng Sudoku hợp lệ bằng phương pháp backtracking, đồng thời tạo ra các đề bài với nhiều mức độ khó khác nhau. Chức năng kiểm tra đáp án cũng được tích hợp dựa trên các quy tắc Sudoku, giúp người chơi dễ dàng nhận biết ô đúng và sai thông qua màu sắc. Giao diện chương trình được thiết kế thân thiện, cho phép người dùng chọn mức độ khó, làm mới bảng, kiểm tra đáp án và thoát chương trình, đảm bảo trải nghiệm tốt và dễ sử dụng.

Từ đề tài này, chúng ta rút ra nhiều bài học kinh nghiệm quan trọng. Về mặt kỹ thuật, việc tìm hiểu và áp dụng thuật toán backtracking đã giúp cải thiện tư duy lập trình, đồng thời tối ưu hóa cấu trúc dữ liệu và thao tác xử lý giao diện đồ họa. Ngoài ra, quá trình lập trình cũng giúp nâng cao kỹ năng debug, xử lý lỗi, và tối ưu hóa mã nguồn. Về tư duy lập trình, việc phân tích bài toán từ lý thuyết đến thực tiễn giúp hiểu rõ hơn quy trình phát triển một ứng dụng hoàn chỉnh.

Mặc dù ứng dụng đã đáp ứng được các yêu cầu cơ bản, vẫn có nhiều hướng để cải thiện và mở rộng. Chúng ta có thể thêm chức năng gợi ý ô trống, lưu và tải trạng thái trò chơi hoặc thậm chí phát triển Sudoku thành một ứng dụng đầy đủ với bảng xếp hạng và chế độ chơi trực tuyến. Về giao diện, việc sử dụng thư viện hiện đại hơn như PyQt hoặc Kivy có thể giúp tạo trải nghiệm chuyên nghiệp hơn, đồng thời kết hợp âm thanh và hiệu ứng động để tăng tính hấp dẫn cho trò chơi.

Liên quan đến Trí tuệ Nhân tạo (AI), bài toán Sudoku có thể được mở rộng bằng cách áp dụng các thuật toán AI hiện đại để tìm kiếm và tối ưu lời giải. Một số phương pháp có thể được ứng dụng gồm:

- Thuật toán ràng buộc (Constraint Satisfaction Problem - CSP) để tìm lời giải hợp lệ một cách hiệu quả hơn so với backtracking truyền thống.
- Học máy (Machine Learning) có thể được sử dụng để huấn luyện mô hình giải Sudoku bằng cách nhận diện mẫu từ dữ liệu bài toán trước đó.
- Mạng nơ-ron nhân tạo (Neural Networks) có thể được áp dụng để xây dựng hệ thống giải Sudoku mà không cần sử dụng các quy tắc cứng nhắc.

Môn học Trí tuệ Nhân tạo giúp sinh viên tiếp cận các mô hình thông minh hơn để giải quyết các bài toán như Sudoku một cách tự động và hiệu quả hơn. Nếu kết hợp AI vào chương trình Sudoku, chúng ta có thể tạo ra một hệ thống không chỉ giải bài toán mà còn có thể tự động phân tích mức độ khó, đưa ra gợi ý hoặc thậm chí tạo ra các bảng Sudoku mới dựa trên mô hình sinh dữ liệu. Điều này mở ra nhiều ứng dụng trong thực tế, từ trò chơi trí tuệ, ứng dụng giáo dục, đến các bài toán phức tạp hơn như tối ưu hóa dữ liệu, xếp lịch, và giải quyết bài toán logic trong AI.

Nhìn chung, đề tài này không chỉ là một bài toán thú vị trong lĩnh vực thuật toán, mà còn có giá trị nghiên cứu và ứng dụng trong Trí tuệ Nhân tạo. Qua việc thực hiện

bài toán này, sinh viên ngành CNTT không chỉ rèn luyện kỹ năng lập trình mà còn có cơ hội tiếp cận với tư duy thuật toán nâng cao và các phương pháp thông minh hơn để giải quyết vấn đề trong AI. Đây là một bước quan trọng giúp sinh viên phát triển tư duy sáng tạo và chuẩn bị cho các nghiên cứu và ứng dụng AI trong tương lai.

DANH MỤC TÀI LIỆU THAM KHẢO

1. https://www.w3schools.com/Js/js_function_call.asp

2. https://vi.wikipedia.org/wiki/Sudoku

3. https://www.geeksforgeeks.org/backtracking-introduction/

4. Nguyễn Văn Hiệu (2018), *Trí tuệ nhân tạo và ứng dụng*, NXB Khoa học và Kỹ thuật.

5. Phạm Văn Hải (2020), *Thuật toán và lập trình*, NXB Đại học Quốc gia Hà Nội.

6. Nguyễn Thanh Bình (2019), *Xử lý dữ liệu và thuật toán tối ưu trong lập trình*, NXB Bách Khoa Hà Nội.

7. Đỗ Năng Toàn (2021), *Học máy và ứng dụng trong trí tuệ nhân tạo*, NXB Thông tin và Truyền thông.