TRƯỜNG ĐẠI HỌC HỌC VĂN LANG

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

NGÀNH: CÔNG NGHỆ THÔNG TIN

***Tên Đồ án:***

**ÁP DỤNG THUẬT TOÁN GIẢI THUẬT MINIMAX - MỞ RỘNG XO GAME 6X6 & XO GAME 8X8**

**SVTH: ĐỖ MINH QUÂN**

**MSSV: 207CT10276**

**GVHD: TRẦN NGỌC VIỆT**

Hồ Chí Minh – năm 2025

LỜI CẢM ƠN  
-------------

Viết một báo cáo đồ án môn học là một trong những việc khó nhất mà chúng em phải  
hoàn thành trong quá trình học một môn học. Trong quá trình thực hiện đề tài chúng  
em đã gặp rất nhiều khó khăn và bỡ ngỡ. Nếu không có những sự giúp đỡ và lời động  
viên chân thành của nhiều người có lẽ chúng em khó có thể hoàn thành tốt tiểu luận  
này. Đầu tiên chúng em xin gửi lời biết ơn chân thành đến thầy Trần Ngọc Việt,  
người trực tiếp hướng dẫn chúng em hoàn thành tiểu luận này.  
Những ý kiến đóng góp của thầy là vô cùng hữu ích, nó giúp chúng em nhận ra các  
khuyết điểm của đồ án. Cảm ơn thầy và các bạn trường Đại học Văn Lang là những  
người đã cùng nhóm em sát cánh và trải nghiệm để hoàn thành đồ án môn học.  
Nhóm thực hiện báo cáo

MỤC LỤC

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT – ÁP DỤNG THUẬT TOÁN GIẢI THUẬT MINIMAX - MỞ RỘNG XO GAME 6X6 & XO GAME 8X8

**1.1. Lý do chọn đề tài**

Trong thời đại hiện nay, Trí tuệ nhân tạo (Artificial Intelligence - AI) đang dần trở thành một lĩnh vực chủ chốt với tiềm năng ứng dụng rộng rãi trong nhiều ngành nghề. Một trong những bài toán kinh điển và mang tính giáo dục cao trong lĩnh vực này là các trò chơi hai người đối kháng như cờ vua, cờ caro, hay Tic-Tac-Toe (XO). Các trò chơi này cho phép tôi nghiên cứu và áp dụng các giải thuật tìm kiếm, ra quyết định, đánh giá trạng thái,… một cách trực quan.

Tuy nhiên, trò chơi XO truyền thống trên bảng 3x3 đã quá đơn giản, dễ dàng giải quyết bằng các thuật toán cơ bản. Vì vậy, tôi chọn mở rộng trò chơi XO lên kích thước lớn hơn như 6x6 và 8x8 để tăng độ phức tạp, yêu cầu thuật toán phải thực sự “suy nghĩ” trước khi đưa ra quyết định.

Đồng thời, tôi muốn triển khai một ứng dụng thực tế chạy được, có khả năng tương tác với người chơi qua bàn phím, và AI có thể phản hồi thông minh. Việc kết hợp giữa kiến thức lý thuyết về AI và thực hành lập trình thuật toán Minimax là lý do chính tôi chọn đề tài này.

**1.2. Mục tiêu của đề tài**

Mục tiêu tôi đặt ra khi thực hiện đồ án này bao gồm:

* Tìm hiểu lý thuyết về thuật toán Minimax và cách nó hoạt động trong trò chơi hai người.
* Mô hình hóa được bài toán chơi XO trên bảng 6x6 và 8x8.
* Cài đặt chương trình chơi XO giữa người chơi và máy tính sử dụng thuật toán Minimax.
* Đảm bảo chương trình có thể nhận input từ người chơi, xử lý bằng thuật toán AI, và trả ra output trực quan sau mỗi lượt.
* Đánh giá hiệu quả thuật toán, nhận xét ưu điểm và hạn chế để từ đó có thể đề xuất cải tiến về sau.

CHƯƠNG 2. ÁP DỤNG GIẢI THUẬT MINIMAX – MỞ RỘNG XO GAME 6x6 & XO GAME 8x8

**2.1 Phát biểu bài toán:**

Viết chương trình mô phỏng trò chơi XO (Tic-Tac-Toe) mở rộng, cụ thể là trên bảng kích thước 6x6 và 8x8. Trò chơi diễn ra giữa hai người chơi: người dùng và máy tính (AI). Mỗi người chơi lần lượt đánh dấu (X hoặc O) lên bảng. Người chiến thắng là người đầu tiên có 5 ký tự liên tiếp theo hàng ngang, hàng dọc hoặc đường chéo.

Trong đó, máy tính sẽ sử dụng thuật toán Minimax để lựa chọn nước đi sao cho có lợi nhất cho mình, đồng thời giả định người chơi sẽ luôn chọn nước đi tốt nhất cho đối thủ. Đây là bài toán điển hình trong lớp bài toán zero-sum game.

**2.2 Hướng giải quyết bài toán - Mô hình tính toán**

a. Đại diện trạng thái

Tôi biểu diễn bảng cờ bằng một mảng hai chiều kích thước BOARD\_SIZE x BOARD\_SIZE, với mỗi ô có giá trị:

• ' ' (dấu cách): ô trống chưa đánh

• 'X': người chơi

• 'O': AI

b. Luật chơi

• Hai bên lần lượt đánh vào các ô trống.

• Người nào tạo được một chuỗi 5 ký tự liên tiếp (ngang, dọc hoặc chéo) trước sẽ thắng.

• Nếu bảng đầy mà không ai thắng, thì kết quả là hòa.

c. Thuật toán Minimax

• Ý tưởng: Tại mỗi trạng thái, tôi xây dựng một cây các nước đi có thể xảy ra.

• Nếu là lượt của AI (O): AI sẽ tìm cách tối đa hóa điểm số.

• Nếu là lượt của người (X): AI sẽ giả định người chơi cố gắng tối thiểu hóa điểm của AI.

• Điểm số đánh giá (tại node lá):

• +10: AI thắng

• -10: người chơi thắng

• 0: hòa hoặc chưa rõ

• Hạn chế độ sâu (Max Depth):

Do số trạng thái tăng rất nhanh với bảng 6x6 hoặc 8x8, tôi giới hạn độ sâu của cây tìm kiếm để tránh việc chương trình phản hồi quá chậm. Độ sâu mặc định là 3.

d. Mô hình chương trình

• Nhập vào: tọa độ hàng/cột mà người chơi chọn

• Hệ thống cập nhật trạng thái bảng

• Gọi thuật toán Minimax để chọn nước đi cho AI

• In ra bảng sau mỗi lượt và thông báo người thắng khi có

e. Lược đồ hoạt động (tóm tắt logic)

[Người chơi nhập nước đi]

↓

[Cập nhật bàn cờ]

↓

[Kiểm tra thắng/thua]

↓

[AI gọi Minimax chọn nước đi tốt nhất]

↓

[Cập nhật bàn cờ]

↓

[Kiểm tra thắng/thua]

2.3 Mã nguồn

Do dung lượng lớn nên tôi chia mã nguồn thành các phần nhỏ sau:

2.3.1 Khởi tạo bàn cờ và in bàn cờ

def create\_board():

return [[' ' for \_ in range(BOARD\_SIZE)] for \_ in range(BOARD\_SIZE)]

def print\_board(board):

print(" " + " ".join(str(i) for i in range(BOARD\_SIZE)))

for idx, row in enumerate(board):

print(f"{idx} " + " ".join(row))

2.3.2 Kiểm tra thắng

def check\_win(board, player):

for r in range(BOARD\_SIZE):

for c in range(BOARD\_SIZE):

if (c + WIN\_CONDITION <= BOARD\_SIZE and all(board[r][c + i] == player for i in range(WIN\_CONDITION))) or \

(r + WIN\_CONDITION <= BOARD\_SIZE and all(board[r + i][c] == player for i in range(WIN\_CONDITION))) or \

(r + WIN\_CONDITION <= BOARD\_SIZE and c + WIN\_CONDITION <= BOARD\_SIZE and all(board[r + i][c + i] == player for i in range(WIN\_CONDITION))) or \

(r - WIN\_CONDITION + 1 >= 0 and c + WIN\_CONDITION <= BOARD\_SIZE and all(board[r - i][c + i] == player for i in range(WIN\_CONDITION))):

return True

return False

2.3.3 Đánh giá điểm và thuật toán Minimax

def evaluate(board):

if check\_win(board, AI): return 10

elif check\_win(board, PLAYER): return -10

else: return 0

def minimax(board, depth, is\_maximizing):

if check\_win(board, PLAYER): return -10, None

if check\_win(board, AI): return 10, None

if not get\_available\_moves(board) or depth == 0: return 0, None

best\_move = None

if is\_maximizing:

max\_eval = -math.inf

for move in get\_available\_moves(board):

r, c = move

board[r][c] = AI

eval, \_ = minimax(board, depth - 1, False)

board[r][c] = ' '

if eval > max\_eval:

max\_eval = eval

best\_move = move

return max\_eval, best\_move

else:

min\_eval = math.inf

for move in get\_available\_moves(board):

r, c = move

board[r][c] = PLAYER

eval, \_ = minimax(board, depth - 1, True)

board[r][c] = ' '

if eval < min\_eval:

min\_eval = eval

best\_move = move

return min\_eval, best\_move

\*Kết luận:

Ưu điểm:

• AI có khả năng phân tích trạng thái và chọn nước đi hợp lý.

• Dễ mô hình hóa, thuật toán rõ ràng, chạy ổn định.

• Có thể mở rộng lên 8x8, thêm Alpha-Beta pruning hoặc nâng cấp GUI.

Nhược điểm:

• Khi mở rộng kích thước bảng, số trạng thái tăng rất nhanh (bùng nổ tổ hợp).

• Tốc độ xử lý giảm nếu không giới hạn độ sâu.

• AI có thể bị hạn chế tầm nhìn nếu depth thấp.

CHƯƠNG 3. KẾT LUẬN

**3.1. Kết quả đạt được**

Qua quá trình thực hiện đồ án từ việc tìm hiểu lý thuyết đến triển khai thực tế, tôi đã đạt được một số kết quả cụ thể như sau:

• Áp dụng thành công thuật toán Minimax vào trò chơi chiến lược hai người. Thuật toán không chỉ giúp AI phản ứng hợp lý mà còn thể hiện được quá trình đánh giá trạng thái, chọn nước đi tốt nhất dựa trên mô hình tính toán.

• Tôi đã xây dựng hoàn chỉnh một chương trình cho phép người chơi đấu với máy (AI) trên hai phiên bản bàn cờ XO: 6x6 và 8x8. Mỗi bước đi của AI được chọn lựa có chiến lược rõ ràng thay vì ngẫu nhiên.

• Để đảm bảo hiệu suất, tôi giới hạn độ sâu tìm kiếm của cây Minimax ở mức hợp lý (ví dụ: depth = 2 hoặc 3). Giải pháp này giúp chương trình không bị chậm, vẫn đảm bảo ra quyết định nhanh chóng mà không tốn quá nhiều tài nguyên xử lý.

• Tôi thiết kế chương trình theo hướng mô-đun hóa, tách biệt từng chức năng: kiểm tra thắng, đánh giá bàn cờ, in bảng, chọn nước đi,… giúp dễ dàng bảo trì hoặc nâng cấp sau này.

• Chương trình chạy ổn định trên môi trường console (terminal), dễ triển khai, dễ kiểm thử và đặc biệt phù hợp với mục tiêu đào sâu kiến thức thuật toán hơn là giao diện.

• Tôi cũng đã thử nghiệm thực tế nhiều lượt chơi, ghi nhận các phản hồi từ AI, từ đó điều chỉnh thuật toán và các điều kiện kết thúc để chương trình hoạt động trơn tru, đáp ứng được kỳ vọng của một sản phẩm học thuật hoàn chỉnh trong phạm vi môn học.

Tóm lại, tôi không chỉ hoàn thành yêu cầu của đề tài, mà còn mở rộng được kiến thức cá nhân về tư duy AI, lập trình hướng thuật toán và cách giải quyết bài toán có tính tổng quát.

**3.2. Đánh giá chung**

Về mặt kỹ thuật:

Thuật toán Minimax được triển khai hiệu quả trên không gian trạng thái mở rộng (bàn cờ 6x6 và 8x8). Việc giới hạn độ sâu phù hợp giúp tránh được tình trạng “bùng nổ tổ hợp” và vẫn giữ được khả năng phản hồi nhanh của AI. Các hàm đánh giá, kiểm tra chiến thắng và chọn nước đi đều hoạt động ổn định.

Về mặt học thuật:

Qua đề tài này, tôi hiểu rõ hơn về cây trạng thái, cách AI mô phỏng các lựa chọn của đối phương, cách đánh giá mỗi trạng thái và nguyên lý đệ quy trong Minimax. Tôi cũng nhận ra rằng việc thiết kế hàm đánh giá (evaluation function) có tác động lớn đến hành vi của AI.

Về mặt lập trình:

Tôi rèn luyện được kỹ năng chia nhỏ bài toán, thiết kế chương trình có cấu trúc rõ ràng, dễ đọc, dễ mở rộng. Ngoài ra, tôi cũng hiểu tầm quan trọng của việc giới hạn độ phức tạp thuật toán để đảm bảo hiệu năng và tính khả thi của hệ thống.

**3.3. Hướng phát triển tiếp theo**

Dù chương trình hiện tại đã chạy ổn định và đạt được mục tiêu đề tài, tôi nhận thấy vẫn còn nhiều hướng để phát triển và tối ưu hóa hệ thống:

1. Tích hợp thuật toán cắt tỉa Alpha-Beta: Đây là phần mở rộng phổ biến của Minimax, giúp giảm đáng kể số lượng trạng thái phải duyệt bằng cách loại bỏ những nhánh không cần thiết, từ đó tăng tốc độ xử lý nhưng vẫn đảm bảo tính chính xác trong lựa chọn.

2. Phát triển giao diện đồ họa:

• Tkinter: cho phép tạo giao diện trực quan trên desktop, người chơi có thể click chuột để chọn ô.

• Flask (Python Web Framework): kết hợp với ReactJS để triển khai trò chơi trên nền tảng web, phù hợp cho nhiều người dùng truy cập và tương tác.

3. Ứng dụng học tăng cường (Reinforcement Learning):

• AI có thể học dần từ các lượt chơi trước.

• Điều chỉnh cách đánh giá nước đi tùy theo kết quả thắng/thua trong các ván cũ.

• Hướng này giúp AI tiến hóa và trở nên thông minh hơn thay vì chỉ “tính toán cứng”.

4. Phát triển thêm chế độ chơi mới:

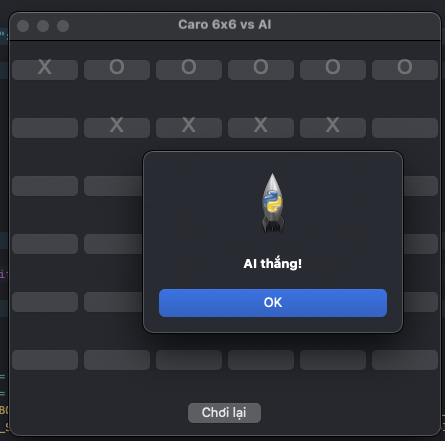
• Chế độ AI vs AI: mô phỏng hai máy đấu nhau để quan sát chiến lược.

• Chế độ hai người chơi thực: cho phép người dùng sử dụng chương trình như một nền tảng chơi XO thông minh.

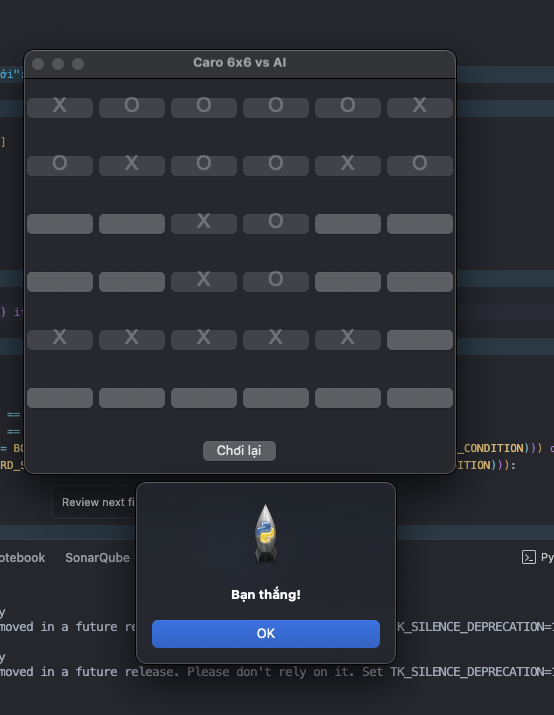
• Tích hợp lưu trữ lịch sử ván chơi, undo/redo, chế độ huấn luyện AI cũng là hướng phát triển có tiềm năng.

3.4. Hình ảnh mình hoạ

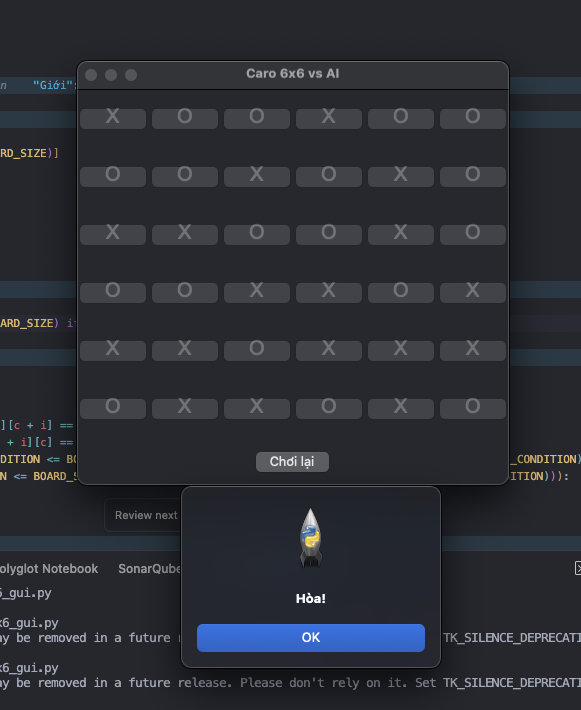
Ai thắng:



Người thắng:



Hoà:



TÀI LIỆU THAM KHẢO

[1] Hoang S, Nguyen K, Huynh T, et al., “Chest X-ray Severity Score as a Putative  
Predictor of Clinical Outcome in Hospitalized Patients: An Experience From a Vietnamese Covid-19 Field Hospital”, 10.7759/cureus.23323, 2022.

[2] L. J. Muhammad, Ebrahem A. Algehyne, et al., “Supervised Machine Learning  
Models for Prediction of COVID-19 Infection using Epidemiology Dataset”, Advances in Computational Approaches for Artificial Intelligence, Image Processing, IoT and Cloud Applications, 2020.

[3] Zoabi, Y., Deri-Rozov, S. & Shomron, N, “Machine learning-based prediction of COVID-19 diagnosis based on symptoms”, 2021.