**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**BÁO CÁO PROJECT**

**Xây dựng hệ thống cờ Caro**   
**Ứng dụng các thuật toán Minimax**

**Và Alpha-Beta Pruning**

**NGUYỄN HUY HOÀNG – 20225845**

**NGUYỄN DUY KHÁNH – 20225865**

**Môn học: Nhập môn Trí tuệ nhân tạo**

**Ngành Công nghệ thông tin Việt Nhật**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | TS. Đặng Tuấn Linh |
| **Trường:** | Trường Công nghệ thông tin và Truyền thông |

**HÀ NỘI, 6/2025**

***Bảng 1.1 Phân công công việc***

|  |  |  |
| --- | --- | --- |
| **Thành viên** | **Nhiệm vụ chính** | **Tỉ lệ** |
| **Nguyễn Huy Hoàng** | **Phát triển giao diện và weight table** | **45%** |
| **Nguyễn Duy Khánh** | **Phát triển các thuật toán Minimax, cắt tỉa Alpha Beta và học tăng cường** | **55%** |

**Tóm tắt nội dung dự án**

Cờ caro là một trò chơi rất phổ biến trong giới trẻ, được nhiều người yêu thích nhờ tính tiện lợi có thể chơi ở bất cứ đâu và đòi hỏi sự tư duy thông minh. Chính vì vậy, nhóm em đã quyết định chọn cờ caro làm đề tài bài tập lớn để nghiên cứu những phương pháp tối ưu nhất nhằm giành chiến thắng trong trò chơi này.

Mục lục

[**Chương 1:** **GIỚI THIỆU BÀI TOÁN VÀ Ý TƯỞNG** 6](#_Toc201175471)

[1.1 Giới thiệu chung 6](#_Toc201175472)

[1.2 Yêu cầu bài toán 6](#_Toc201175473)

[1.3 Công cụ sử dụng 6](#_Toc201175474)

[**Chương 2: GIẢI QUYẾT BÀI TOÁN** 7](#_Toc201175475)

[2.1 Phân tích yêu cầu 7](#_Toc201175476)

[2.1.1 Phát biểu bài toán 7](#_Toc201175477)

[2.1.2 Xây dựng thuật toán 8](#_Toc201175478)

[2.2 Cải thiện thuật toán 9](#_Toc201175479)

[2.2.1 Cơ chế hoạt động: 9](#_Toc201175480)

[2.2.2 Mục tiêu: 10](#_Toc201175481)

[2.2.3 So sánh giữa AI có và không học tăng cường: 10](#_Toc201175482)

[**Chương 3: HƯỚNG DẪN CÀI ĐẶT VÀ THỬ NGHIỆM TRÒ CHƠI** 10](#_Toc201175483)

[3.1 Yêu cầu hệ thống 10](#_Toc201175484)

[3.2 Các thư viện cần thiết 11](#_Toc201175485)

[3.3 Cách cài đặt 11](#_Toc201175486)

[3.4 Cách chạy và thử nghiệm trò chơi 11](#_Toc201175487)

[3.5 Chơi trò chơi: 12](#_Toc201175488)

[3.6 Quan sát và điều chỉnh (Nâng cao) 12](#_Toc201175489)

[**Chương 4: CÁC KHÓ KHĂN GẶP PHẢI KHI THỰC HIỆN BÀI TOÁN** 13](#_Toc201175490)

[4.1 Thiết lập hàm đánh giá (Evaluation Function) hiệu quả: 13](#_Toc201175491)

[4.2 Tối ưu hóa thuật toán Minimax và Alpha-Beta Pruning: 13](#_Toc201175492)

[4.3 Thách thức trong học tăng cường (Reinforcement Learning) đơn giản để tinh chỉnh trọng số: 14](#_Toc201175493)

[4.4 Tối ưu hóa hiệu suất: 14](#_Toc201175494)

[4.5 Giao diện người dùng (Tkinter): 14](#_Toc201175495)

[**Chương 5: KẾT LUẬN VÀ ĐỀ XUẤT HƯỚNG PHÁT TRIỂN** 15](#_Toc201175496)

[5.1 Kết luận 15](#_Toc201175497)

[5.2 Định hướng phát triển 16](#_Toc201175498)

**Chương 1:** **GIỚI THIỆU BÀI TOÁN VÀ Ý TƯỞNG**

* 1. Giới thiệu chung

Đề tài tập trung vào việc xây dựng một hệ thống chơi cờ Caro thông minh, trong đó trí tuệ nhân tạo (AI) được phát triển dựa trên thuật toán Minimax kết hợp cắt tỉa Alpha-Beta nhằm đưa ra các nước đi tối ưu. Thông qua việc mô phỏng trạng thái bàn cờ và đánh giá chiến lược tấn công hoặc phòng thủ, hệ thống giúp người chơi trải nghiệm một đối thủ máy có khả năng tư duy chiến thuật. Đề tài không chỉ mang tính giải trí mà còn góp phần ứng dụng các thuật toán AI vào thực tế, đồng thời tạo điều kiện cho sinh viên thực hành các kiến thức về tìm kiếm và ra quyết định trong môi trường đối kháng.

* 1. Yêu cầu bài toán
* Yêu cầu 1: Xây dựng 1 bàn cờ với kích thước 15x15
* Yêu cầu 2: Người chơi là X và máy là O. Người thắng là người đi được 5 quân cờ cùng 1 hàng, 1 cột hoặc 1 đường chéo hoặc hoà khi đã hết chỗ trên bảng
* Yêu cầu 3: Tối ưu làm sao cho máy chơi tốt nhất có thể
  1. Công cụ sử dụng

**Ngôn ngữ lập trình**:

* **Python**: Ngôn ngữ chính dùng để xây dựng thuật toán AI nhờ cú pháp đơn giản, thư viện đa dạng.

**Thư viện và framework AI (nếu dùng Python)**:

* NumPy: Hỗ trợ xử lý ma trận bàn cờ.
* pygame: Dùng để xây dựng giao diện người dùng cho game cờ Caro.
* matplotlib (tùy chọn): Dùng để hiển thị trực quan quá trình đánh giá trạng thái.
* scikit-learn hoặc TensorFlow/Keras: Nếu mở rộng AI bằng học máy hoặc học tăng cường.

**Môi trường phát triển (IDE):**

* **Visual Studio Code / PyCharm**: Công cụ lập trình hỗ trợ code hiệu quả.

**Hệ điều hành**:

* Windows, macOS hoặc Linux đều tương thích (nên dùng Python đa nền tảng).

**Chương 2: GIẢI QUYẾT BÀI TOÁN**

2.1 Phân tích yêu cầu

2.1.1 Phát biểu bài toán

a. Mục tiêu bài toán: Người chơi là X và máy là O. Người thắng là người đi được 5 quân cờ cùng 1 hàng, 1 cột hoặc 1 đường chéo hoặc hoà khi đã hết chỗ trên bảng

Ảnh có chứa ảnh chụp màn hình, văn bản, hàng

Nội dung do AI tạo ra có thể không chính xác.

b. Phát biểu bài toán:

Bàn cờ (Board) là 1 ma trận bao gồm các ô cờ (Pos) được tổ chức trong một mảng hai chiều có kích thước a x b.

* Mỗi ô Pos có thể xác định được các thông tin sau:
* Vị trí ô: bao gồm hàng (row) và cột (column).
* Trạng thái ô (Status): có thể là trống (0), nước đi của đối thủ (2), hoặc nước đi của máy (1).
* Độ nguy hiểm của ô: phụ thuộc vào trạng thái của ô và có thể thay đổi theo tình huống.
* Trong quá trình đánh giá các ô cờ, người chơi thường phân loại nước đi theo cảm nhận như: nguy hiểm, bình thường hoặc ít nguy hiểm. Trong khi đó, máy tính đánh giá cụ thể hơn bằng cách gán giá trị số cho từng ô, phản ánh mức độ quan trọng hoặc nguy hiểm của nước đi đó.

c. Tìm kiếm giải pháp:

Sau khi người chơi đã chọn được ô đầu tiên, thay vì chọn ngẫu nhiên 1 ô trống trên bàn cờ, máy sẽ chọn 1 ô xung quanh ô đã được đánh để thực hiện tính toán

Nếu ô đó rơi vào trường hợp đặc biệt:

           + Người chơi đi là thắng => chặn ngay

           + Máy đi là thắng => đi ngay

Còn không, máy sẽ giả lập đặt quân vào các ô, tính toán xem nước đi nào tốt nhất và đi quân đó

2.1.2 Xây dựng thuật toán

a. Bảng thế trận và bảng trọng lượng

* Bảng thế trận là trạng thái hiện tại của bàn cờ, giúp cho máy xác định được đâu là ô có thể đi được
* Bảng trọng lượng là bảng gồm các thế cờ kèm với trọng số tương ứng, phản ánh mức độ nguy hiểm nếu thuộc về đối thủ, phản ánh mức độ ưu tiên nếu thuộc về mình
* Máy sẽ chọn nước đi sao cho tổng điểm đánh giá của bàn cờ là cao nhất, tương đương với khả năng thắng cao nhất hoặc nguy cơ thua thấp nhất.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

b. Giải thuật tìm kiếm Minimax

* Sau khi đặt thử nghiệm quân cờ vào ô trống, máy sẽ dùng giải thuật tìm kiếm minimax để tính toán nước đi tốt nhất
* Dựa vào thế trận bảng hiện tại và bảng trọng lượng, máy sẽ gọi đệ quy luân phiên giữa máy và người chơi X lần cho đến khi tìm được nước đi tốt nhất thông qua đánh giá min-max
* Máy sẽ cố gắng chọn nước đi có lợi nhất (nhiều điểm nhất) trong khi người chơi sẽ cố gắng chọn nước đi bất lợi nhất (ít điểm nhất) cho máy

c. Giải thuật cắt tỉa Alpha-Beta

* Gọi alpha là điểm tốt nhất có thể đạt được và beta là điểm thấp nhất có thể đạt được.
* Nếu đến lượt của máy và xuất hiện nước đi có nhiều điểm hơn alpha => cập nhật điểm mới cho alpha và không xét các nước đi còn lại của nhánh đấy (tỉa bớt)
* Ngược lại nếu đến lượt của người chơi mà xuất hiện nước đi có ít điểm hơn beta => cập nhật điểm mới cho beta và không xét các nước đi còn lại của nhánh đó nữa (tỉa bớt)
* Việc này giúp tối ưu hoá nước đi, làm cho máy tính toán nhanh hơn

2.2 Cải thiện thuật toán

Thuật toán Học tăng cường (Reinforcement Learning - RL)

### 2.2.1 Cơ chế hoạt động:

1. Chơi ván đấu: AI chơi một ván Gomoku với người chơi hoặc một AI khác.
2. Xác định kết quả: Sau khi ván đấu kết thúc, hệ thống xác định xem AI đã thắng (+1), thua (-1), hay hòa (0).
3. Trích xuất đặc trưng cuối cùng: Tại trạng thái bàn cờ cuối cùng của ván đấu, AI phân tích lại các “đặc trưng” (patterns) như chuỗi 2, 3, 4, 5 quân cờ, vị trí gần trung tâm, v.v., cho cả quân của AI và quân của đối thủ.
4. Điều chỉnh trọng số:
   * Nếu AI thắng: Các trọng số liên quan đến các đặc trưng tích cực của AI (ví dụ: own\_open\_four) sẽ được tăng nhẹ, và các trọng số liên quan đến đặc trưng tiêu cực của đối thủ (ví dụ: opp\_open\_four) sẽ được giảm nhẹ hơn (làm chúng trở nên “tiêu cực” hơn trong mắt AI).
   * Nếu AI thua: Các trọng số liên quan đến đặc trưng tích cực của AI sẽ được giảm nhẹ, và các trọng số liên quan đến đặc trưng tiêu cực của đối thủ sẽ được giảm mạnh hơn nhiều (làm chúng trở nên *rất* “tiêu cực” và AI sẽ học cách tránh/chặn chúng mạnh mẽ hơn).
   * Nếu hòa: Không có sự điều chỉnh trọng số đáng kể nào trong cơ chế đơn giản này.
5. Lưu trọng số: Các trọng số mới được lưu vào một tệp (ví dụ: gomoku\_weights.json) để sử dụng cho các ván đấu tiếp theo.

### 2.2.2 Mục tiêu:

* Giúp AI tự động tinh chỉnh hàm đánh giá của mình. Theo thời gian, qua hàng trăm ván đấu, AI sẽ dần "học" được những mô hình nào thực sự có giá trị để đạt được chiến thắng và những mô hình nào của đối thủ là cực kỳ nguy hiểm cần phải chặn ngay lập tức, dẫn đến việc chơi tốt hơn mà không cần lập trình viên phải can thiệp thủ công liên tục.

### 2.2.3 So sánh giữa AI có và không học tăng cường:

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **AI Có Học tăng cường** | **AI Không có Học tăng cường** |
| **Hiệu quả (Khả năng chơi)** | AI tự học và tiến bộ theo thời gian, có thể đạt chiến lược mạnh mẽ hơn. | Bị giới hạn bởi chất lượng trọng số ban đầu, không tự cải thiện được. |
| **Tài nguyên tiêu tốn** | Có chi phí đọc/ghi file trọng số và tính toán cập nhật sau mỗi ván đấu. | Chỉ cần đọc trọng số một lần khi khởi động. |
| **Thời gian chạy** | Do có bước cập nhật trọng số sau mỗi ván. | Không có bước cập nhật trọng số. |

**Chương 3: HƯỚNG DẪN CÀI ĐẶT VÀ THỬ NGHIỆM TRÒ CHƠI**

## 3.1 Yêu cầu hệ thống

* Hệ điều hành: Bất kỳ hệ điều hành nào có thể chạy Python (Windows, macOS, Linux).
* Python: Phiên bản Python 3.x đã được cài đặt trên máy tính của bạn. Bạn có thể tải Python từ trang web chính thức: [python.org](https://www.python.org/).

## 3.2 Các thư viện cần thiết

Trò chơi này sử dụng thư viện đồ họa Tkinter, đây là một thư viện chuẩn của Python và thường được cài đặt sẵn khi bạn cài đặt Python. Tuy nhiên, nếu vì lý do nào đó mà bạn gặp lỗi liên quan đến tkinter (ví dụ: ModuleNotFoundError: No module named 'tkinter'), bạn có thể cần cài đặt nó thủ công:

* Trên Windows: Tkinter thường được bao gồm trong quá trình cài đặt Python. Nếu thiếu, bạn có thể chạy lại trình cài đặt Python và đảm bảo tùy chọn "tcl/tk" được chọn.
* Trên Debian/Ubuntu (Linux): sudo apt-get install python3-tk
* Trên Fedora (Linux): sudo dnf install python3-tkinter
* Trên macOS: Tkinter thường đi kèm với Python cài đặt qua Homebrew. Nếu không, bạn có thể cần cài đặt Xcode Command Line Tools hoặc một số bản phân phối Python khác.

## 3.3 Cách cài đặt

1. Lưu file: Lưu nội dung mã Python bạn đã cung cấp vào một file có tên là gomoku.py (hoặc bất kỳ tên nào khác kết thúc bằng .py). Đảm bảo file được lưu ở một vị trí dễ tìm trên máy tính của bạn (ví dụ: trên Desktop hoặc trong thư mục Documents).
2. File trọng số (tùy chọn nhưng khuyến nghị): Khi bạn chạy trò chơi lần đầu, nó sẽ tự động tạo một file tên là gomoku\_weights.json trong cùng thư mục với file gomoku.py. File này lưu trữ các trọng số (weights) của AI để AI có thể học hỏi và cải thiện qua các ván đấu.

## 3.4 Cách chạy và thử nghiệm trò chơi

1. Mở cửa sổ dòng lệnh/terminal:
   * Trên Windows: Nhấn Win + R, gõ cmd và nhấn Enter. Hoặc tìm "Command Prompt" hoặc "PowerShell" trong Start Menu.
   * Trên macOS/Linux: Mở ứng dụng "Terminal".
2. Điều hướng đến thư mục chứa file: Sử dụng lệnh cd (change directory) để đi đến thư mục mà bạn đã lưu file gomoku.py.
   * Ví dụ, nếu bạn lưu file trong thư mục Downloads, bạn sẽ gõ: cd Downloads và nhấn Enter.
3. Chạy trò chơi: Sau khi đã ở đúng thư mục, gõ lệnh sau và nhấn Enter:

Bash

python gomoku.py

## 3.5 Chơi trò chơi:

* + Một cửa sổ trò chơi Gomoku sẽ xuất hiện.
  + Bạn là người chơi X (quân đen), AI là người chơi O (quân trắng).
  + Nhấp chuột vào một ô trống trên bảng để đặt quân của bạn.
  + AI sẽ tự động thực hiện nước đi của nó sau một khoảng thời gian ngắn.
  + Mục tiêu: Là người đầu tiên xếp được 5 quân của mình liên tiếp theo chiều ngang, dọc hoặc chéo.
  + Bạn có thể nhấn nút "Reset Game" để bắt đầu một ván đấu mới bất cứ lúc nào.

## 3.6 Quan sát và điều chỉnh (Nâng cao)

* Theo dõi AI học: Sau mỗi ván đấu kết thúc (bạn thắng, AI thắng, hoặc hòa), chương trình sẽ cập nhật các trọng số của AI và lưu chúng vào file gomoku\_weights.json. Bạn có thể mở file này bằng một trình soạn thảo văn bản để xem các giá trị trọng số thay đổi như thế nào. AI sẽ dần học cách chơi tốt hơn qua các ván đấu.
* Thay đổi độ khó của AI: Bạn có thể chỉnh sửa file gomoku.py để thay đổi độ khó của AI bằng cách điều chỉnh biến MAX\_DEPTH:
  + Mở file gomoku.py bằng một trình soạn thảo văn bản (ví dụ: Notepad trên Windows, TextEdit trên macOS, Sublime Text, VS Code...).
  + Tìm dòng: MAX\_DEPTH = 2
  + Giá trị 2 là độ sâu tìm kiếm mặc định của thuật toán Minimax.
    - Tăng giá trị này (ví dụ: lên 3 hoặc 4): AI sẽ mạnh hơn nhưng thời gian tính toán nước đi cũng sẽ lâu hơn đáng kể.
    - Giảm giá trị này (ví dụ: xuống 1): AI sẽ yếu hơn và đưa ra nước đi nhanh hơn.
  + Lưu ý: Giá trị MAX\_DEPTH càng lớn, AI càng "thông minh" nhưng cũng càng "ngốn" tài nguyên máy tính và thời gian chờ đợi sẽ lâu hơn.
* Thay đổi kích thước bàn cờ hoặc số quân để thắng: Bạn cũng có thể chỉnh sửa các biến BOARD\_SIZE và WIN\_LENGTH ở đầu file gomoku.py nếu muốn thay đổi quy tắc chơi.

**Chương 4: CÁC KHÓ KHĂN GẶP PHẢI KHI THỰC HIỆN BÀI TOÁN**

## 4.1 Thiết lập hàm đánh giá (Evaluation Function) hiệu quả:

* Định nghĩa các mẫu (patterns): Việc xác định và đếm chính xác các loại chuỗi quân (ví dụ: "năm quân", "bốn quân mở", "ba quân bán mở"...) là rất phức tạp. Mã nguồn hiện tại đã cố gắng định nghĩa các trường hợp này, nhưng việc đảm bảo rằng tất cả các biến thể được nhận diện đúng (và không bị đếm trùng hoặc bỏ sót) là một thách thức lớn. Đặc biệt, việc phân biệt giữa "mở" (open) và "bán mở" (half-open) đòi hỏi kiểm tra kỹ lưỡng các ô trống và ô bị chặn ở hai đầu chuỗi.
* Gán trọng số ban đầu: Các trọng số ban đầu (INITIAL\_WEIGHTS) được gán cho mỗi loại mẫu (ví dụ: own\_five, opp\_open\_four). Việc tìm ra các giá trị này sao cho chúng phản ánh đúng mức độ quan trọng của từng mẫu trong chiến thuật Gomoku là không hề dễ dàng và thường đòi hỏi thử nghiệm, điều chỉnh thủ công rất nhiều.
* Cân bằng giữa các mẫu của mình và đối thủ: Hàm đánh giá cần phải cân bằng giữa việc khuyến khích tạo ra các chuỗi quân của mình và ngăn chặn các chuỗi quân của đối thủ. Ví dụ, opp\_open\_four có trọng số rất âm để thể hiện mức độ nguy hiểm cao và cần phải chặn ngay lập tức.

## 4.2 Tối ưu hóa thuật toán Minimax và Alpha-Beta Pruning:

* Độ sâu tìm kiếm (MAX\_DEPTH): Gomoku là một trò chơi có không gian trạng thái rất lớn. Với kích thước bàn cờ 15x15, việc tìm kiếm sâu ngay cả với độ sâu nhỏ (hiện tại là 2) cũng có thể tốn kém về mặt tính toán. Tăng MAX\_DEPTH để AI mạnh hơn sẽ làm tăng đáng kể thời gian tính toán cho mỗi nước đi.
* Tạo nước đi hợp lệ và sắp xếp nước đi: Thuật toán Minimax cần duyệt qua các nước đi có thể. Việc lấy tất cả các ô trống (get\_empty\_cells) và sau đó sắp xếp chúng một cách thông minh (\_get\_ordered\_moves) để duyệt những nước đi "hứa hẹn" trước giúp cắt tỉa Alpha-Beta hiệu quả hơn. Tuy nhiên, việc xây dựng một heuristic sắp xếp nước đi tốt là một thách thức riêng. Mã nguồn hiện tại ưu tiên các ô gần quân đã có và kiểm tra các nước đi thắng/chặn ngay lập tức, nhưng có thể chưa tối ưu hoàn toàn.
* Hiện tượng Horizon Effect: Với độ sâu tìm kiếm hạn chế, AI có thể không "nhìn thấy" được các mối đe dọa hoặc cơ hội ở sâu hơn trong cây tìm kiếm, dẫn đến các nước đi không tối ưu.

## 4.3 Thách thức trong học tăng cường (Reinforcement Learning) đơn giản để tinh chỉnh trọng số:

* Cơ chế cập nhật trọng số: Phương pháp update\_weights trong mã nguồn là một dạng học tăng cường rất đơn giản, dựa trên kết quả cuối cùng của ván đấu (thắng/thua/hòa). Việc điều chỉnh trọng số chỉ dựa vào trạng thái cuối cùng của bàn cờ có thể không phản ánh đúng đóng góp của từng nước đi trong suốt ván đấu. Một phương pháp robust hơn (như TD-learning) sẽ phức tạp hơn để triển khai.
* Tốc độ hội tụ và hiệu quả học: Với một cơ chế cập nhật đơn giản, AI có thể cần rất nhiều ván đấu để các trọng số hội tụ về một tập hợp tối ưu. Hơn nữa, việc đảm bảo các trọng số không "trôi dạt" quá xa hoặc thay đổi dấu một cách bất ngờ đối với các tính năng quan trọng (như "five" liên tiếp) cũng là một vấn đề cần kiểm soát.

## 4.4 Tối ưu hóa hiệu suất:

* Việc duyệt qua tất cả các hàng, cột, đường chéo để đếm mẫu có thể tốn kém, đặc biệt trên bàn cờ lớn hơn. Cần có các phương pháp tối ưu hóa việc trích xuất đặc trưng để giảm thời gian tính toán cho hàm đánh giá.

## 4.5 Giao diện người dùng (Tkinter):

* Mặc dù Tkinter tương đối đơn giản, việc đảm bảo giao diện phản hồi nhanh chóng và mượt mà, đặc biệt khi AI đang tính toán (có thể gây ra hiện tượng giao diện bị "đơ" nếu không xử lý đúng cách), cũng có thể là một vấn đề. Việc sử dụng master.after là một cách để giải quyết vấn đề này, nhưng cần được áp dụng đúng chỗ.

**Chương 5: KẾT LUẬN VÀ ĐỀ XUẤT HƯỚNG PHÁT TRIỂN**

5.1 Kết luận

Qua môn học và quá trình tìm hiểu để thực hiện đề tài này, nhóm em đã có cái nhìn toàn diện hơn về việc ứng dụng trí tuệ nhân tạo vào giải quyết các bài toán thực tiễn. Cờ caro là một trò chơi phù hợp để áp dụng các thuật toán như Minimax và giải thuật cắt tỉa Alpha-Beta.

Tuy nhiên, trong quá trình triển khai chương trình, nhóm em không tránh khỏi một số sai sót và những điểm chưa thực sự tối ưu. Rất mong nhận được sự góp ý của thầy để nhóm có thể hoàn thiện hơn trong tương lai.

5.2 Định hướng phát triển

Phát triển đa nền tảng

* Triển khai game trên web, mobile (Android/iOS) hoặc thậm chí ứng dụng desktop có giao diện đồ họa hấp dẫn.
* Đồng bộ người dùng giữa các nền tảng.

Chơi online nhiều người (Multiplayer)

* Phát triển chế độ chơi trực tuyến 1v1 qua mạng, kết hợp phòng chờ, xếp hạng (ranking).
* Tạo hệ thống chat, thách đấu, hoặc xếp hạng người chơi (leaderboard).

Tăng khả năng tương tác người dùng

* Bổ sung gợi ý nước đi, đánh giá thế cờ, hoặc chế độ luyện tập giúp người chơi cải thiện kỹ năng.
* Cho phép người dùng tùy chỉnh cấp độ AI (dễ, trung bình, khó).

Ứng dụng AI vào các biến thể cờ khác

* Mở rộng sang các biến thể như cờ vây (Go), cờ tướng, hoặc cờ Othello, trong đó AI phức tạp hơn rất nhiều và đòi hỏi xử lý dữ liệu lớn hơn.

Nhóm em xin chân thành cảm ơn!