**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙡🕮🙣**

****

**MÔN HỌC: TRÍ TUỆ NHÂN TẠO**

**BÁO CÁO CUỐI KỲ**

**Đề tài:**

**THUẬT TOÁN MINIMAX, A\*, NAIVE BAYES CLASSIFIER VÀ CÁC ỨNG DỤNG CỦA THUẬT TOÁN**

**GVHD: Ths. Trần Tiến Đức**

**SVTH: MSSV**

**Trần Diễm Quỳnh 22133046**

**Mã lớp: ARIN330585\_23\_2\_05**

**Tp. Hồ Chí Minh, tháng 5 năm 2024**

**ĐIỂM SỐ**

|  |  |  |  |
| --- | --- | --- | --- |
| **TIÊU CHÍ** | **NỘI DUNG** | **TRÌNH BÀY** | **TỔNG** |
| **ĐIỂM** |  |  |  |

**NHẬN XÉT**

*Ký tên*

**Ths. Trần Tiến Đức**

**Mục lục**

[LỜI CẢM ƠN 1](#_Toc166576869)

[MỞ ĐẦU 2](#_Toc166576870)

[**1. Lý do chọn đề tài 2**](#_Toc166576871)

[**2. Mục đích nghiên cứu 3**](#_Toc166576872)

[**3. Đối tượng và phạm vi nghiên cứu 3**](#_Toc166576873)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 5](#_Toc166576874)

[**I. Sơ lược thuật toán A\* 5**](#_Toc166576875)

[**1.1. Tư tưởng 5**](#_Toc166576876)

[**1.2. Phương pháp 5**](#_Toc166576877)

[**1.3. Mô hình của bài toán 6**](#_Toc166576878)

[**1.4. Ưu điểm và nhược điểm 7**](#_Toc166576879)

[**II. Sơ lược về thuật toán Minimax 7**](#_Toc166576880)

[**2.1. Tư tưởng 7**](#_Toc166576881)

[**2.2. Phương pháp 7**](#_Toc166576882)

[**2.3. Mô hình của bài toán 8**](#_Toc166576883)

[**2.4. Ưu điểm và nhược điểm 9**](#_Toc166576884)

[**III. Sơ lược về thuật toán Naive Bayes Classifier 9**](#_Toc166576885)

[**3.1. Tư tưởng 9**](#_Toc166576886)

[**3.2. Phương pháp 9**](#_Toc166576887)

[**3.3. Mô hình của bài toán 10**](#_Toc166576888)

[**3.4. Ưu điểm và nhược điểm 10**](#_Toc166576889)

[CHƯƠNG 2: ỨNG DỤNG CỦA THUẬT TOÁN TRONG THỰC TẾ 11](#_Toc166576890)

[**I. Giới thiệu trò chơi Maze 11**](#_Toc166576891)

[**1.1. Chạy chương trình trò chơi Maze trên desktop và streamlit 11**](#_Toc166576892)

[**1.2. Áp dụng thuật toán A\* vào trò chơi Maze 13**](#_Toc166576893)

[**1.3. Cách hoạt động của trò chơi Maze 15**](#_Toc166576894)

[**II. Giới thiệu trò chơi 8 - puzzle 16**](#_Toc166576895)

[**2.1. Chạy chương trình trò chơi 8 - puzzle trên desktop và streamlit 16**](#_Toc166576896)

[**2.2. Áp dụng thuật toán Minimax vào trò chơi 8 – puzzle 19**](#_Toc166576897)

[**2.3. Cách hoạt động của trò chơi 8 - puzzle 19**](#_Toc166576898)

[**III. Giới thiệu trò chơi 2048 20**](#_Toc166576899)

[**3.1. Chạy chương trình trò chơi 2048 trên desktop và streamlit 20**](#_Toc166576900)

[**3.2. Áp dụng thuật toán Minimax vào trò chơi 2048 24**](#_Toc166576901)

[**3.3. Cách hoạt động của trò chơi 2048 26**](#_Toc166576902)

[**IV. Giới thiệu trò chơi Connect Four 27**](#_Toc166576903)

[**4.1. Chạy chương trình trò chơi Connect Four trên desktop và streamlit 27**](#_Toc166576904)

[**4.2. Áp dụng thuật toán Minimax vào trò chơi Connect Four 31**](#_Toc166576905)

[**4.3. Cách hoạt động của trò chơi Connect Four 32**](#_Toc166576906)

[**V. Giới thiệu trò chơi Tic-tac-toe 34**](#_Toc166576907)

[**5.1. Chạy chương trình trò chơi Tic-tac-toe trên desktop và streamlit 34**](#_Toc166576908)

[**5.2. Áp dụng thuật toán Minimax vào trò chơi Tic-toc-toe 38**](#_Toc166576909)

[**5.3. Cách hoạt động của trò chơi Tic-toc-toe 39**](#_Toc166576910)

[**VI. Giới thiệu mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim 40**](#_Toc166576911)

[**6.1. Chạy mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim trên desktop và streamlit 40**](#_Toc166576912)

[**6.2. Áp dụng thuật toán Naive Bayes Classifier vào mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim 42**](#_Toc166576913)

[**6.3. Cách hoạt động của mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim 43**](#_Toc166576914)

[KẾT LUẬN 44](#_Toc166576915)

[TÀI LIỆU THAM KHẢO 46](#_Toc166576916)

# LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn chân thành nhất đến ThS. Trần Tiến Đức, người đã truyền đạt cho em những kiến thức vô giá về bộ môn Trí tuệ Nhân tạo. Trong suốt quá trình học tập và nghiên cứu đề tài này, em đã được thầy hướng dẫn, giúp đỡ tận tình và tâm huyết. Thầy đã dành nhiều thời gian và nỗ lực để em có thể hiểu sâu hơn về các khái niệm, lý thuyết và ứng dụng của Trí tuệ Nhân tạo.

Những kiến thức và kinh nghiệm mà thầy truyền đạt cho em sẽ luôn là những tài sản quý giá trong sự nghiệp học tập và phát triển sau này. Nhờ vào sự giảng dạy của thầy, em đã có cơ hội tiếp cận và tìm hiểu về những xu hướng mới nhất, những công nghệ tiên tiến trong lĩnh vực Trí tuệ Nhân tạo.

Tuy nhiên, sự tiếp nhận kiến thức của mỗi người có những giới hạn và hạn chế riêng. Do đó, trong quá trình hoàn thành đề tài này, chắc chắn không tránh khỏi những thiếu sót và những khuyết điểm. Vì vậy, em mong nhận được những lời góp ý và đánh giá từ thầy để bài tiểu luận của em và các bạn cùng lớp được hoàn thiện hơn.

Cuối cùng, em xin chúc thầy sức khỏe, hạnh phúc và thành công trên con đường sự nghiệp giảng dạy. Thầy là một người thầy tuyệt vời và đầy tâm huyết, luôn là người động viên và truyền cảm hứng cho mọi người. Mong thầy sẽ ở lại giảng dạy em trong thời gian tới vì em mong muốn sẽ được gặp thầy ở các môn khác trong các môn học chuyên nghành của em hơn nữa. Em chân thành cảm ơn thầy vì sự đóng góp to lớn của thầy trong quá trình học tập và nghiên cứu của em.

# MỞ ĐẦU

1. **Lý do chọn đề tài**

Trong thời đại công nghệ 4.0, trí tuệ nhân tạo (AI) đã trở thành một lĩnh vực đầy triển vọng và thu hút sự quan tâm lớn từ cộng đồng khoa học và công nghệ. AI đại diện cho khả năng của máy tính trong việc học và tự động giải quyết các vấn đề, tương tự như con người. Trong lĩnh vực này, thuật toán đóng vai trò then chốt và là nền tảng cốt lõi để tạo ra các ứng dụng đột phá.

Do đó, trong báo cáo này, em đã chọn ba thuật toán cụ thể để nghiên cứu và trình bày: A\*, Minimax, Naive Bayes Classifier. Có ba lý do chính để em chọn đề tài này.

Thứ nhất, thuật toán A\* là một công cụ xuất sắc trong việc tìm kiếm đường đi ngắn nhất trong không gian tìm kiếm. Đặc biệt hữu ích trong robotica, trò chơi máy tính và lập kế hoạch đường đi, A\* kết hợp tìm kiếm theo chiều rộng và tìm kiếm theo chi phí để tìm được đường đi tối ưu nhanh chóng. Nghiên cứu thuật toán này sẽ giúp chúng ta hiểu rõ hơn về cách máy tính tìm kiếm và ra quyết định trong không gian tìm kiếm phức tạp.

Thứ hai, thuật toán Minimax là một công cụ quan trọng trong trò chơi và tìm kiếm. Nó cho phép máy tính ra quyết định thông minh trong các trò chơi đối kháng, nơi hai bên cố gắng tối đa hóa lợi ích của mình và tối thiểu hóa lợi ích của đối thủ. Minimax đánh giá các nước đi tiềm năng và tìm kiếm chuỗi các nước đi tối ưu để đạt được kết quả tốt nhất. Nghiên cứu thuật toán này sẽ giúp chúng ta hiểu rõ hơn về cách máy tính ra quyết định thông minh và ứng dụng của nó trong các lĩnh vực như cờ vua, cờ caro và trò chơi video.

Cuối cùng thuật toán Naive Bayes Classifier là một thuật toán phân loại quan trọng trong lĩnh vực học máy. Nó dựa trên nguyên tắc của định lý Bayes để phân loại các mẫu dữ liệu vào các nhóm khác nhau. Thuật toán này đặc biệt hữu ích trong việc xử lý và phân loại văn bản, email spam, phân loại ảnh, và nhiều ứng dụng khác. Nghiên cứu về thuật toán Naive Bayes Classifier sẽ giúp chúng ta hiểu cách máy tính xử lý thông tin và đưa ra các quyết định dựa trên dữ liệu đầu vào.

Tóm lại, việc nghiên cứu và áp dụng ba thuật toán Minimax, A\* và Naive Bayes Classifier trong báo cáo này sẽ giúp chúng ta hiểu sâu và rõ hơn về cách các thuật toán này hoạt động và ứng dụng của chúng trong thực tế.

1. **Mục đích nghiên cứu**

Hiểu Mục đích nghiên cứu của ba thuật toán đã được đề cập là Minimax, A\* và như sau:

A\*: Mục đích nghiên cứu A\* là tìm hiểu và áp dụng thuật toán tìm kiếm đường đi ngắn nhất trong không gian tìm kiếm. A\* kết hợp tìm kiếm theo chiều rộng và tìm kiếm theo chi phí để tìm ra đường đi tối ưu nhanh chóng và hiệu quả. Mục tiêu là hiểu cách A\* hoạt động và sử dụng nó trong các lĩnh vực như robotica, trò chơi máy tính và lập kế hoạch đường đi để giải quyết các vấn đề liên quan đến tìm kiếm và định tuyến.

Minimax: Mục đích nghiên cứu Minimax là hiểu và áp dụng thuật toán này trong các trò chơi đối kháng. Minimax cho phép máy tính ra quyết định thông minh bằng cách đánh giá và tìm kiếm chuỗi các nước đi tối ưu trong một trò chơi nơi hai bên cố gắng tối đa hóa lợi ích của mình và tối thiểu hóa lợi ích của đối thủ. Mục tiêu là hiểu cách Minimax hoạt động và áp dụng nó trong các lĩnh vực như cờ vua, cờ caro và trò chơi điện tử để phát triển các hệ thống trí tuệ nhân tạo thông minh trong trò chơi.

Naive Bayes Classifier: Mục đích của nghiên cứu là tìm hiểu cách Naive Bayes Classifier làm việc, cách xây dựng mô hình và cách đào tạo và sử dụng nó để phân loại dữ liệu. Hiểu về thuật toán này sẽ giúp chúng ta tận dụng tiềm năng của nó trong các ứng dụng thực tế và phát triển các hệ thống thông minh dựa trên học máy.

1. **Đối tượng và phạm vi nghiên cứu**
   1. **Thuật toán A\***

*Đối tượng nghiên cứu:* Đối tượng nghiên cứu của A\* là các bài toán tìm kiếm và định tuyến trong các lĩnh vực như robotica, trò chơi máy tính, lập kế hoạch đường đi, và các hệ thống tự động hóa.

*Phạm vi nghiên cứu:* Phạm vi nghiên cứu có thể bao gồm nghiên cứu và phát triển các biến thể và cải tiến của A\*, tối ưu hóa thuật toán để ang tốc độ tìm kiếm, và áp dụng A\* trong các tình huống định tuyến và lập kế hoạch đường đi thực tế.

* 1. **Thuật toán Minimax**

*Đối tượng nghiên cứu:* Đối tượng nghiên cứu của Minimax là các trò chơi đối kháng và các hệ thống trí tuệ nhân tạo liên quan đến trò chơi này.

*Phạm vi nghiên cứu:* Phạm vi nghiên cứu có thể bao gồm việc nghiên cứu và phát triển các biến thể và cải tiến của Minimax, tối ưu hóa thuật toán để ang hiệu suất tính toán, và áp dụng Minimax trong các trò chơi đối kháng mới hoặc các lĩnh vực ứng dụng khác.

* 1. **Thuật toán Naive Bayes Classifier**

*Đối tượng nghiên cứu:* Đối tượng nghiên cứu của thuật toán Naive Bayes Classifier là các phương pháp và kỹ thuật liên quan đến phân loại dữ liệu, phân loại các mẫu dữ liệu vào các nhóm khác nhau dựa trên các đặc trưng của chúng.

*Phạm vi nghiên cứu:* Phạm vi nghiên cứu của Naive Bayes Classifier bao gồm các lĩnh vực: xử lý ngôn ngữ tự nhiên, phân loại văn bản, phân loại ảnh, học máy và trí tuệ nhân tạo.

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1. **Sơ lược thuật toán A\***
   1. **Tư tưởng**

Thuật toán A\* kết hợp giữa tìm kiếm theo chiều rộng (BFS) và tìm kiếm theo chi phí thấp nhất (Dijkstra) để tìm đường đi tối ưu trong đồ thị hoặc lưới. Nó sử dụng một hàm đánh giá (heuristic) để ước lượng chi phí còn lại từ điểm hiện tại đến điểm đích, giúp tìm ra đường đi tối ưu nhanh chóng.

* 1. **Phương pháp**
* Đầu tiên, thuật toán A\* đánh giá mỗi điểm trong đồ thị hoặc lưới dựa trên hai thông số:

g(x): Chi phí từ điểm xuất phát đến điểm hiện tại.

h(x): Ước lượng chi phí từ điểm hiện tại đến điểm đích (heuristic).

* Thuật toán duyệt qua các điểm trong đồ thị hoặc lưới bằng cách thực hiện các bước sau:

+ Khởi tạo điểm xuất phát và đánh dấu nó.

+ Đặt giá trị hàm ước lượng f(x) = g(x) + h(x) của điểm xuất phát.

+ Sử dụng một hàng đợi ưu tiên (priority queue) để lưu trữ các điểm dựa trên giá trị f(x).

+ Lặp cho đến khi hàng đợi ưu tiên trống:

Lấy điểm hiện tại từ hàng đợi ưu tiên, điểm này có giá trị f(x) nhỏ nhất.

Nếu điểm hiện tại là điểm đích, kết thúc thuật toán và trả về đường đi từ điểm xuất phát đến điểm đích.

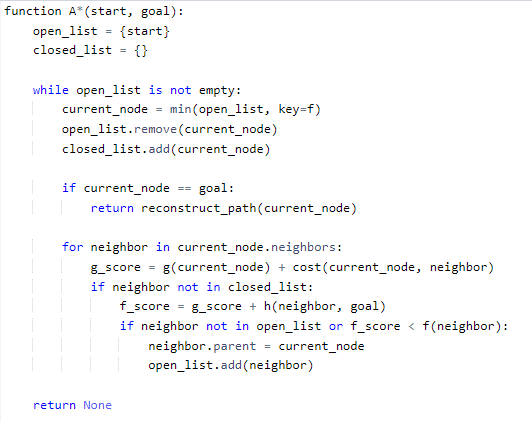
Duyệt qua các điểm kề với điểm hiện tại:

Nếu điểm kề chưa được đánh dấu:

Đánh dấu điểm kề và tính giá trị f(x) của nó.

Thêm điểm kề vào hàng đợi ưu tiên.

***Thuật toán A\* có thể được mô tả bằng giả mã sau:***



Trong giả mã này:

* start là nút khởi đầu.
* goal là nút đích.
* open\_list là danh sách mở.
* closed\_list là danh sách đóng.
* f(node) là giá trị f của nút node.
* g(node) là chi phí di chuyển từ nút khởi đầu đến nút node.
* h(node, goal) là ước tính chi phí di chuyển từ nút node đến nút đích.
* cost(node1, node2) là chi phí di chuyển từ nút node1 đến nút node2.
* reconstruct\_path(node) là hàm trả về đường đi từ nút khởi đầu đến nút node.
  1. **Mô hình của bài toán**

Thuật toán A\* có thể được áp dụng cho bất kỳ bài toán tìm đường đi trong đồ thị hoặc lưới. Đầu vào của thuật toán bao gồm:

Đồ thị hoặc lưới: Một cấu trúc dữ liệu mô tả mối quan hệ giữa các điểm trong không gian.

Điểm xuất phát: Điểm bắt đầu của đường đi.

Điểm đích: Điểm kết thúc của đường đi.

* 1. **Ưu điểm và nhược điểm**

*Ưu điểm:*

Tìm kiếm tối ưu: Thuật toán A\* đảm bảo tìm ra đường đi tối ưu từ điểm xuất phát đến điểm đích (nếu tồn tại).

Hiệu suất cao: Nhờ sử dụng hàm heuristic, thuật toán A\* thường tìm kiếm nhanh chóng và hiệu quả.

Tính linh hoạt: Thuật toán A\* có thể được áp dụng cho nhiều loại bài toán tìm đường đi trong đồ thị hoặc lưới.

*Nhược điểm:*

Hàm heuristic không chính xác: Nếu hàm heuristic không được thiết kế tốt, thuật toán A\* có thể không tìm ra đường đi tối ưu.

Tốn bộ nhớ: Thuật toán A\*cần lưu trữ thông tin về tất cả các điểm đã được duyệt qua, dẫn đến tốn nhiều bộ nhớ khi áp dụng cho đồ thị hoặc lưới lớn.

1. **Sơ lược về thuật toán Minimax**
   1. **Tư tưởng**

Thuật toán Minimax là một thuật toán được sử dụng trong lĩnh vực trò chơi để tìm cách ra quyết định tốt nhất trong một tình huống đối mặt với người chơi đối thủ. Tư tưởng cơ bản của thuật toán Minimax là đánh giá tất cả các nước đi có thể có và lựa chọn nước đi tối ưu dựa trên giả định rằng đối thủ cũng sẽ chọn nước đi tối ưu cho mình.

* 1. **Phương pháp**

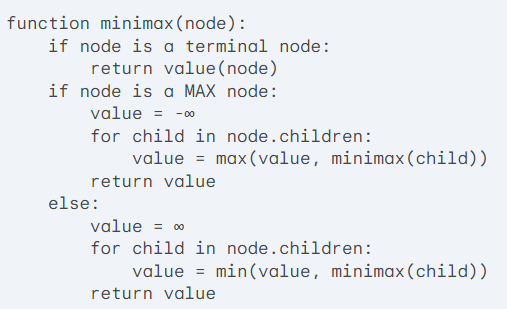
Thuật toán Minimax thường được áp dụng trong các trò chơi lược đồ như cờ vua, cờ caro, tic-tac-toe, và cả các trò chơi video phức tạp hơn như cờ thế, cờ vua 3D.

Thuật toán đi sâu vào cây trò chơi, trong đó mỗi nút đại diện cho một trạng thái của trò chơi và các cạnh đi từ nút cha đến các nút con đại diện cho các nước đi có thể thực hiện từ trạng thái hiện tại.

Thuật toán thực hiện đệ quy và lựa chọn các nước đi tối ưu bằng cách đánh giá giá trị của các trạng thái kết thúc (ví dụ: thắng, thua, hòa) hoặc đánh giá ước lượng của các trạng thái tạm thời.

Khi đến lượt của một nút làm việc, thuật toán Minimax sẽ lựa chọn nước đi tối ưu cho nút đó dựa trên việc chọn nước đi tối ưu cho các nút con của nó. Đối với người chơi là Maximizer (người chơi cố gắng tối đa hóa giá trị), thuật toán sẽ chọn nước đi có giá trị lớn nhất. Đối với người chơi là Minimizer (người chơi cố gắng tối thiểu hóa giá trị), thuật toán sẽ chọn nước đi có giá trị nhỏ nhất.

***Thuật toán Minimax có thể được mô tả bằng giả mã sau:***

******

Trong giả mã này:

 node là nút hiện tại trong cây trò chơi.

 value(node) là giá trị của nút node.

 node.children là danh sách các nút con của nút node.

* 1. **Mô hình của bài toán**

Thuật toán Minimax có thể được áp dụng cho bất kỳ trò chơi lược đồ nào mà có thể biểu diễn dưới dạng cây trò chơi.

Đầu vào của thuật toán bao gồm:

* Trạng thái ban đầu của trò chơi.
* Các quy tắc và hành động có thể thực hiện trong trò chơi.
* Hàm đánh giá (heuristic) để ước lượng giá trị của các trạng thái tạm thời.
  1. **Ưu điểm và nhược điểm**

*Ưu điểm:*

Tìm kiếm tối ưu: Thuật toán Minimax đảm bảo tìm ra nước đi tối ưu trong trò chơi lược đồ.

Áp dụng rộng rãi: Thuật toán Minimax có thể được áp dụng cho nhiều loại trò chơi lược đồ khác nhau.

Tính chất hợp tác: Thuật toán Minimax cho phép lựa chọn nước đi tối ưu dựa trên giả định rằng đối thủ cũng sẽ chọn nước đi tối ưu cho mình, điều này giúp trong việc phân tích và đánh giá chiến lược.

*Nhược điểm:*

Độ phức tạp: Thuật toán Minimax có thể trở nên rất phức tạp khi áp dụng cho các trò chơi lớn hoặc có không gian trạng thái lớn. Việc duyệt qua toàn bộ cây trò chơi có thể tốn nhiều thời gian và tài nguyên tính toán.

Giả định đối thủ tưởng tượng: Thuật toán Minimax giả định rằng đối thủ luôn chọn nước đi tối ưu cho mình. Tuy nhiên, trong thực tế, đối thủ không thể có thêm thủ công chiến lược tối ưu, dẫn đến kết quả không xác thực.

1. **Sơ lược về thuật toán Naive Bayes Classifier**
   1. **Tư tưởng**

Naive Bayes Classifier dựa trên định lý Bayes để phân loại dữ liệu. Tư tưởng cơ bản của thuật toán là tính toán xác suất để một mẫu dữ liệu thuộc vào một nhóm dựa trên các đặc trưng của nó. Naive Bayes Classifier giả định rằng các đặc trưng độc lập và đóng góp đồng nhất vào quá trình phân loại, điều này làm cho việc tính toán xác suất trở nên dễ dàng hơn

* 1. **Phương pháp**

Thuật toán Naive Bayes Classifier sử dụng hai nguyên tắc chính: nguyên tắc xác suất tiên nghiệm và nguyên tắc xác suất hậu nghiệm. Đầu tiên, xác suất tiên nghiệm được tính toán dựa trên dữ liệu huấn luyện. Sau đó, khi có dữ liệu mới, xác suất hậu nghiệm được tính toán sử dụng xác suất tiên nghiệm và các đặc trưng của mẫu dữ liệu đó. Mẫu dữ liệu sẽ được phân loại vào nhóm có xác suất hậu nghiệm cao nhất.

* 1. **Mô hình của bài toán**

Mô hình Naive Bayes Classifier xây dựng một hàm phân loại dựa trên các đặc trưng của dữ liệu. Các đặc trưng được đại diện bằng các biến ngẫu nhiên độc lập. Công thức Bayes được áp dụng để tính toán xác suất phân loại dựa trên các đặc trưng. Các mô hình Naive Bayes phổ biến bao gồm Naive Bayes Multinomial, Naive Bayes Gaussian và Naive Bayes Bernoulli, phù hợp với các loại dữ liệu khác nhau.

* 1. **Ưu điểm và nhược điểm**

*Ưu điểm:*

Naive Bayes Classifier dễ hiểu và triển khai.

Yêu cầu ít dữ liệu huấn luyện so với các thuật toán phức tạp hơn.

Xử lý tốt với các đặc trưng rời rạc và liên tục.

Chịu được nhiễu và dữ liệu bị thiếu.

*Nhược điểm:*

Giả định về tính độc lập của các đặc trưng có thể không đúng trong thực tế.

Naive Bayes Classifier có thể cho kết quả không chính xác nếu các giả định không được đáp ứng.

Đòi hỏi giả định về phân phối xác suất của dữ liệu.

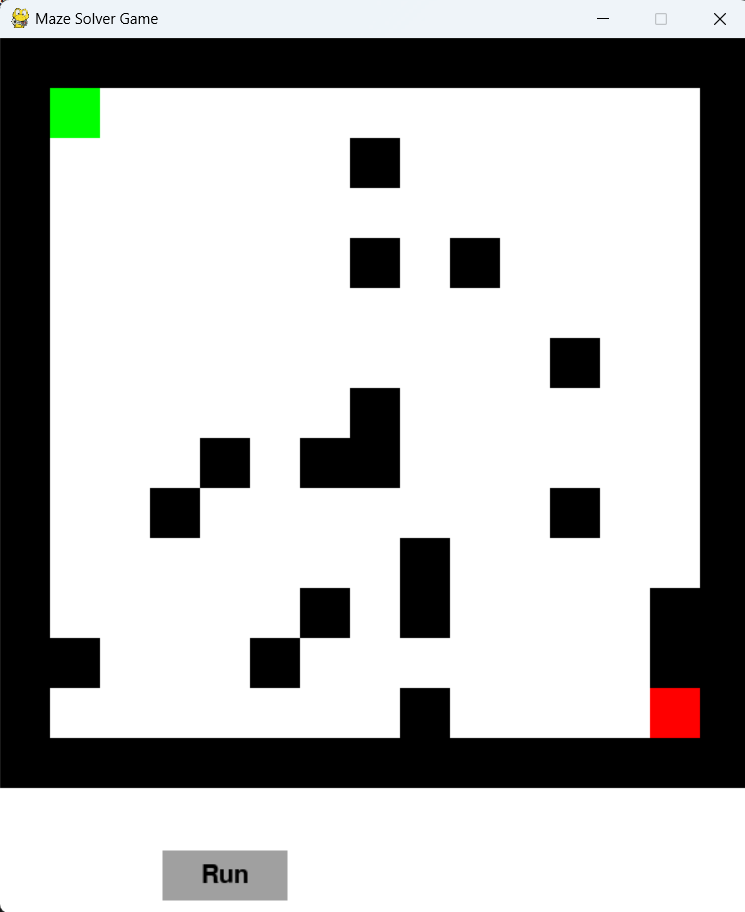
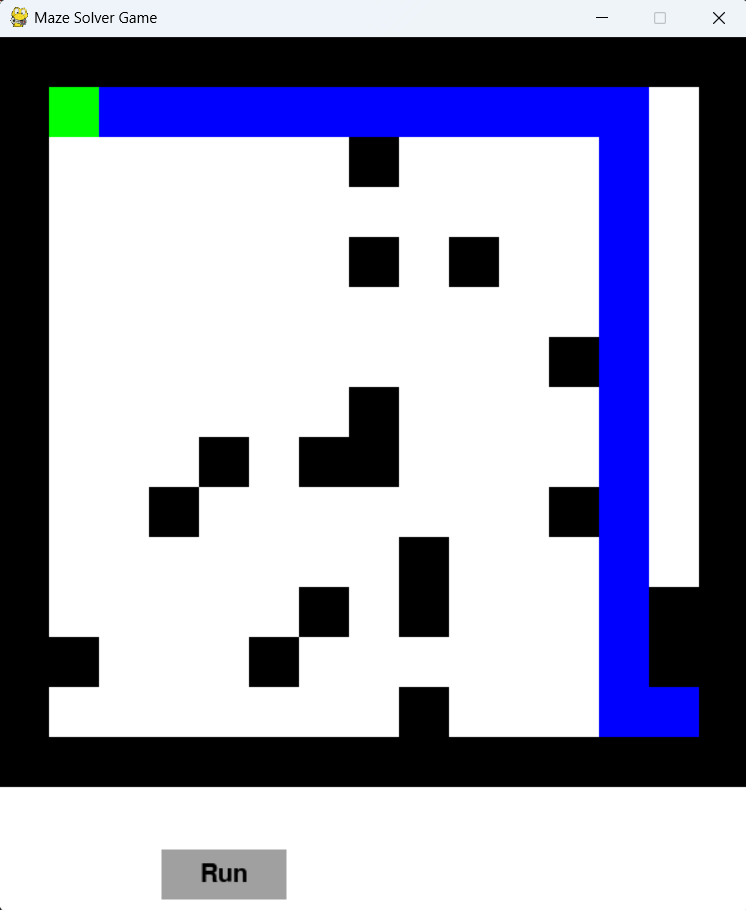
# CHƯƠNG 2: ỨNG DỤNG CỦA THUẬT TOÁN TRONG THỰC TẾ

## I. Giới thiệu trò chơi Maze

Maze là một trò chơi logic thú vị nơi người chơi cố gắng tìm đường từ điểm bắt đầu đến mục tiêu thông qua một mê cung phức tạp. Mục tiêu chính của trò chơi là tìm ra lối thoát khỏi mê cung một cách nhanh nhất có thể.

* 1. **Chạy chương trình trò chơi Maze trên desktop và streamlit**

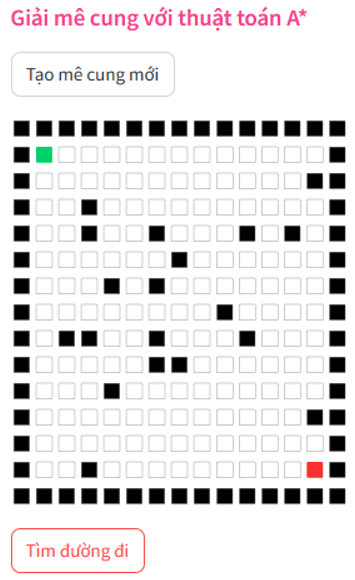
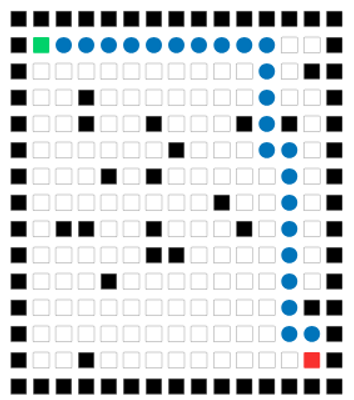
***Chạy chương trình trò chơi Maze trên desktop:***

******

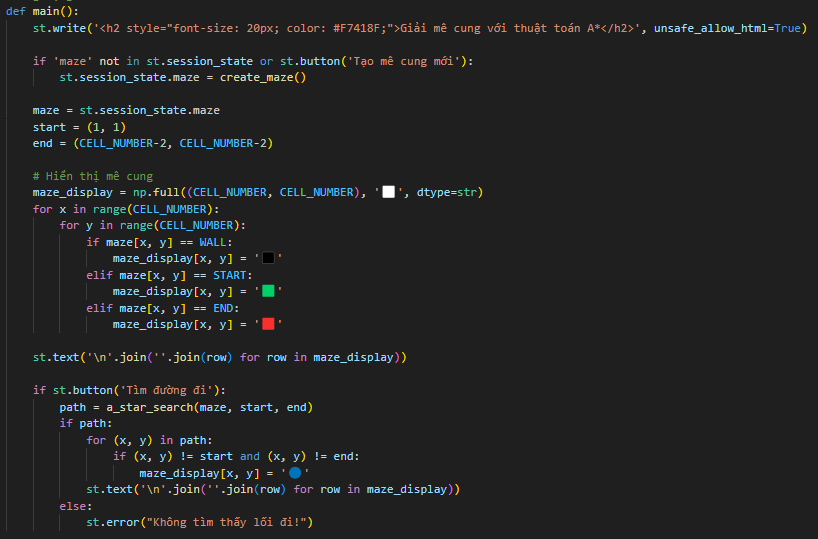
***Code giao diện chương trình trò chơi Maze trên desktop:***

******

***Chạy chương trình trò chơi Maze trên streamlit:***

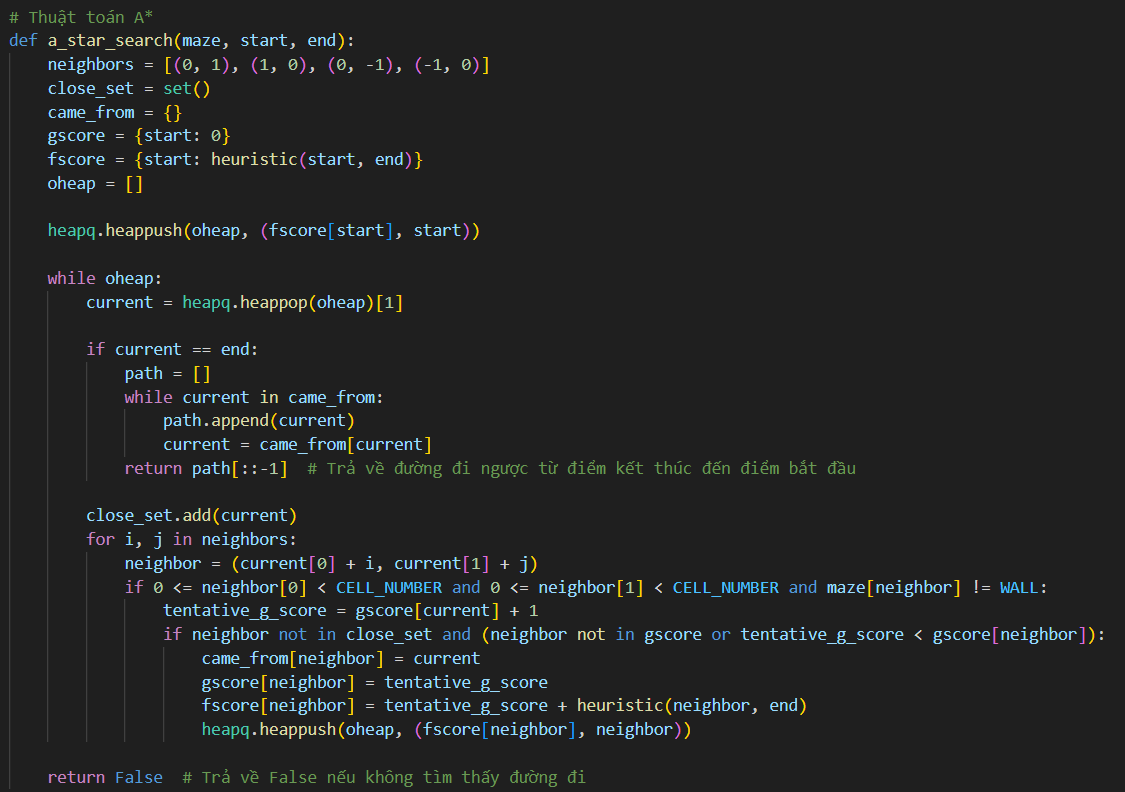


***Code giao diện chương trình trò chơi Maze trên Streamlit:***



* 1. **Áp dụng thuật toán A\* vào trò chơi Maze**

Hàm áp dụng thuật toán A\* vào trò chơi Maze

****

Hàm a\_star\_search cung cấp mã cho thuật toán tìm kiếm A\*, để tìm đường đi ngắn nhất từ điểm bắt đầu đến điểm kết thúc trong một mê cung. Dưới đây là cách thức hoạt động và các thành phần chính của thuật toán tìm kiếm A\*:

* 1. Khởi tạo:

neighbors: Một danh sách các tuple biểu thị các khả năng di chuyển từ vị trí hiện tại (phải, xuống, trái, lên).

close\_set: Một tập hợp dùng để lưu trữ các nút (ô trong mê cung) đã được đánh giá.

came\_from: Một từ điển để theo dõi đường đi, mỗi cặp khóa-giá trị biểu thị nút nào đến từ nút nào.

gscore: Một từ điển lưu chi phí đường đi rẻ nhất biết đến từ nút bắt đầu đến mỗi nút.

fscore: Một từ điển lưu ước tính tổng chi phí từ đầu đến cuối qua mỗi nút. Giá trị này bao gồm chi phí thực tế từ điểm bắt đầu đến nút hiện tại và ước tính chi phí từ nút hiện tại đến điểm kết thúc.

* 1. Quá trình tìm kiếm:

Hàm bắt đầu bằng việc đẩy nút bắt đầu vào oheap, một heap được sử dụng để lưu trữ các nút cần được xem xét dựa trên fscore thấp nhất (ưu tiên xử lý).

Trong mỗi lần lặp của vòng lặp chính, nút hiện tại được lấy ra từ oheap.

Nếu nút hiện tại là điểm kết thúc, thuật toán sẽ tạo và trả lại đường đi bằng cách theo dõi ngược lại từ điểm kết thúc đến điểm bắt đầu thông qua từ điển came\_from.

Nếu không, thuật toán sẽ thực hiện vòng lặp qua mỗi hàng xóm của nút hiện tại để xem xét các nút tiếp theo có thể đi đến. Nếu một hàng xóm chưa được đóng (không nằm trong close\_set) và đường đi đến nó rẻ hơn (tức là có gscore thấp hơn), nó sẽ được thêm vào oheap và cập nhật các giá trị trong came\_from, gscore, và fscore.

* 1. Kết quả:

Nếu tìm thấy đường đi, hàm trả về một danh sách các nút biểu thị đường đi từ điểm bắt đầu đến điểm kết thúc.

Nếu không tìm thấy đường đi, hàm trả về False.

### 1.3. Cách hoạt động của trò chơi Maze

***a. Khởi tạo mê cung***

create\_maze: Hàm này tạo ra một mê cung ngẫu nhiên. Mê cung được biểu diễn bằng một ma trận numpy với các ký hiệu:

0 cho đường đi.

1 cho tường.

2 cho điểm bắt đầu.

3 cho điểm kết thúc.

Các biên của mê cung là tường và một số tường ngẫu nhiên được thêm vào trong mê cung. Điểm bắt đầu và kết thúc được đặt ở hai góc đối diện.

***b. Tính toán heuristic***

heuristic: Sử dụng khoảng cách Manhattan để ước tính khoảng cách từ một ô đến điểm kết thúc, là tổng giá trị tuyệt đối của hiệu số giữa các tọa độ tương ứng của hai điểm.

***c. Thuật toán A\****

a\_star\_search: Hàm này thực hiện thuật toán A\* để tìm đường đi ngắn nhất từ điểm bắt đầu đến điểm kết thúc:

Khởi tạo các biến lưu các ô đã kiểm tra, đường đi, và chi phí đến các ô.

Sử dụng một heap để lấy ô có fscore thấp nhất để xử lý.

Kiểm tra các ô lân cận và cập nhật gscore, fscore, và đường đi khi tìm thấy lộ trình tốt hơn.

Khi tới được điểm kết thúc, trả lại đường đi tìm được. Nếu không tìm thấy đường đi, trả về False.

***d. Chức năng hiển thị và tương tác người dùng***

main: Hàm chính để chạy ứng dụng Streamlit:

Hiển thị mê cung và cho phép người dùng tạo mê cung mới hoặc tìm đường đi trong mê cung hiện tại.

Sử dụng các biểu tượng để thể hiện tường, đường đi, điểm bắt đầu, điểm kết thúc, và đường đi tìm được.

## II. Giới thiệu trò chơi 8 - puzzle

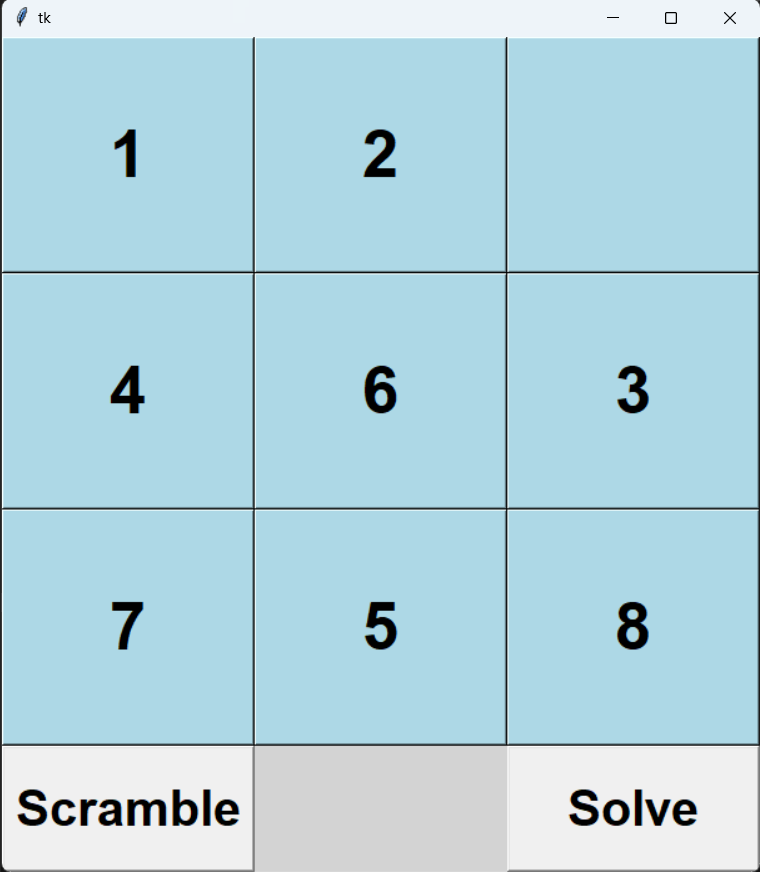
Trò chơi 8-puzzle là một trò chơi xếp hình logic kinh điển, thách thức người chơi trong việc di chuyển các viên gạch số từ vị trí ban đầu thành một trạng thái đích đã cho trước.

Trong trò chơi, có một bảng vuông gồm 9 ô (3x3) được chia thành 8 ô chứa các viên gạch số từ 1 đến 8, và một ô trống. Mục tiêu của trò chơi là di chuyển các viên gạch số để sắp xếp chúng theo trật tự tăng dần từ trái qua phải và từ trên xuống dưới, với ô trống nằm ở vị trí cuối cùng.

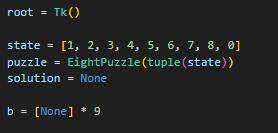
Người chơi có thể di chuyển các viên gạch số bằng cách hoán đổi vị trí giữa viên gạch số và ô trống. Mỗi lần di chuyển, người chơi chỉ có thể hoán đổi ô trống với một viên gạch nằm cạnh ô trống (trên, dưới, trái hoặc phải). Mục tiêu là tìm ra một chuỗi di chuyển các bước nhỏ để đưa các viên gạch về vị trí đúng.

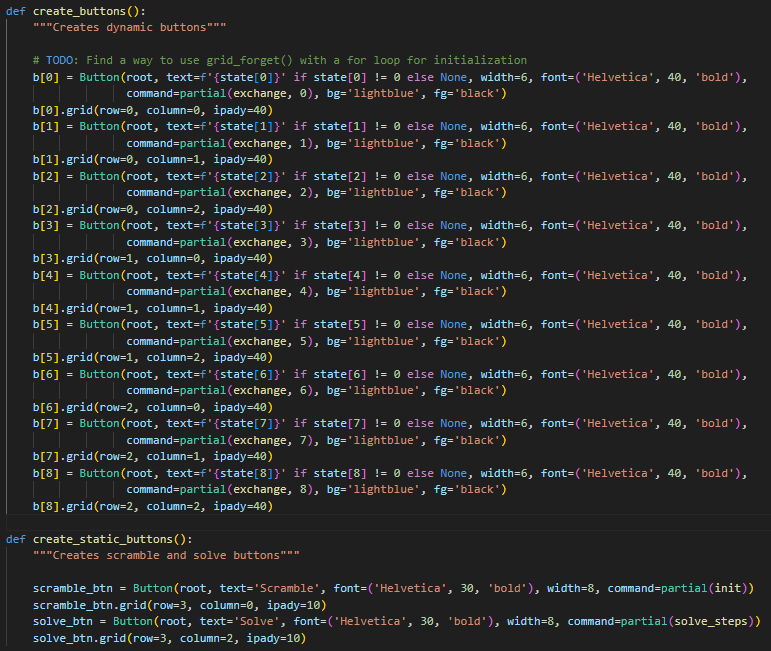
### 2.1. Chạy chương trình trò chơi 8 - puzzle trên desktop và streamlit

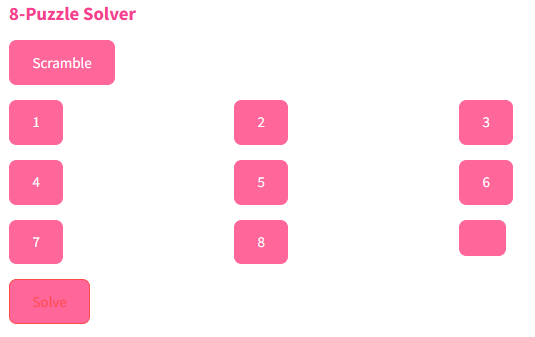
***Chạy chương trình trò chơi 8 - puzzle trên desktop:***

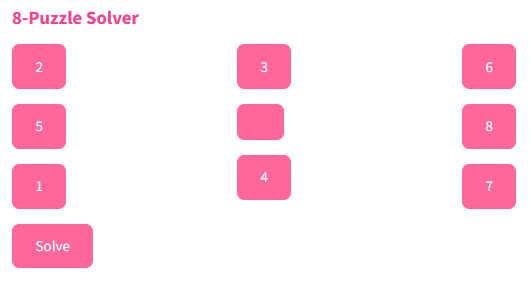
******

***Code giao diện chương trình trò chơi 8 – puzzle trên desktop:***

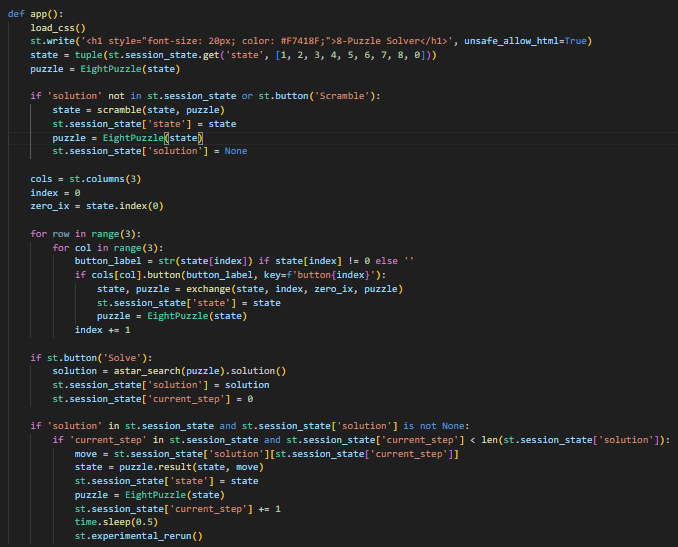
******

******

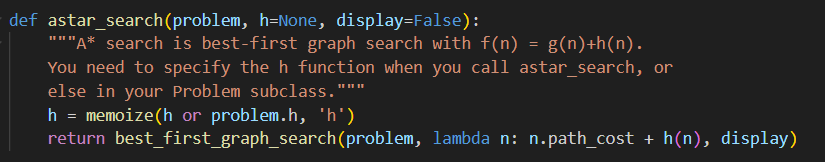
***Chạy chương trình trò chơi 8 – puzzle trên streamlit:***

****

***Code giao diện chương trình trò chơi 8 – puzzle trên streamlit:***

****

### 2.2. Áp dụng thuật toán Minimax vào trò chơi 8 – puzzle

****

*Hàm Tìm Kiếm A\* (astar\_search)*

astar\_search: Hàm này thực hiện thuật toán tìm kiếm A\*, là một hình thức tìm kiếm đầu tiên tốt nhất dựa trên đồ thị với giá trị hàm f(n) = g(n) + h(n). Trong đó, g(n) là chi phí đường đi từ nút gốc đến nút hiện tại, và h(n) là hàm heuristic, ước tính chi phí từ nút hiện tại đến mục tiêu.

h: Là hàm heuristic, cần được định nghĩa khi gọi hàm astar\_search hoặc trong lớp vấn đề con. Nó được ghi nhớ (memoize) để tránh tính toán lại nhiều lần.

lambda n: n.path\_cost + h(n): Là hàm cho best\_first\_graph\_search, tính toán giá trị f của mỗi nút.

### 2.3. Cách hoạt động của trò chơi 8 - puzzle

*scramble(state, puzzle)*

Hàm này xáo trộn trạng thái hiện tại của bài toán 8-Puzzle bằng cách áp dụng 60 hành động ngẫu nhiên. Mục đích là tạo ra một trạng thái khởi đầu thách thức cho người dùng hoặc để kiểm tra giải thuật.

*exchange(state, index, zero\_ix, puzzle)*

Hàm này cho phép người dùng đổi chỗ giữa ô được chọn và ô trống trên bảng. Điều này chỉ xảy ra nếu thao tác là hợp lệ theo quy tắc của trò chơi.

*Hàm app()*

Quản lý trạng thái: Quản lý trạng thái của trò chơi thông qua st.session\_state, bao gồm trạng thái bảng hiện tại và bất kỳ giải pháp nào được tìm thấy.

Xử lý nhập từ người dùng: Người dùng có thể xáo trộn bảng hoặc yêu cầu giải bài toán.

Hiển thị và tương tác với bảng: Bảng được hiển thị như một dãy các nút mà người dùng có thể nhấp vào để di chuyển các ô.

Tìm kiếm giải pháp: Khi người dùng nhấp vào "Solve", thuật toán A\* được sử dụng để tìm giải pháp, và bất kỳ bước nào trong giải pháp đó sẽ được tự động thực hiện.

Cập nhật liên tục: Sau mỗi bước giải, ứng dụng chờ đợi nửa giây rồi tự động cập nhật lại để hiển thị trạng thái mới.

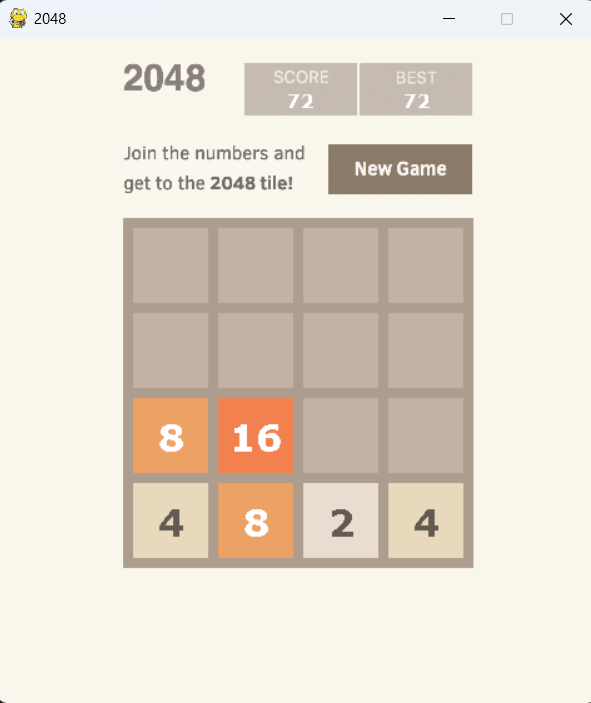
## III. Giới thiệu trò chơi 2048

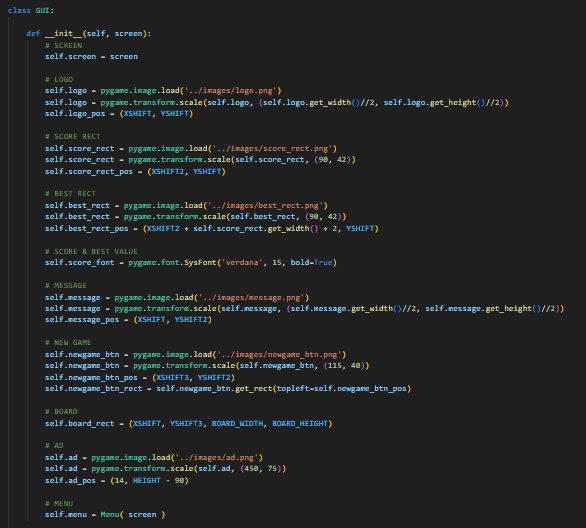
Trong trò chơi 2048, bạn sẽ thấy một ô vuông có kích thước 4x4 chứa các ô con số. Mục tiêu của trò chơi là kết hợp các ô số có cùng giá trị để tạo ra một ô số lớn hơn. Ban đầu, bạn sẽ có hai ô số 2 trên bảng. Bạn cần di chuyển các ô số này bằng cách sử dụng các phím mũi tên (lên, xuống, trái, phải) để tạo ra các ô số lớn hơn.

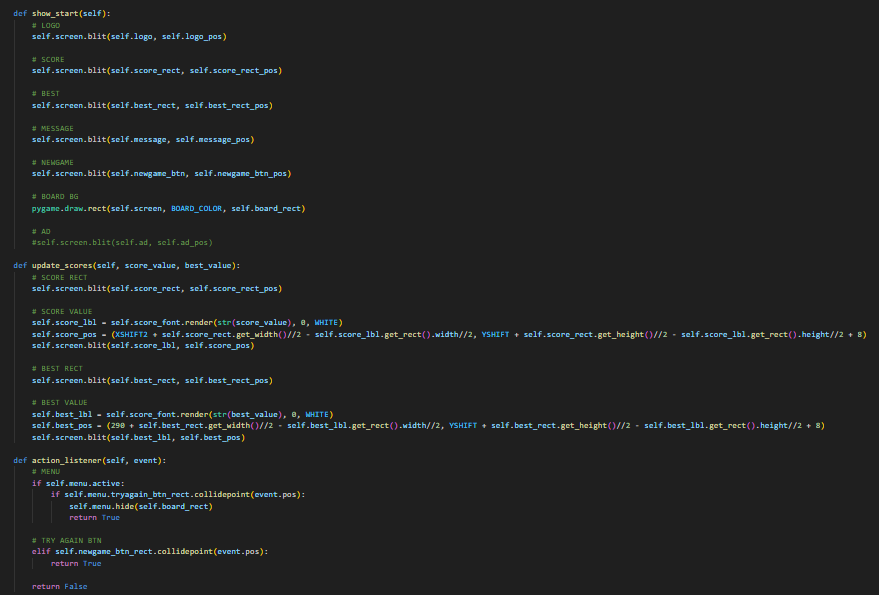
Khi bạn thực hiện một lần di chuyển, tất cả các ô số trên bảng sẽ di chuyển theo hướng đó cho đến khi chạm vào một bức tường hoặc gặp một ô số khác có giá trị khác. Nếu hai ô số cùng giá trị chạm vào nhau, chúng sẽ kết hợp thành một ô số mới có giá trị gấp đôi (ví dụ: 2 + 2 = 4, 4 + 4 = 8) và một ô trống sẽ xuất hiện. Mục tiêu cuối cùng là tạo ra một ô số có giá trị 2048 trên bảng.

### 3.1. Chạy chương trình trò chơi 2048 trên desktop và streamlit

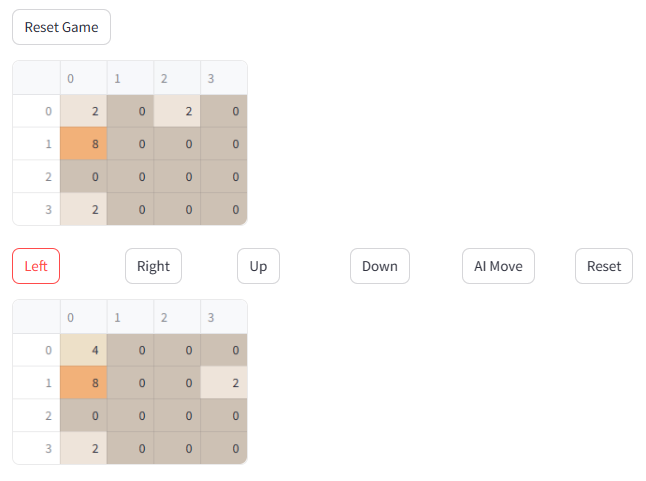
***Chạy chương trình trò chơi 2048 trên streamlit:***

******

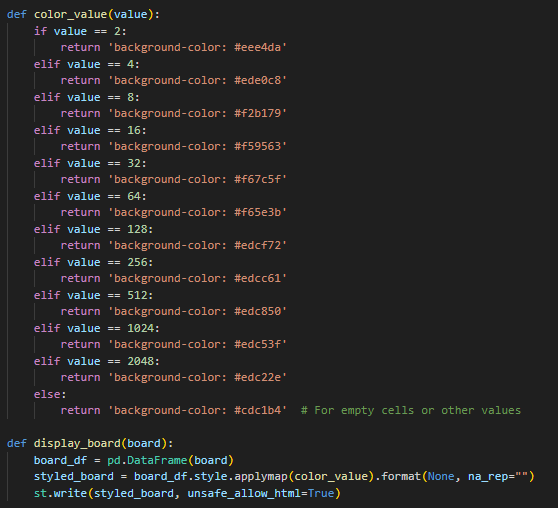
***Code chạy chương trình trò chơi 2048 trên desktop:***

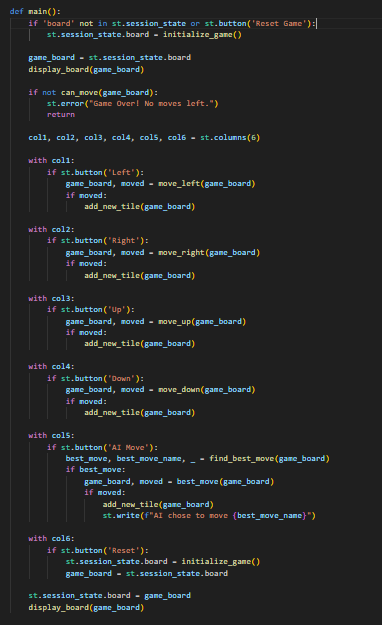
******

***Chạy chương trình trò chơi 2048 trên streamlit:***

****

***Code chạy chương trình trò chơi 2048 trên streamlit:***

****

****

### 3.2. Áp dụng thuật toán Minimax vào trò chơi 2048

Tham số:

board: Bàn cờ hiện tại, được biểu diễn bởi một mảng numpy.

depth: Độ sâu hiện tại của cây tìm kiếm. Độ sâu giới hạn việc tiếp tục tìm kiếm của thuật toán.

alpha: Giá trị tốt nhất mà người chơi tối đa hóa (maximizing player) có thể đảm bảo; bắt đầu từ -infinity.

beta: Giá trị tốt nhất mà người chơi tối thiểu hóa (minimizing player) có thể đảm bảo; bắt đầu từ infinity.

is\_maximizing: Biến boolean cho biết người chơi hiện tại có phải là người tối đa hóa không.

Cách thức hoạt động của thuật toán:

Kiểm tra Điều Kiện Dừng: Nếu độ sâu bằng 0 hoặc không còn nước đi hợp lệ trên bàn cờ, thuật toán sẽ trả về điểm số hiện tại của bàn cờ và không có nước đi nào.

Xử lý cho Người chơi Tối đa hóa (Maximizing Player):

Khởi tạo max\_eval với giá trị vô cùng nhỏ.

Duyệt qua các nước đi có thể (chẳng hạn như di chuyển trái, phải, lên, xuống).

Đối với mỗi nước đi, tạo ra bàn cờ mới và kiểm tra nếu nước đi đó được thực hiện (moved là True).

Gọi đệ quy hàm minimax với độ sâu giảm xuống, chuyển sang người chơi tối thiểu hóa.

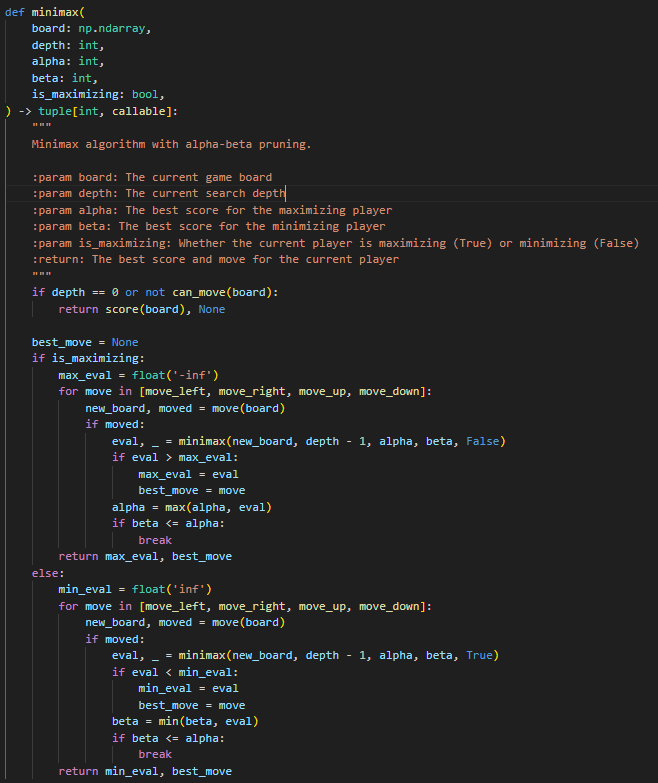
Cập nhật max\_eval và best\_move nếu giá trị đánh giá (eval) cao hơn giá trị hiện tại.

Cập nhật alpha và kiểm tra điều kiện cắt tỉa (beta <= alpha).

Xử lý cho Người chơi Tối thiểu hóa (Minimizing Player):

Tương tự như trên nhưng cập nhật min\_eval, kiểm tra và cập nhật beta.

Nếu beta nhỏ hơn hoặc bằng alpha, thì cắt tỉa nhánh hiện tại để không tìm kiếm thêm.

****

### 3.3. Cách hoạt động của trò chơi 2048

display\_board(board) Hiển thị bàn cờ trong trò chơi sử dụng DataFrame của pandas và áp dụng định dạng màu sắc đã định nghĩa.

main() Hàm chính của ứng dụng Streamlit. Quản lý trạng thái của trò chơi, tương tác người dùng, và hiển thị giao diện.

Khởi tạo trò chơi, xử lý nhập của người dùng qua các nút bấm để di chuyển các ô trên bàn cờ, và gọi AI để tìm nước đi tối ưu.

initialize\_game() Khởi tạo bàn cờ với tất cả các ô là 0 và thêm hai ô có giá trị là 2 vào vị trí ngẫu nhiên.

add\_new\_tile(board) Thêm một ô mới với giá trị 2 vào vị trí ngẫu nhiên trên bàn cờ.

compress(board) Nén các ô trong bàn cờ để chuẩn bị cho thao tác gộp các ô giống nhau.

merge(board) Gộp các ô giống nhau trên bàn cờ (các ô cạnh nhau có cùng giá trị sẽ được gộp lại và nhân đôi giá trị).

move\_left(board), move\_right(board), move\_up(board), move\_down(board)

Thực hiện các thao tác di chuyển trên bàn cờ. Mỗi hàm này gọi hàm compress và merge để xử lý bàn cờ sau mỗi lượt di chuyển.

can\_move(board) Kiểm tra xem còn nước đi hợp lệ trên bàn cờ hay không.

score(board) Tính điểm hiện tại trên bàn cờ dựa vào giá trị lớn nhất trên bàn cờ.

minimax(...) Thuật toán Minimax với cắt tỉa Alpha-Beta, được sử dụng để tìm nước đi tối ưu cho AI trong trò chơi.

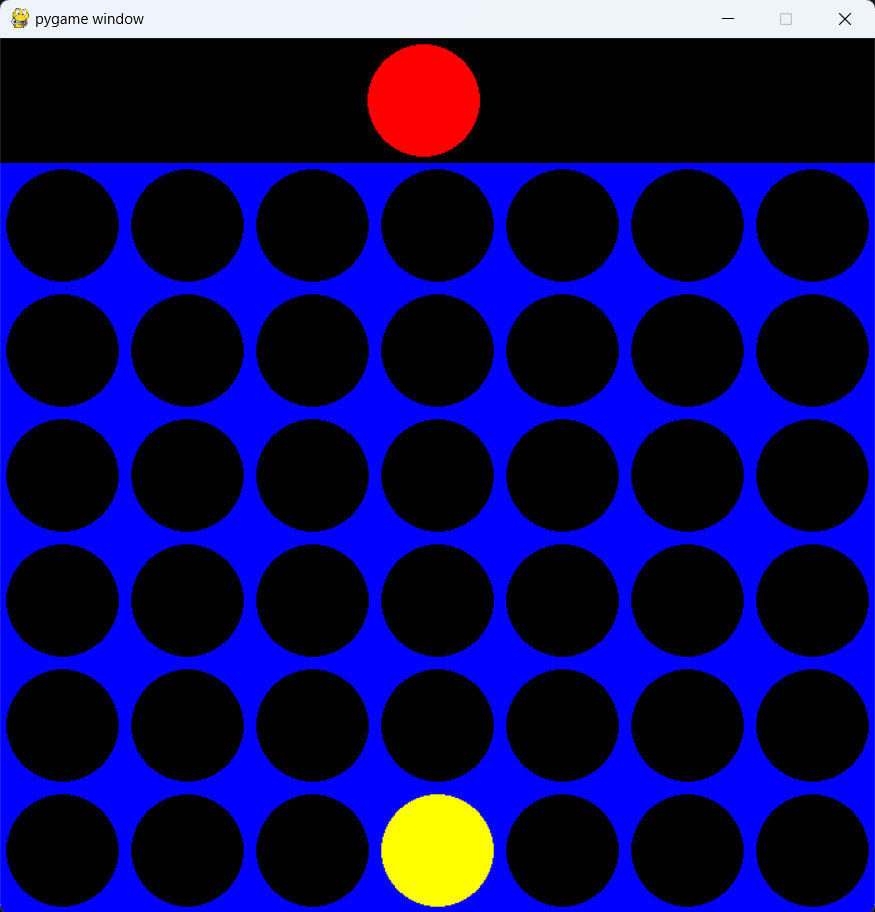
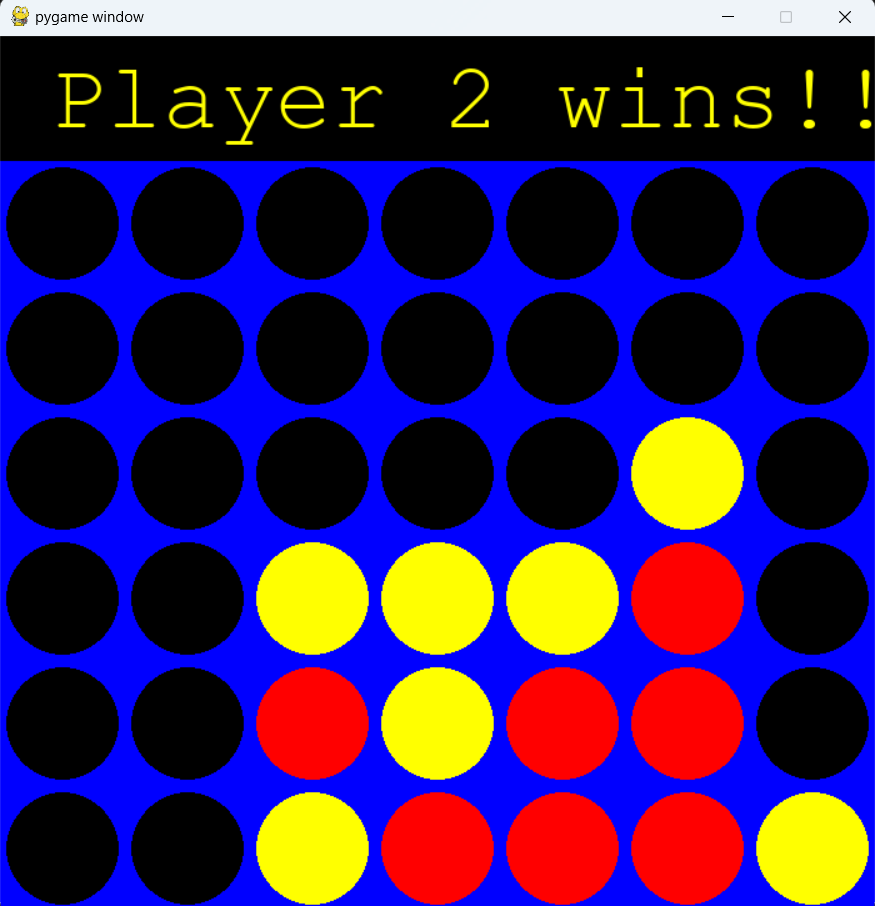
find\_best\_move(board) Tìm nước đi tốt nhất dựa vào thuật toán Minimax.

## IV. Giới thiệu trò chơi Connect Four

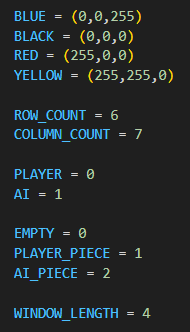
Connect Four là xếp 4 quân cờ của mình theo hàng dọc, hàng ngang hoặc chéo trên một bảng có kích thước 7x6. Hai người chơi sẽ luân phiên đặt quân cờ của mình vào các cột. Quân cờ sẽ rơi xuống cột và chiếm vị trí trống đầu tiên từ dưới lên (hoặc từ trên xuống dưới). Người chơi nào đạt được chuỗi 4 quân cờ liên tiếp đầu tiên trên bảng là người chiến thắng.

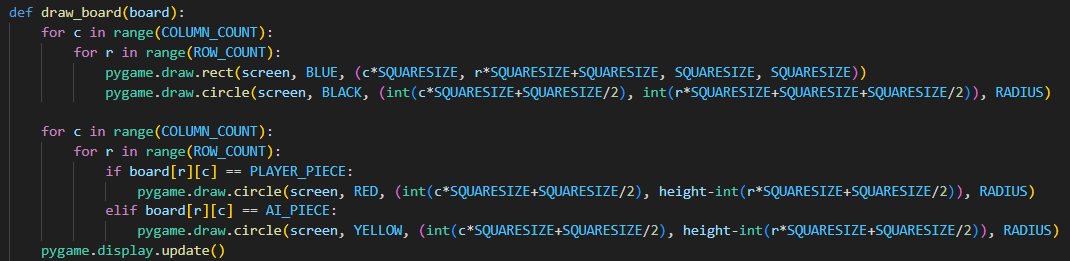
### 4.1. Chạy chương trình trò chơi Connect Four trên desktop và streamlit

***Chạy chương trình trò chơi Connect Four trên streamlit:***

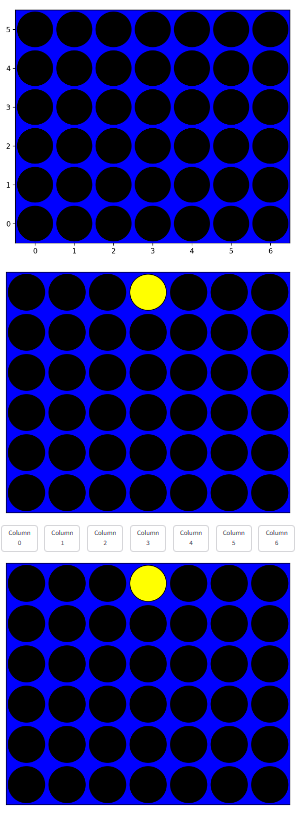
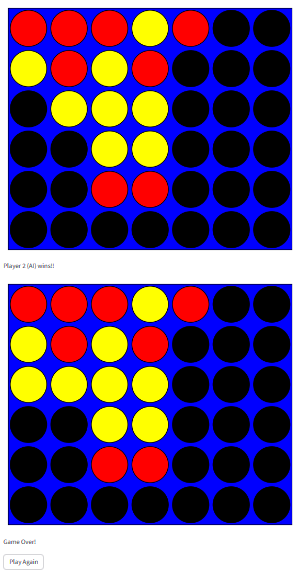
******

***Code giao diện chương trình trò chơi Connect Four trên desktop:***

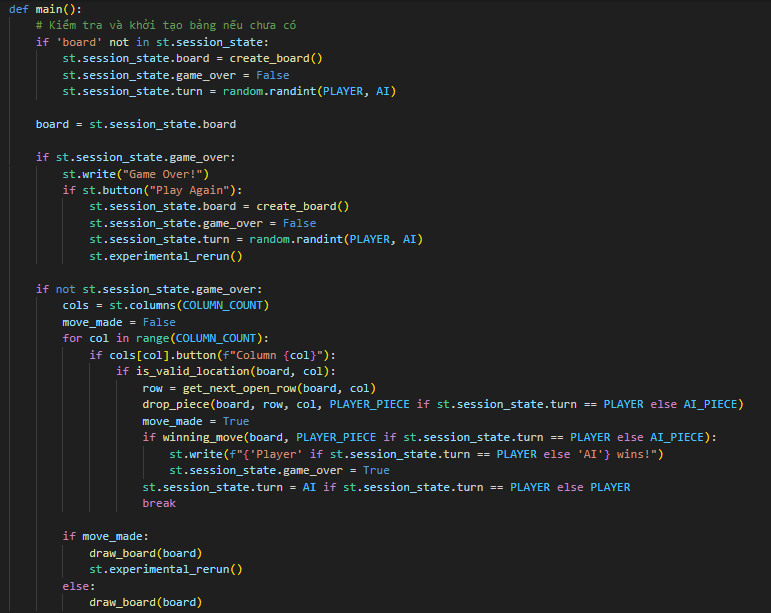
******

******

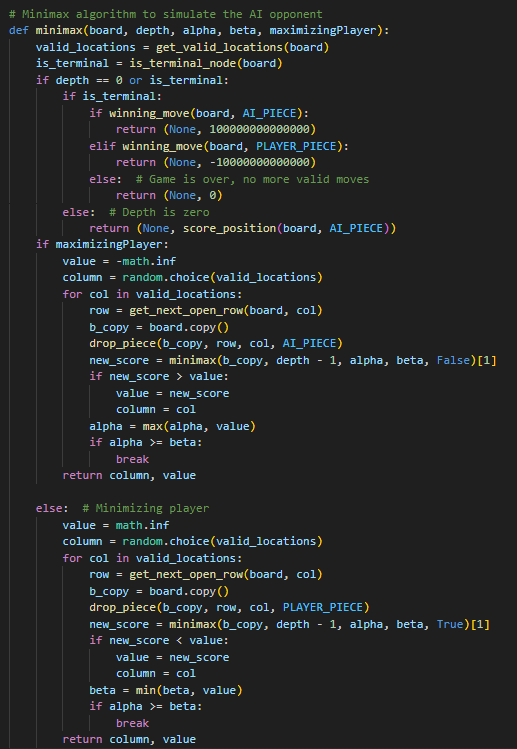
***Chạy chương trình trò chơi Connect Four trên streamlit:***

******

***Code giao diện chương trình trò chơi Connect Four trên streamlit:***

******

### 4.2. Áp dụng thuật toán Minimax vào trò chơi Connect Four

****

minimax(board, depth, alpha, beta, maximizingPlayer): Hàm này triển khai giải thuật Minimax với cắt tỉa Alpha-Beta. Nó được sử dụng để tính toán nước đi tối ưu trong các trò chơi trên bảng hai người chơi.

Các thành phần chính của hàm:

1. Các biến đầu vào:

board: Bảng trò chơi hiện tại.

depth: Độ sâu còn lại của cây trò chơi mà thuật toán sẽ xem xét.

alpha và beta: Giá trị để thực hiện cắt tỉa, giúp giảm số lượng nút cần phải xem xét.

maximizingPlayer: Một biến boolean cho biết đây có phải là lượt của người chơi cố gắng tối đa hóa điểm số không (thường là AI).

1. Xử lý Trường Hợp Cơ Sở:

Nếu độ sâu bằng 0 hoặc nút là nút kết thúc (không còn nước đi hợp lệ nào), thuật toán sẽ trả về giá trị cho vị trí hiện tại.

Nếu đây là nút kết thúc, và AI là người chiến thắng, trả về giá trị rất cao. Nếu người chơi là người chiến thắng, trả về giá trị rất thấp. Nếu hòa, trả về 0.

1. Quá trình Tối đa hóa/Minh họa:

Nếu là lượt của người tối đa hóa (thường là AI), thuật toán tìm kiếm nước đi có điểm số cao nhất có thể.

Nếu là lượt của người chơi tối thiểu hóa, thuật toán sẽ tìm nước đi có điểm số thấp nhất.

1. Cắt tỉa Alpha-Beta:

alpha là giá trị tốt nhất mà người tối đa hóa có thể đảm bảo từ lịch sử duyệt trước đó.

beta là giá trị tốt nhất mà người tối thiểu hóa có thể đảm bảo.

Nếu alpha lớn hơn hoặc bằng beta, nhánh hiện tại sẽ bị cắt bỏ, không cần xem xét thêm.

1. Lựa chọn nước đi:

Đối với mỗi nước đi hợp lệ, thuật toán sẽ thử đặt quân cờ vào bảng (bản sao của bảng hiện tại) và gọi đệ quy hàm minimax cho bảng mới với độ sâu giảm xuống.

### 4.3. Cách hoạt động của trò chơi Connect Four

1. Khởi tạo và quản lý bảng chơi

create\_board(): Tạo một bảng mới dưới dạng mảng NumPy với kích thước được xác định bởi ROW\_COUNT và COLUMN\_COUNT, khởi tạo tất cả các giá trị là 0 (rỗng).

drop\_piece(): Đặt quân cờ vào vị trí cụ thể trên bảng đã được kiểm tra là hợp lệ.

is\_valid\_location(): Kiểm tra xem cột đã chọn có chỗ trống ở hàng trên cùng hay không.

get\_next\_open\_row(): Tìm hàng trống tiếp theo trong cột để đặt quân cờ.

1. Kiểm tra điều kiện thắng:

winning\_move(): Kiểm tra xem đã có quân cờ nào tạo thành một dãy liên tục bốn quân cùng loại theo chiều ngang, dọc, chéo xuôi, hay chéo ngược chưa.

1. Tính điểm và đánh giá bảng chơi:

evaluate\_window(): Đánh giá và tính điểm cho một cửa sổ con (một dãy bốn ô) dựa trên số lượng quân cờ của người chơi hoặc AI.

score\_position(): Tính toán điểm số cho toàn bộ bảng dựa trên tất cả các cửa sổ có thể và ưu tiên các vị trí trung tâm.

1. Thuật toán Minimax:

minimax(): Thuật toán nhằm mô phỏng người chơi AI. Nó đánh giá các nước đi tiềm năng để tìm ra nước đi tốt nhất dựa trên độ sâu tối đa được cho và cắt tỉa các nhánh không cần thiết nhờ giá trị alpha và beta.

get\_valid\_locations(): Trả về danh sách các cột có thể đặt quân cờ.

pick\_best\_move(): Lựa chọn nước đi tốt nhất dựa trên điểm số đánh giá cho mỗi cột hợp lệ.

1. Vẽ bảng chơi và xử lý giao diện người dùng:

draw\_board(): Vẽ bảng chơi sử dụng Matplotlib.

Main game loop: Xử lý lượt chơi của người và AI thông qua giao diện Streamlit, bao gồm nút để chọn cột và đặt quân, hiển thị trạng thái trò chơi và tạo mới trò chơi.

1. Luồng chính của chương trình:

Chương trình chính main() và một vòng lặp game chính được tích hợp sẵn trong Streamlit để xử lý các sự kiện, như người dùng đặt quân hoặc máy tính (AI) thực hiện nước đi tiếp theo. Trò chơi tiếp tục cho đến khi có một người thắng hoặc trò chơi kết thúc.

## V. Giới thiệu trò chơi Tic-tac-toe

Tic-tac-toe, còn được gọi là Xs and Os hoặc Noughts and Crosses, là một trò chơi giấy bút đơn giản và phổ biến. Trò chơi này được biết đến trên toàn thế giới và thường được chơi bởi hai người, sử dụng một bảng vuông 3x3.

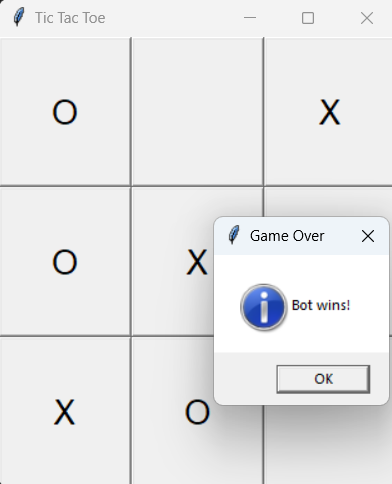
Mục tiêu của Tic-tac-toe là tạo thành một hàng ngang, hàng dọc hoặc hàng chéo gồm ba con X hoặc ba con O. Hai người chơi lần lượt đặt ký hiệu của mình lần lượt là X và O trên các ô trống của bảng cho đến khi một người chiến thắng hoặc bảng đầy đủ.

Luật chơi đơn giản: người chơi X đi trước, sau đó người chơi O, và tiếp tục luân phiên cho đến khi một người thắng hoặc hòa. Người chơi cố gắng đặt ký hiệu của mình vào một ô trống trên bảng một cách thông minh để tạo thành một chuỗi gồm ba ký hiệu của mình.

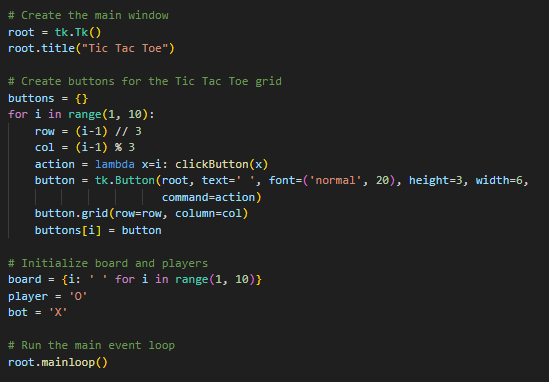
Nếu một người chơi đạt được chuỗi gồm ba ký hiệu của mình trên một hàng ngang, hàng dọc hoặc hàng chéo, anh ta sẽ là người chiến thắng. Nếu bảng đầy đủ mà không có ai thắng, trò chơi sẽ kết thúc với kết quả hòa.

### 5.1. Chạy chương trình trò chơi Tic-tac-toe trên desktop và streamlit

***Chạy chương trình trò chơi* Tic-tac-toe *trên desktop:***

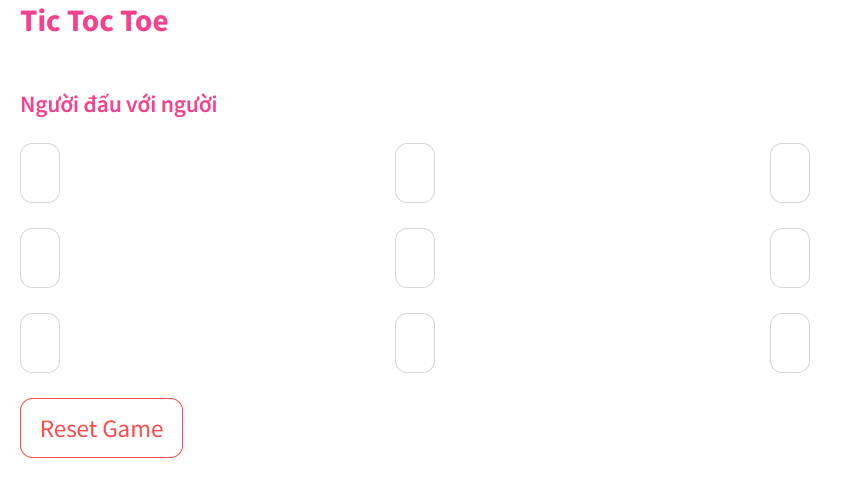
****

***Code giao diện chương trình trò chơi* Tic-tac-toe *trên desktop:***

****

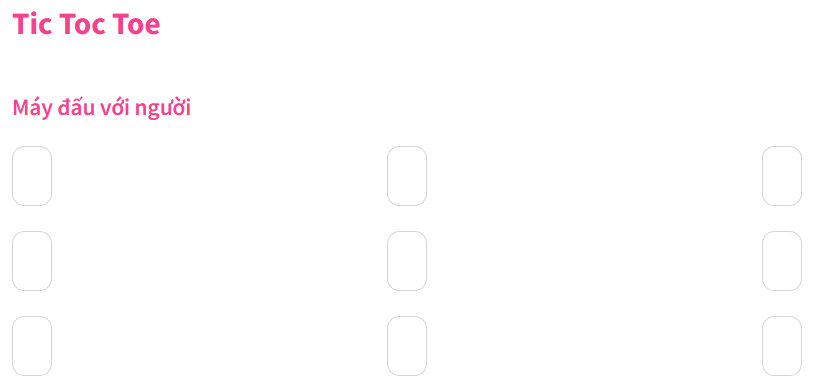
***Chạy chương trình trò chơi* Tic-tac-toe *trên streamlit:***

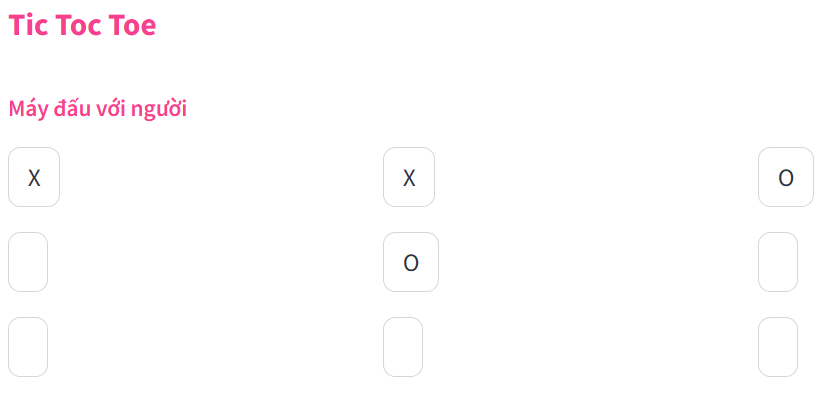
***a. Người đấu với người***

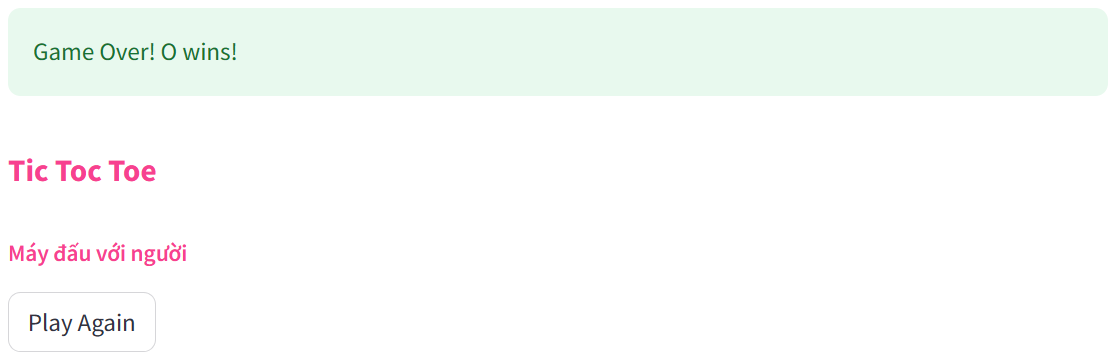
****

****

***b. Máy đấu với người***

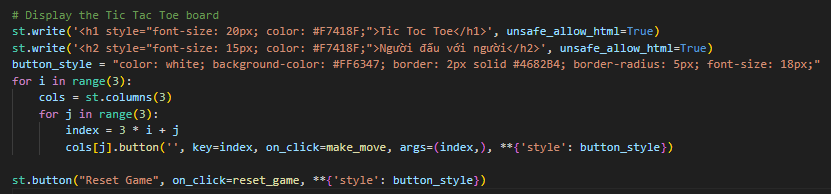
******

******

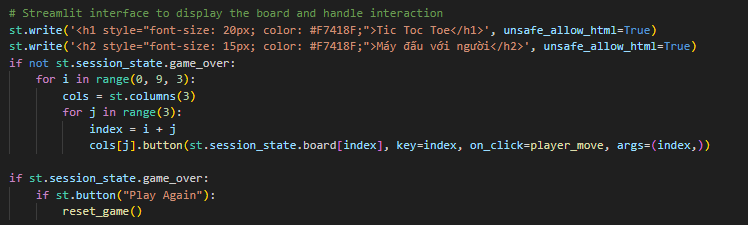
******

***Code giao diện chương trình trò chơi* Tic-tac-toe *trên streamlit:***

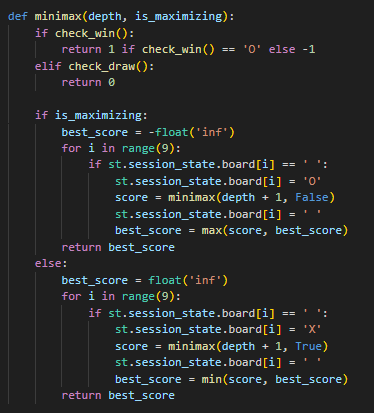
***Người đấu với người***

******

***Máy đấu với người***

******

### 5.2. Áp dụng thuật toán Minimax vào trò chơi Tic-toc-toe

****

1. Các Thành Phần Chính:

depth: Độ sâu hiện tại của lời gọi đệ quy. Độ sâu này tăng lên với mỗi lượt đệ quy.

is\_maximizing: Một biến boolean xác định liệu người chơi hiện tại có phải là người tối đa hóa (sử dụng 'O') hay người tối thiểu hóa (sử dụng 'X').

1. Logic Của Hàm:

* Kiểm Tra Điều Kiện Kết Thúc:

check\_win(): Kiểm tra xem có người chơi nào đã thắng cuộc chưa. Nếu người chơi 'O' thắng, trả về 1; nếu người chơi 'X' thắng, trả về -1.

check\_draw(): Kiểm tra xem trò chơi có kết thúc với kết quả hòa không. Nếu hòa, trả về 0.

* Đánh Giá Nước Đi:

Nếu là lượt của người tối đa hóa (is\_maximizing = True), thuật toán sẽ tìm kiếm điểm số cao nhất có thể đạt được:

Duyệt qua từng ô trên bảng (for i in range(9)), nếu ô đó trống (' '), đặt 'O' vào đó và gọi đệ quy minimax với is\_maximizing = False.

Sau khi đệ quy trả về, khôi phục lại bảng (st.session\_state.board[i] = ' ').

Cập nhật best\_score nếu điểm số thu được cao hơn.

Nếu là lượt của người tối thiểu hóa, thuật toán sẽ tìm kiếm điểm số thấp nhất có thể đạt được:

Tương tự như trên nhưng đặt 'X' thay vì 'O' và gọi đệ quy với is\_maximizing = True.

### 5.3. Cách hoạt động của trò chơi Tic-toc-toe

1. Khởi tạo trạng thái:

st.session\_state được sử dụng để lưu trữ các biến như game\_over, board, và current\_player. Nó đảm bảo rằng trạng thái của trò chơi được giữ nguyên qua các lần tải lại trang.

1. Kiểm tra kết quả:

check\_win(): Kiểm tra có người thắng cuộc không bằng cách duyệt qua các dòng, cột và đường chéo để xem có ba dấu giống nhau liên tiếp không.

check\_draw(): Kiểm tra xem trò chơi có kết thúc hòa hay không (tức là không còn ô trống nào trên bảng).

1. Chức năng di chuyển:

player\_move(index): Xử lý nước đi của người chơi và sau đó chuyển lượt cho AI nếu trò chơi chưa kết thúc.

ai\_move(): Tìm kiếm nước đi tối ưu cho AI dựa trên thuật toán Minimax. AI sẽ đánh vào ô nào đó có điểm số Minimax cao nhất.

minimax(depth, is\_maximizing): Tính toán điểm số tối ưu cho từng nước đi có thể dựa trên thuật toán Minimax.

1. Đặt lại trò chơi:

reset\_game(): Đặt lại bảng và các biến liên quan để bắt đầu lại từ đầu.

1. Giao diện người dùng Streamlit:

Một lưới 3x3 được tạo với các nút, mỗi nút đại diện cho một ô trong trò chơi Tic-Tac-Toe. Khi người dùng nhấp vào một nút (nếu ô đó trống), hàm player\_move sẽ được gọi.

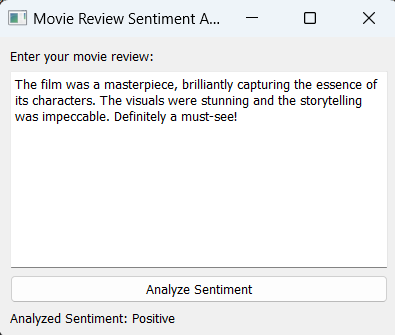
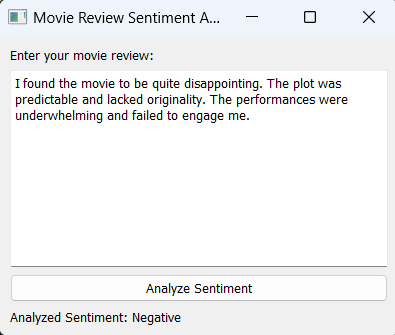
Một nút cho phép người chơi bắt đầu lại trò chơi khi trò chơi kết thúc.

## VI. Giới thiệu mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim

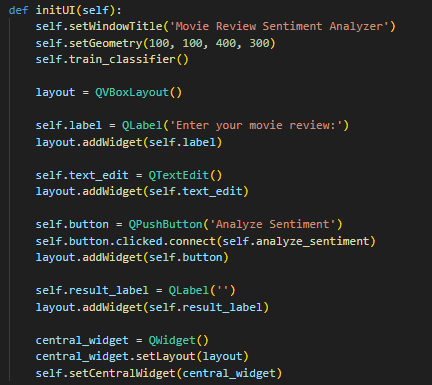
Mô hình phân loại Naive Bayes trong việc phân loại cảm xúc của các bình luận về phim là một ứng dụng phổ biến của thuật toán này trong xử lý ngôn ngữ tự nhiên. Mô hình này được sử dụng để xác định xem một bình luận về phim có tính tích cực (positive) hay tiêu cực (negative) dựa trên nội dung của nó.

### 6.1. Chạy mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim trên desktop và streamlit

***Chạy mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim trên desktop:***

****

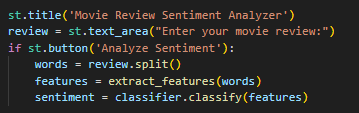
***Code giao diện mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim trên desktop:***



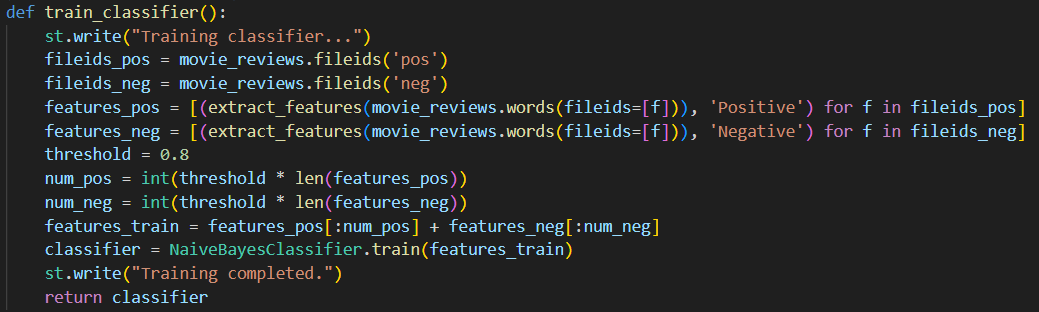
***Chạy mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim trên streamlit:***

******

***Code giao diện mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim trên streamlit:***

******

### 6.2. Áp dụng thuật toán Naive Bayes Classifier vào mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim

****

Hàm train\_classifier() để huấn luyện một mô hình phân loại cảm xúc của các bình luận về phim bằng thuật toán Naive Bayes

1. Xây dựng tập dữ liệu huấn luyện:

Dữ liệu từ tập dữ liệu movie\_reviews trong thư viện NLTK được sử dụng. Tập dữ liệu này chứa các bình luận về phim được gán nhãn là tích cực hoặc tiêu cực.

Mỗi bình luận được chia thành các từ và đặc trưng được tạo ra bằng hàm extract\_features, trong đó mỗi từ là một đặc trưng và được đánh dấu là True.

Xác định ngưỡng (threshold) để chia tập dữ liệu thành tập huấn luyện và tập kiểm tra. Trong trường hợp này, 80% dữ liệu được sử dụng cho huấn luyện.

Tạo tập dữ liệu huấn luyện bằng cách kết hợp các đặc trưng tích cực và tiêu cực.

1. Huấn luyện mô hình:

Mô hình Naive Bayes Classifier được huấn luyện trên tập dữ liệu huấn luyện bằng cách sử dụng phương thức train() từ thư viện NLTK.

Xuất thông báo "Training completed." để thông báo rằng quá trình huấn luyện đã hoàn thành.

1. Dự đoán cảm xúc:

Khi người dùng nhập một bình luận mới, nó được chia thành các từ và đặc trưng được tạo ra.

Mô hình phân loại Naive Bayes được sử dụng để dự đoán cảm xúc của bình luận dựa trên các đặc trưng này.

1. Hiển thị kết quả:

Kết quả dự đoán cảm xúc được hiển thị trên giao diện người dùng bằng cách thiết lập nội dung của result\_label.

### 6.3. Cách hoạt động của mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim

1. Hàm extract\_features:

Định nghĩa hàm extract\_features(words) để trích xuất các đặc trưng từ danh sách các từ. Mỗi từ được chuyển đổi thành một từ điển với key là từ đó và giá trị là True.

1. Hàm train\_classifier:

Định nghĩa hàm train\_classifier() để huấn luyện mô hình phân loại. Quá trình này bao gồm chia dữ liệu thành tập huấn luyện và tập kiểm tra, trích xuất các đặc trưng từ các bình luận và huấn luyện một mô hình Naive Bayes Classifier trên tập dữ liệu huấn luyện.

1. Khởi tạo và sử dụng mô hình phân loại:

Khởi tạo mô hình phân loại bằng cách gọi hàm train\_classifier().

Tạo giao diện người dùng với Streamlit, cho phép người dùng nhập bình luận về phim.

Khi người dùng nhấn nút "Analyze Sentiment", mô hình phân loại sẽ dự đoán cảm xúc của bình luận và kết quả sẽ được hiển thị trên giao diện.

# KẾT LUẬN

Qua bài báo cáo này, chúng ta đã tìm hiểu về ba thuật toán quan trọng là A\*, Minimax và Naive Bayes Classifier, cùng với các ứng dụng của chúng trong thực tế.

Thuật toán A\* là một thuật toán tìm kiếm đường đi tối ưu trong các bài toán trò chơi và lập kế hoạch. Với ý tưởng kết hợp giữa giá trị chi phí di chuyển và giá trị ước lượng từ trạng thái hiện tại đến đích, A\* cho phép tìm ra đường đi ngắn nhất hoặc tối ưu nhất. Ưu điểm của A\* là khả năng tìm kiếm nhanh và tìm ra lời giải tối ưu, nhưng nhược điểm là yêu cầu lưu trữ nhiều thông tin và có thể phụ thuộc vào chất lượng hàm ước lượng.

Thuật toán Minimax là một phương pháp quan trọng trong lĩnh vực trò chơi và quyết định tối ưu. Với tư tưởng tìm kiếm các nước đi tiếp theo dựa trên việc dự đoán hành động của đối thủ, Minimax cho phép tìm ra chiến lược tối ưu cho người chơi trong các trò chơi hai người. Ưu điểm của Minimax là khả năng đưa ra quyết định tối ưu dựa trên các tình huống đối kháng, nhưng nhược điểm là yêu cầu tìm kiếm trên không gian trạng thái lớn và có thể gặp khó khăn khi áp dụng cho các trò chơi phức tạp.

Naive Bayes Classifier là một thuật toán phân loại dựa trên nguyên tắc xác suất. Với giả định về tính độc lập giữa các đặc trưng, Naive Bayes tính toán xác suất để một mẫu dữ liệu thuộc vào một lớp cụ thể dựa trên các đặc trưng của nó. Ưu điểm của Naive Bayes Classifier là đơn giản, dễ triển khai và xử lý tốt với dữ liệu văn bản, nhưng nhược điểm là giả định về tính độc lập có thể không đúng trong một số tình huống.

Trên cơ sở lý thuyết, chúng ta đã tìm hiểu về cách hoạt động và ưu điểm, nhược điểm của các thuật toán A\*, Minimax và Naive Bayes Classifier. Tiếp theo, chúng ta đã xem xét các ứng dụng của chúng trong thực tế, bao gồm trò chơi Maze, 8-puzzle, 2048, Connect Four và Tic-tac-toe, cũng như mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim.

Tổng kết lại, A\*, Minimax và Naive Bayes Classifier là những công cụ quan trọng trong lĩnh vực trò chơi, lập kế hoạch và phân loại dữ liệu. Các ứng dụng của chúng đa dạng và mang lại những giải pháp tối ưu và hiệu quả cho nhiều bài toán khác nhau trong thực tế.

# TÀI LIỆU THAM KHẢO

1. Game Maze: Lấy ý tưởng từ sách "Artificial Intelligence with Python" tại phần "Building a maze solver" (trang 242–246).
2. Game 8-puzzle: Trích từ sách "Artificial Intelligence with Python" tại phần "Building an 8-puzzle solver" (trang 237–241).
3. Game 2048 trích nguồn từ kênh Youtube: <https://www.youtube.com/watch?v=S73rXJtNxp8> với mã nguồn trên GitHub: <https://github.com/AlejoG10/2048-timelapse-yt>
4. Game Connect Four: Tham khảo từ sách "Artificial Intelligence with Python" tại phần “Building two bots to play Connect Four™ against each other ” (trang 320 – 323), trích nguồn từ GitHub: <https://github.com/SoundNandu/Connect-4-Reinforcement-learning> .
5. Game Tic-tac-toe: Trích từ sách "Artificial Intelligence with Python" tại phần "Building a bot to play Tic-Tac-Toe" (trang 317–319).
6. Mô hình phân loại Naive Bayes để phân loại cảm xúc của các bình luận về phim: Trích từ sách "Artificial Intelligence with Python" tại phần "Building a sentiment analyzer" (trang 370–373).