TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KÌ**

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

*Người hướng dẫn*: **Ths**. **Nguyễn Thành An**

*Người thực hiện*: **Lương Chí Trung - 52200166**

**Hà Trọng Nguyễn – 52200148**

**Đặng Nhật Toàn – 52200141**

**Đỗ Thị Kiều Thanh - 52200144**

**Hồ Thu Yến Ngọc - 52200149**

Lớp**: 22050301**

Khoá**: 26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

A red and blue logo

Description automatically generated

**BÁO CÁO CUỐI KÌ**

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

*Người hướng dẫn*: **Ths**. **Nguyễn Thành An**

*yNgười thực hiện*: **Lương Chí Trung - 52200166**

**Hà Trọng Nguyễn – 52200148**

**Đặng Nhật Toàn – 52200141**

**Đỗ Thị Kiều Thanh - 52200144**

**Hồ Thu Yến Ngọc - 52200149**

Lớp**: 22050301**

Khoá**: 26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến **ThS. Nguyễn Thành An** vì đã giúp đỡ trong quá trình làm bài báo cáo vừa qua. Nhờ sự hướng dẫn và giải đáp của thầy, nhóm đã có thể hoàn thành bài báo cáo một cách tốt nhất có thể.

Thầy đã dành thời gian và tâm huyết để hướng dẫn chúng em từng bước, giải thích những khái niệm khó hiểu, và chỉ ra những lỗi sai mà tôi mắc phải trong quá trình viết bài. Những lời khuyên và sự hỗ trợ của thầy đã giúp nhóm tự tin hơn trong việc hoàn thiện bài báo cáo của mình.

Chúng em rất biết ơn sự giúp đỡ của thầy và mong rằng sẽ có nhiều học trò khác được thầy hướng dẫn và giúp đỡ, hy vọng sẽ có cơ hội được học hỏi thêm từ thầy trong tương lai.

**BÁO CÁO ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Chúng em cam đoan rằng ***báo cáo*** này là sản phẩm của riêng nhóm và đã được hướng dẫn bởi **ThS. Nguyễn Thành An.** Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong báo cáo còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào nhóm em xin hoàn toàn chịu trách nhiệm về nội dung báo cáo của mình.** Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 19 tháng 05 năm 2024*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Lương Chí Trung*

*Hà Trọng Nguyễn*

*Đỗ Thị Kiều Thanh*

*Hồ Thu Yến Ngọc*

*Đặng Nhật Toàn*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Báo cáo này gồm **03 chương**:

**Chương 1: Bài toán 8x8 Tic-Tac-Toe**

Bài toán yêu cầu viết chương trình chơi cờ Caro 8x8 cho người dùng chơi với máy tính. Sử dụng thuật toán alpha-beta pruning để tìm nước cờ tối ưu cho máy tính. Cần thiết kế các lớp Problem, SearchStrategy, Player và Game để quản lý bài toán và điều khiển luồng trò chơi. Cần hiển thị giao diện console, xử lý đầu vào từ người chơi và kiểm tra thắng thua.

**Chương 2: Bài toán N-Queens với CNFs**

Bài toán N-Queens là bài toán đặt N quân hậu lên bàn cờ vua N x N sao cho không có hai quân nào tấn công nhau. Bài toán này có thể được giải quyết hiệu quả bằng kỹ thuật CNF (Constraint Satisfaction Networks).

Kỹ thuật CNF biểu diễn các ràng buộc của bài toán dưới dạng tập các mệnh đề logic, sau đó sử dụng thuật toán giải CNF để tìm các giải pháp cho bài toán. Ưu điểm của kỹ thuật này là dễ dàng biểu diễn các ràng buộc phức tạp, sử dụng thuật toán giải CNF hiệu quả và có thể áp dụng cho nhiều bài toán tối ưu hóa khác. Tuy nhiên, nhược điểm của nó là kích thước CNF có thể tăng nhanh với N lớn và thuật toán giải CNF có thể tốn thời gian cho các bài toán lớn.

**Chương 3: Bài toán Decision Trees**

Cây quyết định là một thuật toán học máy phổ biến được sử dụng cho cả nhiệm vụ phân loại và hồi quy. Nó được ưa chuộng bởi khả năng giải thích dễ dàng và hiệu suất mạnh mẽ.

Cấu trúc: Cây quyết định có cấu trúc giống như cây với các nhánh và lá. Mỗi nút nội bộ đại diện cho một quyết định dựa trên một thuộc tính của dữ liệu. Các nhánh đại diện cho kết quả của các quyết định đó, và các lá đại diện cho phân loại hoặc dự đoán cuối cùng.

Học tập: Cây quyết định được xây dựng bằng cách phân tích một tập dữ liệu huấn luyện. Thuật toán lặp lại để xác định thuộc tính phân biệt tốt nhất để chia dữ liệu tại mỗi nút. Mục tiêu của việc phân chia này là tạo ra các nhóm dữ liệu tinh khiết hơn tại các nhánh lá, dẫn đến dự đoán chính xác hơn.

Dự đoán: Sau khi được huấn luyện, cây quyết định có thể được sử dụng để đưa ra dự đoán cho dữ liệu mới. Bắt đầu từ nút gốc, chúng ta di chuyển xuống các nhánh dựa trên giá trị của các thuộc tính, cho đến khi đến được nhánh lá chứa kết quả dự đoán.

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc167030216)

[TÓM TẮT 4](#_Toc167030217)

[MỤC LỤC 6](#_Toc167030218)

[DANH MỤC HÌNH ẢNH 8](#_Toc167030219)

[1.1 Giới Thiệu 9](#_Toc167030220)

[1.1.1 Mô tả bài toán 9](#_Toc167030221)

[1.1.2 Yêu cầu đề ra 9](#_Toc167030222)

[1.2 Phương pháp giải 9](#_Toc167030223)

[1.2.1 Thuật toán MINMAX 9](#_Toc167030224)

[1.2.2 Thuật toán Alpha-Beta Pruning 12](#_Toc167030225)

[1.2.3 Mô hình hóa bài toán 15](#_Toc167030226)

[1.3 Thực nghiệm 23](#_Toc167030227)

[1.3.1 Một số hình ảnh minh họa 23](#_Toc167030228)

[CHƯƠNG 2: BÀI TOÁN N-QUEENS VỚI CNFs 25](#_Toc167030229)

[2.1 Giới thiệu 25](#_Toc167030230)

[2.1.1 Mô tả bài toán 25](#_Toc167030231)

[2.1.2 Yêu cầu đề ra 25](#_Toc167030232)

[2.2 Phương pháp giải 25](#_Toc167030233)

[2.2.1 Thuật toán CNF (Conjunctive Normal Form) 26](#_Toc167030234)

[2.2.2 Mô hình hóa bài toán 27](#_Toc167030235)

[2.3 Thực nghiệm 32](#_Toc167030236)

[CHƯƠNG 3: BÀI TOÁN DECISION TREES 33](#_Toc167030237)

[3.1 Giới thiệu 33](#_Toc167030238)

[3.1.1 Mô tả bài toán 33](#_Toc167030239)

[3.1.2 Yêu cầu đề ra 36](#_Toc167030240)

[TÀI LIỆU THAM KHẢO 45](#_Toc167030241)

[BẢNG TỰ ĐÁNH GIÁ 46](#_Toc167030242)

# DANH MỤC HÌNH ẢNH

[Hình 1. 1 Mô tả thuật toán Minmax 12](#_Toc167029601)

[Hình 1. 2 Mô tả thuật toán Alpha-Beta Pruning 15](#_Toc167029602)

[Hình 1. 3 Mô phỏng bài toán 23](#_Toc167029603)

[Hình 1. 4 Mô phỏng AI đánh lượt đầu tiên 24](#_Toc167029604)

[Hình 1. 5 Mô phỏng người đánh lượt đầu tiên 25](#_Toc167029605)

[Hình 1. 6 Flow chart cho N-Queens with CNFs 27](#_Toc167029606)

[Hình 1. 7 Mô hình hóa trò chơi N-Queens 33](#_Toc167029607)

[Hình 1. 8 Mô hình hóa trò chơi N-Queens 33](#_Toc167029608)

[Hình 1. 9 Đồ thị biểu diễn Entropy 36](#_Toc167029609)

[Hình 1. 10 Flow chart của mô hình thuật toán C4.5 39](#_Toc167029610)

[Hình 1. 11 Flow chart của mô hình thuật toán CART 41](#_Toc167029611)

[Hình 1. 12 Báo cáo đánh giá của model sử dụng cây quyết định với thuật toán C4.5 42](#_Toc167029612)

[Hình 1. 13 Hình ảnh trực quan của cây quyết định kết quả với thuật toán C4.5 42](#_Toc167029613)

[Hình 1. 14 Báo cáo đánh giá của model sử dụng cây quyết định với thuật toán CART 43](#_Toc167029614)

[Hình 1. 15 Hình ảnh trực quan của cây quyết định kết quả với thuật toán CART 43](#_Toc167029615)

[Hình 1. 16 Biểu đồ phân tán dữ liệu trong không gian hai chiều với đường ranh giới quyết định 44](#_Toc167029616)

[Hình 1. 17 Hình ảnh trực quan cấu trúc cây quyết định 44](#_Toc167029617)

**CHƯƠNG 1: BÀI TOÁN 8X8 TIC-TAC-TOE**

1. Giới Thiệu
2. Mô tả bài toán

Bài toán yêu cầu xây dựng một chương trình cho phép người dùng chơi Tic-Tac-Toe với máy tính trên bàn cờ 8x8. Trò chơi kết thúc khi một trong hai bên có bốn quân thẳng hàng ngang, dọc hoặc chéo. Người chơi chọn ô bằng cách nhập tọa độ từ bàn phím, và giao diện bàn cờ được cập nhật sau mỗi lượt đi. Thuật toán được sử dụng cho máy tính là Alpha-Beta Pruning.

1. Yêu cầu đề ra

* Chương trình cho phép người dùng chơi Tic-Tac-Toe với máy tính trên bàn cờ 8x8.
* Người chơi chọn ô bằng cách nhập tọa độ từ bàn phím.
* Cập nhật giao diện bàn cờ sau mỗi lượt đi.
* Máy tính sử dụng thuật toán Alpha-Beta Pruning để tìm nước đi tối ưu.
* Trò chơi hoạt động trên màn hình console.
* Tổ chức chương trình với lớp ‘Problem’ để quản lý trò chơi và lớp ‘SearchStrategy’ để triển khai thuật toán Alpha-Beta Pruning.

1. Phương pháp giải
2. Thuật toán MINMAX
3. *Mô tả thuật toán*

Minimax là một thuật toán phổ biến trong lý thuyết trò chơi, đặc biệt là trong các trò chơi đối kháng hai người như cờ vua, cờ caro, và cờ tướng. Thuật toán này nhằm tìm nước đi tốt nhất cho người chơi (Maximizer) khi đối thủ (Minimizer) cũng cố gắng chọn nước đi tốt nhất cho mình. Minimax duyệt qua tất cả các nước đi có thể, xây dựng một cây trò chơi để tính toán giá trị của từng nước đi dựa trên các trạng thái kết thúc trò chơi.­

Người chơi Maximizer cố gắng tối đa hóa giá trị trò chơi.

Người chơi Minimizer cố gắng tối thiểu hóa giá trị trò chơi.

1. *Mã giả*

|  |
| --- |
| function MINIMAX-DECISION(state) returns an action  return argmaxa ∈ ACTIONS(s)MIN-VALUE(RESULT(state, a))  function MAX-VALUE(state) returns a utility value  if TERMINAL-TEST(state) then return UTILITY(state)  v ← -∞  for each a in ACTIONS(state) do  v ← MAX(v, MIN-VALUE(RESULT(s, a)))  return v  function MIN-VALUE(state) returns a utility value  if TERMINAL-TEST(state) then return UTILITY(state)  v←∞  for each a in ACTIONS(state) do  v ← MIN(v, MAX-VALUE(RESULT(s, a)))  return v |

1. *Mô tả các bước giải*

**Bước 1**: Khởi tạo và gọi hàm Minimax

* + Thuật toán bắt đầu từ trạng thái hiện tại của trò chơi.
  + Hàm MINIMAX-DECISION được gọi, nó duyệt qua tất cả các hành động có thể thực hiện từ trạng thái hiện tại.
  + Hành động nào có giá trị MIN-VALUE lớn nhất sẽ được chọn.

**Bước 2**: MAX-VALUE và MIN-VALUE

* MAX-VALUE và MIN-VALUE là các hàm đệ quy, được gọi lần lượt từ trạng thái hiện tại.
* MAX-VALUE tính toán giá trị tốt nhất mà người chơi Maximizer có thể đạt được từ trạng thái hiện tại.
* MIN-VALUE tính toán giá trị tốt nhất mà người chơi Minimizer có thể đạt được từ trạng thái hiện tại.

**Bước 3**: Duyệt qua các trạng thái con

* Trong MAX-VALUE, thuật toán duyệt qua tất cả các hành động có thể thực hiện từ trạng thái hiện tại và gọi đệ quy hàm MIN-VALUE cho các trạng thái kết quả.
* Tương tự, trong MIN-VALUE, thuật toán duyệt qua tất cả các hành động có thể thực hiện từ trạng thái hiện tại và gọi đệ quy hàm MAX-VALUE cho các trạng thái kết quả.

**Bước 4**: Tính toán giá trị utility

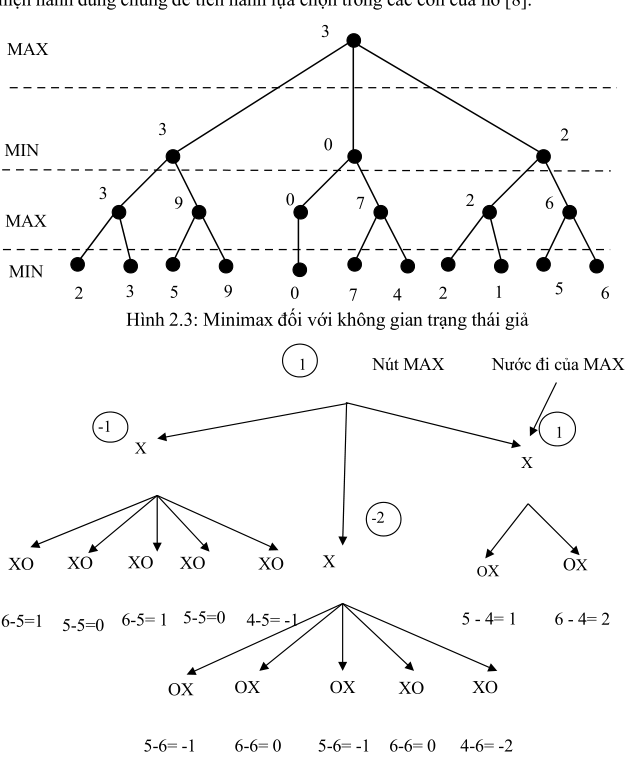
* Khi thuật toán đạt tới một trạng thái kết thúc (terminal state), nó trả về giá trị utility của trạng thái đó (do hàm UTILITY xác định).
* Giá trị utility này được sử dụng để tính toán các giá trị cho các trạng thái cha của nó.

**Bước 5:** Trả về giá trị

* MAX-VALUE trả về giá trị lớn nhất trong số các giá trị MIN-VALUE của các trạng thái con.
* MIN-VALUE trả về giá trị nhỏ nhất trong số các giá trị MAX-VALUE của các trạng thái con.

**Bước 6**: Chọn hành động tốt nhất

* Khi hàm MINIMAX-DECISION hoàn tất việc duyệt qua tất cả các hành động từ trạng thái gốc, nó chọn hành động có giá trị MIN-VALUE lớn nhất.



Hình 1. 1 Mô tả thuật toán Minmax

1. Thuật toán Alpha-Beta Pruning
2. Mô tả thuật toán

Alpha-Beta Pruning là một cải tiến của thuật toán Minimax, giúp giảm số lượng các nút được đánh giá trong cây tìm kiếm. Thuật toán sử dụng hai giá trị alpha (α) và beta (β) để theo dõi giới hạn trên và dưới của giá trị tối ưu cho người chơi và đối thủ. Thuật toán bắt đầu từ trạng thái hiện tại của trò chơi, duyệt qua từng nút trong cây tìm kiếm. Khi duyệt qua các nút con, nếu một nút con có thể chứng minh rằng giá trị của nó không thể cải thiện giá trị tối ưu hiện tại, nó sẽ được cắt bỏ và không cần phải duyệt tiếp. Thuật toán tiếp tục duyệt qua các nút con khác cho đến khi hoàn thành việc đánh giá tất cả các nút cần thiết trong cây.

1. Mã giả

|  |
| --- |
| function ALPHA-BETA-SEARCH(state) returns an action  v ← MAX-VALUE(state,-∞,+∞)  return the action in ACTIONS(state) with value v  function MAX-VALUE(state,α,β) returns a utility value  if TERMINAL-TEST(state) then return UTILITY(state)  v ← -∞  for each a in ACTIONS(state) do  v ← MAX(v, MIN-VALUE(RESULT(s,a),α,β))  if v ≥ β then return v  α ← MAX(α, v)  return v  function MIN-VALUE(state,α,β) returns a utility value  if TERMINAL-TEST(state) then return UTILITY(state)  v ← +∞  for each a in ACTIONS(state) do  v ← MIN(v, MAX-VALUE(RESULT(s,a) ,α,β))  if v ≤ α then return v  β ← MIN(β, v)  return v |

1. Mô tả các bước giải

**Bước 1**: MAX-VALUE và MIN-VALUE

* + Thuật toán bắt đầu từ nút gốc của cây trò chơi, với trạng thái hiện tại của trò chơi.
  + MAX-VALUE và MIN-VALUE lần lượt được gọi, bắt đầu từ nút gốc.
  + MAX-VALUE và MIN-VALUE là các hàm đệ quy, tính toán giá trị tốt nhất mà người chơi có thể đạt được và giá trị tệ nhất mà đối thủ có thể đạt được, tương ứng.

**Bước 2**: Lặp qua các nút con

* Trong quá trình tính toán, thuật toán lặp qua các nút con của nút hiện tại, thực hiện đệ quy để tính toán giá trị tốt nhất cho nút hiện tại.
* Lặp qua tất cả các hành động có thể thực hiện từ trạng thái hiện tại, và gọi đệ quy tương ứng với MAX-VALUE hoặc MIN-VALUE để tính toán giá trị tốt nhất hoặc tồi nhất cho các trạng thái kế tiếp.

**Bước 3**: Cắt tỉa

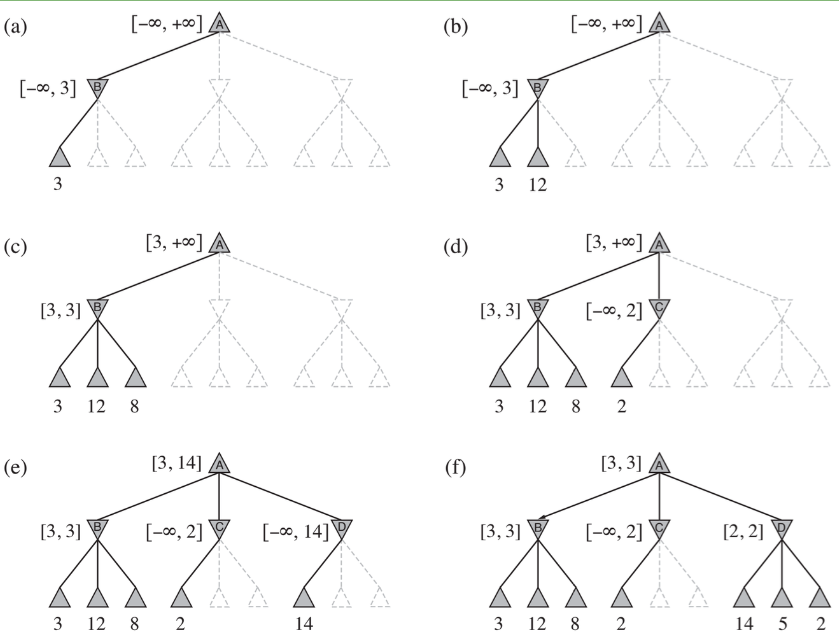
* + Trong quá trình tính toán, nếu một nút con đã cho thấy rằng giá trị của nút cha sẽ không thay đổi (ví dụ: giá trị của nút con là v ≥ β trong MAX-VALUE hoặc v ≤ α trong MIN-VALUE), thuật toán sẽ ngưng tính toán và trả về giá trị tương ứng.
  + Điều này giúp loại bỏ các nhánh không cần thiết của cây tìm kiếm, giảm độ phức tạp của thuật toán.

**Bước 4**: Trả về giá trị

* Khi hoàn thành việc tính toán cho một nút, MAX-VALUE hoặc MIN-VALUE sẽ trả về giá trị tốt nhất hoặc tồi nhất mà nút đó có thể đạt được, tương ứng.
* Giá trị này được truyền lên các nút cha và được sử dụng để quyết định hành động tốt nhất từ trạng thái hiện tại của trò chơi.

**Bước 5**: Trả về hành động tốt nhất

* Khi đã duyệt qua tất cả các nút con của nút gốc, thuật toán trả về hành động tốt nhất được tính toán từ hàm MAX-VALUE, đó là hành động mà người chơi có thể thực hiện để đạt được giá trị tốt nhất từ trạng thái hiện tại của trò chơi.



Hình 1. 2 Mô tả thuật toán Alpha-Beta Pruning

1. Mô hình hóa bài toán
2. Lớp Board

**Thuộc tính:**

* size: Kích thước của bàn cờ (size x size).
* empty: Ký tự đại diện cho ô trống trên bàn cờ.
* board: Mảng 2 chiều lưu trữ trạng thái hiện tại của bàn cờ, với kích thước là size x size.
* HEURISTIC: Mảng 2 chiều lưu giá trị heuristic cho mỗi ô trên bàn cờ, được tính dựa trên vị trí của ô đó trong bàn cờ.

**Hàm định nghĩa:**

* Khởi tạo một đối tượng Board

|  |
| --- |
| **Hàm:**  *init(self, size=8)*  **Trong đó:**   * size: kích thước size x size, nếu không có kích thước được cung cấp, mặc định là 8. |

* Tạo một mảng 2 chiều lưu giá trị heuristic cho mỗi ô trên bàn cờ. Giá trị heuristic được tính dựa trên vị trí của ô trong bàn cờ và khoảng cách đến các biên của bàn cờ.

|  |
| --- |
| **Hàm:**  *generate\_heuristic(self, size)*  **Trong đó:**   * size: Kích thước của bàn cờ (size x size). |

* Hiển thị trạng thái hiện tại của bàn cờ trên màn hình console.

|  |
| --- |
| **Hàm:**  *draw(self)* |

* Kiểm tra xem nước đi được chọn có hợp lệ hay không.

|  |
| --- |
| **Hàm:**  *is\_valid\_move(self, move)*  **Trong đó:**   * move: Tuple (x, y) đại diện cho tọa độ của nước đi được kiểm tra. |

* Kiểm tra xem nước đi được chọn có hợp lệ hay không.

|  |
| --- |
| **Hàm:**  *make\_move(self, x, y, player)*  **Trong đó:**   * x: Tọa độ x của ô cần thực hiện nước đi. * y: Tọa độ y của ô cần thực hiện nước đi. * player: Ký tự đại diện cho quân cờ của người chơi thực hiện nước đi. |

* Hủy bỏ nước đi tại ô có tọa độ (x, y) trên bàn cờ.

|  |
| --- |
| **Hàm:**  *undo\_move(self, x, y)*  **Trong đó:**   * x: Tọa độ x của ô cần hủy bỏ nước đi. * y: Tọa độ y của ô cần hủy bỏ nước đi. |

* Lấy tất cả các dòng, cột và đường chéo trên bàn cờ.

|  |
| --- |
| **Hàm:**  *get\_all\_lines(self)* |

* Kiểm tra xem bàn cờ đã đầy hay chưa.

|  |
| --- |
| **Hàm:**  *is\_full(self)* |

1. Lớp Problem

**Thuộc tính:**

* board: Đối tượng Board đại diện cho bàn cờ, được truyền vào khi khởi tạo lớp Problem.
* human\_player: Ký tự thể hiện quân cờ của người chơi, mặc định là "X", được truyền vào khi khởi tạo lớp Problem.
* ai\_player: Ký tự thể hiện quân cờ của máy, mặc định là “O”, được truyền vào khi khởi tạo lớp Problem.
* opponent\_factor: Hệ số để điều chỉnh độ khó của AI, mặc định là 1.05.
* current\_player: Ký tự thể hiện quân cờ của người chơi hiện tại.
* UTILITY: Dictionary chứa các giá trị đánh giá cho các mẫu trên bàn cờ. Sử dụng các ký tự để biểu diễn các mẫu (patterns) trên bàn cờ:

'x': Đại diện cho một quân cờ của người chơi.

'b': Đại diện cho một vị trí bị chặn.

'e': Đại diện cho một ô trống trên bàn cờ.

**Hàm định nghĩa:**

* Khởi tạo trạng thái game với bàn cờ.

|  |
| --- |
| **Hàm:**  *\_\_init\_\_(self, size=8, human\_player="X", opponent\_factor=1.05)*  **Trong đó:**   * opponent\_factor: tham số giúp điều chỉnh giá trị đánh giá của AI, làm cho AI cẩn trọng hơn đối với các nước đi của đối thủ. Hệ số này giúp tăng độ chính xác của các quyết định chiến lược trong trò chơi, đảm bảo rằng các mối đe dọa từ đối thủ được đánh giá đúng mức độ quan trọng của chúng. Mặc định là 1.05.   Hệ số 1.05 có nghĩa là xem xét các nước đi của đối thủ khó hơn 5% so với nước đi của người chơi hiện tại.  Giả sử chúng ta có:   * Điểm heuristic cho người chơi hiện tại: 1000 * Điểm heuristic cho đối thủ: 800 * opponent\_factor: 1.05   Giá trị heuristic cuối cùng sẽ là:  1000 − (1.05 × 800) = 1000 – 840 = 160  Điều này cho thấy rằng, với opponent\_factor là 1.05, giá trị heuristic cuối cùng giảm xuống, phản ánh rằng việc đối thủ có các nước đi mạnh hơn cần được xem xét kỹ lưỡng hơn.   * human\_player: là người chơi, có thể là người hoặc máy tính, mặc định người chơi hiện tại là người. |

* Đổi lượt chơi giữa người chơi là người và người chơi là máy tính (AI).

|  |
| --- |
| **Hàm:**  *switch\_player(self)* |

* Kiểm tra xem người chơi có chiến thắng hay không bằng cách xem xét tất cả các dòng, cột và đường chéo trên bàn cờ.

|  |
| --- |
| **Hàm:**  *check\_winner(self)* |

* Kiểm tra xem game đã kết thúc chưa bằng cách xem xét nếu có người chiến thắng hoặc bàn cờ đầy.

|  |
| --- |
| **Hàm:**  *is\_game\_over(self)* |

* Lấy tất cả các nước đi hợp lệ, tức là các ô trống trên bàn cờ.

|  |
| --- |
| **Hàm:**  *get\_valid\_moves(self)* |

* Đánh giá trạng thái game hiện tại dựa trên người chơi là máy tính.

|  |
| --- |
| **Hàm:**  *evaluate(self)* |

* Đếm số lần xuất hiện của một mẫu (pattern) trong các dòng (lines).

|  |
| --- |
| **Hàm:**  *count\_patterns\_in\_lines(self, lines, pattern)*  **Trong đó:**   * lines: Danh sách các dòng (lines) trên bàn cờ, mỗi dòng là một chuỗi ký tự đại diện cho các ô trên bàn cờ. * pattern: Chuỗi ký tự đại diện cho mẫu (pattern) cần tìm kiếm trong các dòng. |

* Tạo ra tất cả các dòng, cột, và đường chéo trên bàn cờ, chuyển đổi các ký tự để phù hợp với các mẫu (pattern) đã định nghĩa.

|  |
| --- |
| **Hàm:**  *generate\_lines(self, matrix, player)*  **Trong đó:**   * matrix: Mảng 2 chiều đại diện cho bàn cờ, mỗi phần tử trong mảng là ký tự đại diện cho trạng thái của ô. * player: Ký tự đại diện cho quân cờ của người chơi hiện tại. |

* Tính giá trị heuristic của trạng thái game dựa trên các mẫu (pattern) và vị trí của các quân cờ. So sánh điểm của người chơi và đối thủ để đưa ra một điểm số tổng hợp. Giá trị heuristic này giúp đánh giá mức độ thuận lợi của trạng thái game cho người chơi hiện tại hoặc đối thủ.

|  |
| --- |
| **Hàm:**  *calculate\_heuristic(self, board, player)*  **Trong đó:**   * board: Đối tượng Board đại diện cho bàn cờ. * player: Ký tự đại diện cho quân cờ của người chơi hiện tại.   **Hàm này dùng 2 hàm con:**   * Tính tổng giá trị heuristic dựa trên các mẫu (patterns) trong các dòng (lines) đã tạo.   *get\_sequence\_score(lines)*   * Trong đó:   lines: Danh sách các dòng, mỗi dòng là một chuỗi ký tự đại diện cho các ô trên bàn cờ.   * Tính giá trị heuristic dựa trên vị trí của các quân cờ của một người chơi trên bàn cờ.   *get\_position\_score(board, player)*   * Trong đó:   board: Mảng 2 chiều đại diện cho bàn cờ.  player: Ký tự đại diện cho quân cờ của người chơi hiện tại. |

* Đánh giá một nước đi cụ thể bằng cách tính toán giá trị heuristic sau khi thực hiện nước đi đó.

|  |
| --- |
| **Hàm:**  *evaluate\_move(self, move)*  **Trong đó:**   * move: Tuple (x, y) đại diện cho tọa độ của nước đi được đánh giá. |

* Sắp xếp các nước đi hợp lệ theo thứ tự gần tâm bàn cờ, ưu tiên các nước đi gần tâm hơn.

|  |
| --- |
| **Hàm:**  *sort\_moves(self)* |

1. Lớp SearchStrategy

**Hàm định nghĩa:**

* Đây là hàm chính của lớp SearchStrategy, thực hiện việc áp dụng thuật toán Alpha-beta Pruning để tìm nước đi tốt nhất cho máy tính.

|  |
| --- |
| **Hàm:**  *alpha\_beta\_search(self, problem)*  **Trong đó:**   * problem: là một tham số truyền vào.   **Hàm bao gồm hai hàm con:**   * Hàm đại diện cho lượt của máy tính. Nó tìm kiếm giá trị lớn nhất có thể đạt được trong các trạng thái tiếp theo của trò chơi.   *max\_value( game, alpha, beta, depth)*   * Hàm đại diện cho lượt của người chơi. Nó tìm kiếm giá trị nhỏ nhất có thể đạt được trong các trạng thái tiếp theo của trò chơi.   *min\_value( game, alpha, beta, depth)*   * Trong đó   alpha: Giá trị alpha trong thuật toán Alpha-beta Pruning.  beta: Giá trị beta trong thuật toán Alpha-beta Pruning.  depth: Độ sâu hiện tại trong việc tìm kiếm cây trò chơi. |

1. Lớp Game

**Thuộc tính:**

* board: Đối tượng Board quản lý trạng thái của bàn cờ trong trò chơi.
* problem: Đối tượng Problem quản lý trạng thái của trò chơi và các hàm liên quan.
* strategy: Đối tượng SearchStrategy thực hiện việc tìm kiếm nước đi tốt nhất cho máy tính.
* ai\_starts: Biến boolean xác định xem máy tính (AI) có bắt đầu trước không.

**Hàm định nghĩa:**

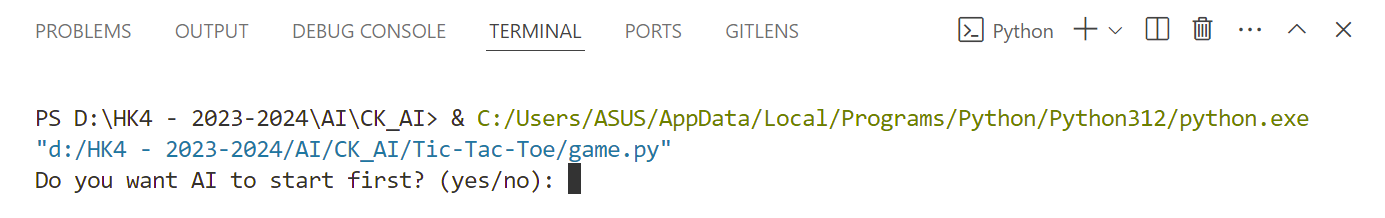
* Khởi tạo một đối tượng Game

|  |
| --- |
| **Hàm:**  *init(self, ai\_starts=False, size=8)*  **Trong đó:**   * ai\_starts: xác định liệu AI có bắt đầu trước không, mặc định là False. * size: kích thước của bàn cờ, mặc định là 8. |

* Bắt đầu trò chơi và tiến hành lần lượt các lượt cho người chơi và máy tính cho đến khi trò chơi kết thúc. Sau đó, hiển thị kết quả trò chơi (thắng, thua hoặc hòa) trên màn hình console.

|  |
| --- |
| **Hàm:**  *play(self)* |

1. Thực nghiệm
2. Một số hình ảnh minh họa



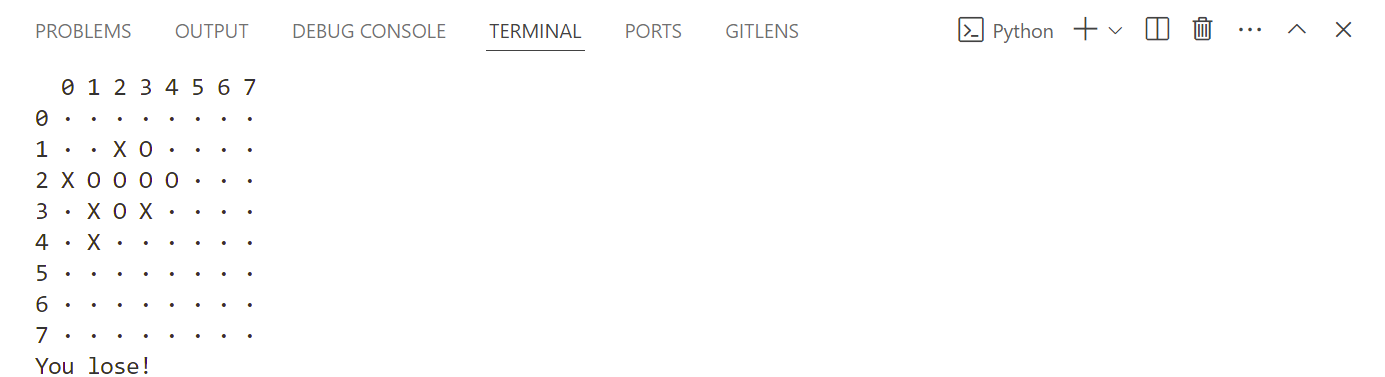
Hình 1. 3 Mô phỏng bài toán

A close up of a computer screen

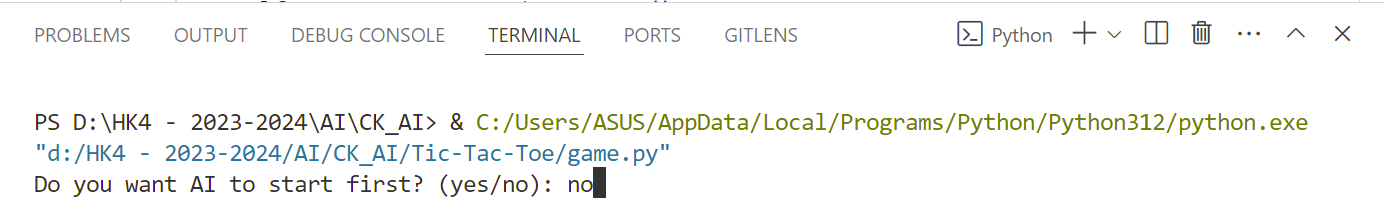
Description automatically generated

A white background with black text

Description automatically generated



Hình 1. 4 Mô phỏng AI đánh lượt đầu tiên



A white background with black text

Description automatically generated

Hình 1. 5 Mô phỏng người đánh lượt đầu tiên

# CHƯƠNG 2: BÀI TOÁN N-QUEENS VỚI CNFs

1. **Giới thiệu**
2. **Mô tả bài toán**

Bài toán N-Queens là một bài toán kinh điển trong lĩnh vực Trí tuệ nhân tạo. Đặt N quân hậu trên bàn cờ NxN sao cho không có hai quân hậu nào có thể tấn công lẫn nhau.

Bài toán N-Queens với CNFs (Conjunctive Normal Form) liên quan đến việc giải bài toán N-Queens thông qua logic mệnh đề và thuật toán SAT biểu diễn bài toán dưới dạng CNFs. Thuật toán SAT nhận vào một công thức logic mệnh đề dưới dạng CNF (Conjunctive Normal Form), tức là một hợp (AND) của các mệnh đề, trong đó mỗi mệnh đề là một tuyển (OR) của các biến hoặc phủ định của biến.

1. **Yêu cầu đề ra**

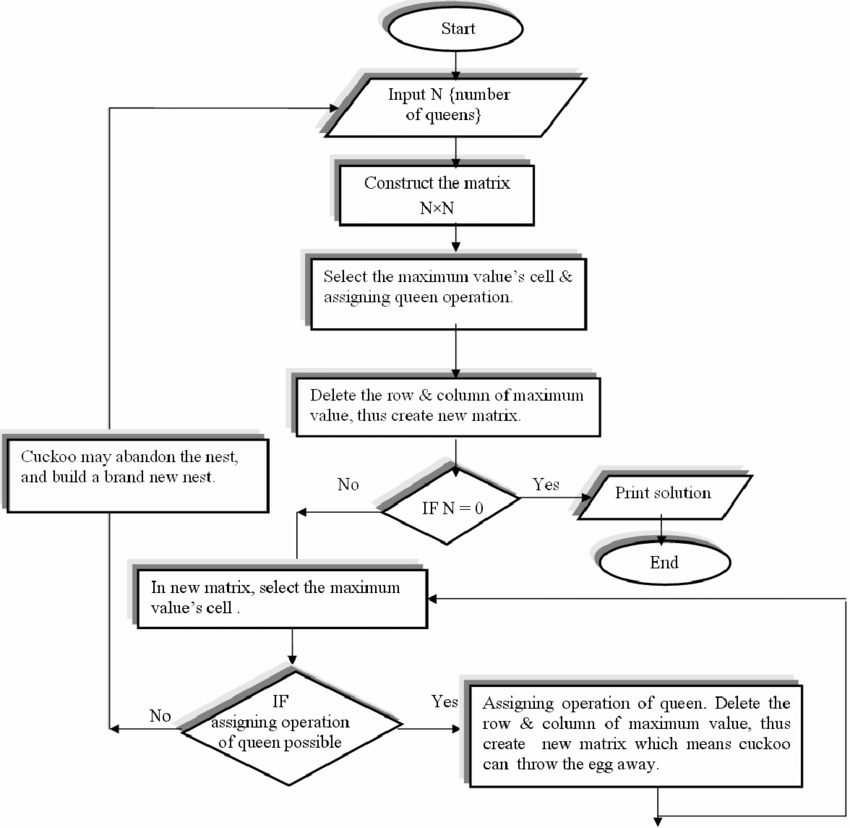
Viết chương trình đặt N quân hậu lên bàn cờ kích thước N x N với N ≥ 4, đảm bảo không có hai quân hậu nào ăn nhau.

* Nhập số N từ người dùng: nhập vào số nguyên dương N ≥ 4.
* Khởi tạo biến và phát sinh ràng buộc: mỗi ô trên bàn cờ được gán một số nguyên dương đại diện cho các biến logic. Nếu biến có giá trị True thì ô có quân hậu, nếu False thì ô trống.
* Xác định và biễu diễn ràng buộc giữa cố biến (ô) bằng logic mệnh đề.
* Chuyển đổi các mệnh đề ở bước trên thành CNFs.
* Sử dụng thư viện Glucose3 để tìm bộ giá trị cho các biến và từ đó suy ra đáp án của bài toán.
* Xuất kết quả:
* Nếu tìm được giải pháp, vẽ bàn cờ kết quả lên màn hình console.
* Nếu không tìm được giải pháp, in ra dòng chữ "Không có đáp án".

1. **Phương pháp giải**
2. **Thuật toán CNF (Conjunctive Normal Form)**
3. *Mô tả thuật toán*

Thuật toán nhận vào một công thức logic mệnh đề dưới dạng CNFs.Thuật toán SAT sẽ tìm một giá trị gán cho các biến sao cho công thức trở nên đúng.Nếu không tồn tại giá trị gán nào làm cho công thức đúng, thuật toán SAT sẽ thông báo rằng công thức không thỏa mãn.

1. *Mã giả*



Hình 1. 6 Flow chart cho N-Queens with CNFs

1. *Mô tả các bước giải*

**Bước 1:** Bắt đầu

* Khởi tạo quy trình với số lượng N quân hậu.

**Bước 2:** Xây dựng ma trận NxN

* Tạo một ma trận NxN để đặt các quân hậu.

**Bước 3:** Chọn Ô có Giá Trị Lớn Nhất và Gán Quân Hậu

* Tìm ô có giá trị lớn nhất trong ma trận và gán một quân hậu vào đó.

**Bước 4:** Xóa Hàng và Cột của Ô Đã Chọn

* Sau khi gán quân hậu, xóa hàng và cột của ô đó khỏi ma trận để tạo ra một ma trận mới.

**Bước 5:** Kiểm Tra Điều Kiện N = 0

* Nếu N bằng 0, tức là tất cả các quân hậu đã được gán, in ra giải pháp và kết thúc quy trình.

**Bước 6:** Tiếp Tục với Ma trận Mới

* Nếu N khác 0, tiếp tục quy trình với ma trận mới và lặp lại các bước từ bước 3.

1. **Mô hình hóa bài toán**
2. *Lớp Problem*

**Thuộc tính:**

* ‘side’: kích thước của bàn cờ (N x N), mặc định 4 (có thể được thay đổi khi khởi tạo đối tượng).
* ‘is\_possible\_solution’: cờ báo hiệu xem có tồn tại lời giải hợp lệ cho bài toán N-Queens hay không(mặc định False)
* ‘clauses’: là các mệnh đề CNF biểu diễn các ràng buộc của bài toán N-Queens (mặc định là kết quả của phương thức create\_clauses được gọi trong hàm khởi tạo).
* ‘board’: là bàn cờ N x N, mỗi ô là một giá trị False (trống) hoặc True (có quân hậu), mặc định là tất cả các ô đều là False.

**Hàm định nghĩa**:

* Khởi tạo một đối tượng Problem với kích thước bàn cờ là side x side

|  |
| --- |
| **Hàm:**  \_\_init\_\_(self, side=4)  **Trong đó:**   * Kích thước của bàn cờ. Mặc định là 4 nếu không được cung cấp.   **Hoạt động:**   * Khởi tạo kích thước bàn cờ với giá trị side. * Gọi phương thức create\_clauses để tạo ra các mệnh đề CNF. * Tạo một bàn cờ trống với kích thước side x side, sử dụng ký hiệu . để biểu thị các ô trống. * Đặt thuộc tính is\_possible\_solution thành False, biểu thị rằng chưa có giải pháp nào được tìm thấy. |

* Trả về danh sách các mệnh đề CNF (clauses) của bài toán.

|  |
| --- |
| **Hàm:**  *get\_clauses(self)*  **Hoạt động:**   * Trả về danh sách các mệnh đề CNF đã được tạo ra bởi phương thức create\_clauses. |

* Tạo ra các mệnh đề CNF biểu diễn ràng buộc của bài toán N-Queens.

|  |
| --- |
| **Hàm:**  *create\_clauses(self)*  **Hoạt động:**   * Tạo ra các mệnh đề CNF đảm bảo rằng:   Mỗi hàng trên bàn cờ phải có ít nhất và nhiều nhất một quân hậu.  Mỗi cột trên bàn cờ phải có ít nhất và nhiều nhất một quân hậu.  Không có hai quân hậu nào nằm trên cùng một đường chéo.   * Trả về danh sách các mệnh đề CNF. |

* Cập nhật trạng thái của bàn cờ với giải pháp (solution) nhận được.

|  |
| --- |
| **Hàm:**  *add\_solution(self, solution)*  **Trong đó:**   * solution (list): Danh sách các vị trí quân hậu trên bàn cờ (các chỉ số hàng và cột).   **Hoạt động:**   * Cập nhật bàn cờ với vị trí của các quân hậu theo giải pháp nhận được. * Đặt thuộc tính is\_possible\_solution thành True, biểu thị rằng đã tìm thấy một giải pháp hợp lệ. |

* Vẽ bàn cờ lên màn hình console.

|  |
| --- |
| **Hàm:**  *draw\_board(self)*  **Hoạt động:**   * Kiểm tra thuộc tính is\_possible\_solution:   Nếu False, in ra thông báo "No possible solution".  Nếu True, vẽ bàn cờ với ký hiệu Q biểu thị các quân hậu và ký hiệu “.” biểu thị các ô trống. |

1. *Lớp CNF*

**Thuộc tính:**

* ‘solver’: thuộc loại đối tượng của lớp Glucose3, là bộ giải SAT sử dụng thư viện Glucose3 để giải quyết các mệnh đề CNF (mặc định được khởi tạo bằng Glucose3() khi tạo đối tượng CNF).

**Hàm định nghĩa:**

* Khởi tạo đối tượng CNF và tạo một bộ giải SAT sử dụng Glucose3.

|  |
| --- |
| **Hàm:**  *\_\_init\_\_(self)*  **Hoạt động:**   * Tạo một đối tượng để lưu trữ các mệnh đề CNF. * Khởi tạo một bộ giải SAT sử dụng thư viện Glucose3, chuẩn bị cho quá trình giải quyết bài toán SAT. |

* Thêm các mệnh đề (clauses) vào bộ giải SAT.

|  |
| --- |
| **Hàm:**  *add\_clauses(self, clauses)*  **Trong đó:**   * Clauses: Mỗi mệnh đề là một danh sách các số nguyên.   **Hoạt động:**   * Thêm từng mệnh đề vào bộ giải SAT Glucose3. Các biến hoặc phủ định của chúng trong mệnh đề được biểu diễn bằng các số nguyên. |

* Giải bài toán SAT với các mệnh đề đã thêm vào và trả về một mô hình (model) thỏa mãn các mệnh đề đó. Nếu không tìm thấy giải pháp, trả về None.

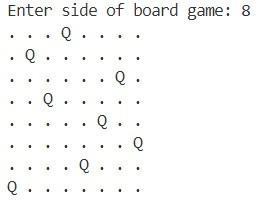
|  |
| --- |
| **Hàm:**  *get\_solution(self)*  **Trong đó:**   * Clauses: Mỗi mệnh đề là một danh sách các số nguyên.   **Hoạt động:**   * Sử dụng bộ giải SAT Glucose3 để cố gắng tìm một mô hình thỏa mãn tất cả các mệnh đề đã thêm vào. * Nếu bộ giải tìm thấy một giải pháp: Trả về mô hình (model), là một danh sách các giá trị của các biến (biến dương hoặc phủ định của chúng). * Nếu bộ giải không tìm thấy giải pháp: Trả về None. |

1. *Lớp game*

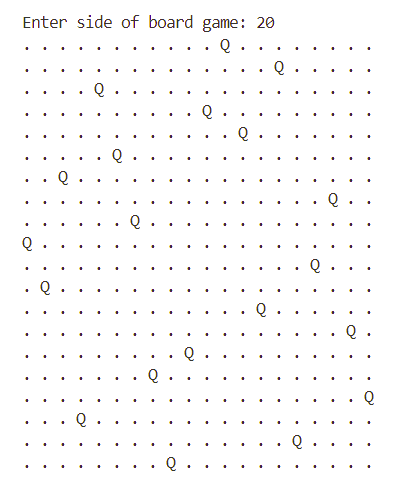
**Thuộc tính:**

* ‘N’: là kích thước của bàn cờ (N x N), được người dùng nhập vào từ bàn phím(số nguyên dương lớn hơn hoặc bằng 4).
* ‘p’: là đối tượng của lớp Problem, đối tượng đại diện cho bài toán N-Queens với kích thước bàn cờ là N.
* ‘cnf’: đối tượng của lớp CNF, đại diện cho bộ giải SAT sử dụng thư viện Glucose3 để giải quyết các mệnh đề CNF.
* ‘clauses’: list của list các int, là các mệnh đề CNF biểu diễn các ràng buộc của bài toán N-Queens. Được lấy từ đối tượng Problem thông qua phương thức get\_clauses().
* ‘solution’: list các int hoặc None, là một mô hình thỏa mãn các mệnh đề CNF nếu tìm thấy. Mỗi phần tử dương trong danh sách biểu diễn một biến logic đúng (quân hậu có mặt trên bàn cờ). Nếu không tìm thấy giải pháp, solution sẽ là None.

1. **Thực nghiệm**

****

Hình 1. 7 Mô hình hóa trò chơi N-Queens



Hình 1. 8 Mô hình hóa trò chơi N-Queens

# CHƯƠNG 3: BÀI TOÁN DECISION TREES

1. **Giới thiệu**

Decision Trees là một phương pháp học giám sát không tham số được sử dụng cho cả phân loại (classification) và hồi quy (regression).

Mục tiêu: tạo ra một mô hình dùng để dự đoán giá trị hoặc phân loại biến mục tiêu bằng cách học các quy tắc quyết định đơn giản từ các đặc trưng của dữ liệu.

Ưu điểm: dễ hiểu và giải thích, có thể hình dung, yêu cầu ít chuẩn bị dữ liệu, có thể xử lý dữ liệu số và phân loại, hỗ trợ vấn đề đa đầu ra, sử dụng mô hình hộp trắng, có thể xác nhận mô hình bằng kiểm định thống kê, hoạt động tốt ngay cả khi giả định bị vi phạm.

Nhược điểm: có thể tạo ra cây quá phức tạp không tổng quát hóa tốt dữ liệu, cũng có thể không ổn định, dự đoán không mượt hoặc liên tục, vấn đề NP-complete( tìm cây tối ưu là một bài toán phức tạp), có thể tạo ra cây thiên vị nếu dữ liệu không cân bằng.

Trong giới hạn của bài báo cáo này, chúng em sẽ sử dụng Decision Trees cho bài toán phân loại.

1. **Mô tả bài toán**

Bài toán đặt ra yêu cầu chúng ta làm việc với một tập dữ liệu có tên dt\_data.csv chứa các cột dữ liệu như sau:

* #: chỉ số dòng.
* Rank: xếp hạng.
* Q1 – Q9: 9 thuộc tính điểm số.

Mục tiêu của bài toán là xây dựng và đánh giá mô hình cây quyết định (Decision Tree) sử dụng tập dữ liệu này. Để đạt được mục tiêu đó, bài toán yêu cầu thực hiện các bước sau:

* Tính toán các giá trị Entropy (H), Average Entropy (AE) và Information Gain (IG) cho một thuộc tính điểm số được nhập từ người dùng.
* Cài đặt, huấn luyện và đánh giá mô hình Decision Tree với tập dữ liệu được cho.
* Trực quan hóa cấu trúc cây quyết định để hiển thị kết quả.

1. *Entropy*

Entropy là thước đo mức độ rối loạn hoặc tạp chất trong tập dữ liệu đã cho. Trong bối cảnh của một biến ngẫu nhiên rời rạc, entropy cung cấp một cách định lượng để đánh giá sự hỗn loạn hoặc không chắc chắn liên quan đến phân phối xác suất của biến đó.

Để hiểu rõ hơn, hãy xem xét một biến ngẫu nhiên rời rạc 𝑥 có thể nhận một trong 𝑛 giá trị khác nhau {𝑥1,𝑥2,...,𝑥𝑛}. Giả sử rằng xác suất để 𝑥 nhận các giá trị này được cho bởi {𝑝1, 𝑝2, ..., 𝑝𝑛} với 𝑝𝑖 = 𝑝 (𝑥 = 𝑥𝑖) và các điều kiện sau đây được thoả mãn:

* 0 ≤ *pi* ​≤1 cho mọi 𝑖

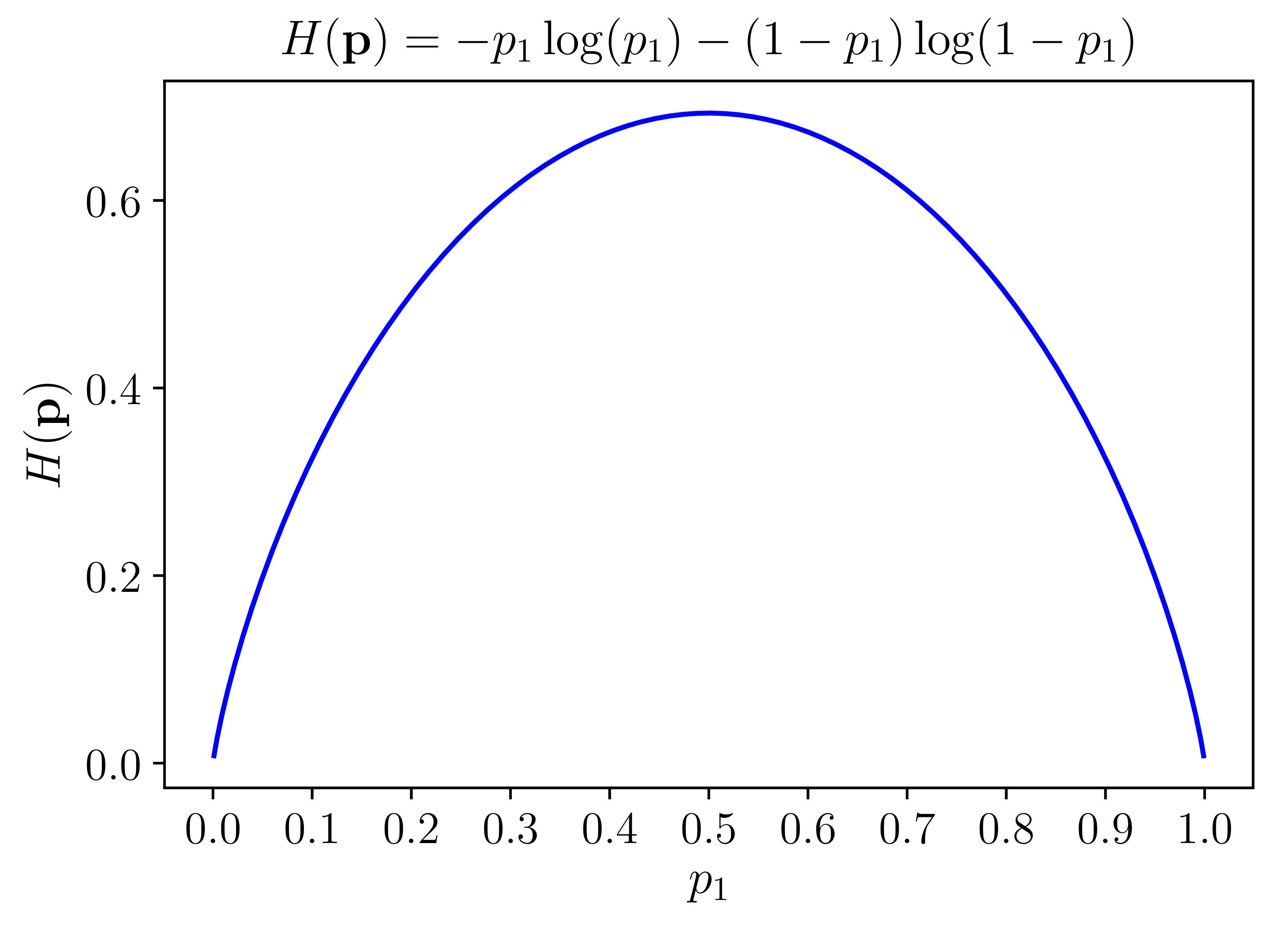
Entropy của phân phối này được định nghĩa bằng công thức sau:

Trong công thức này:

* 𝐻(𝑝) là entropy của phân phối 𝑝.
* 𝑝𝑖 là xác suất của giá trị 𝑥𝑖​.
* ​ là logarit cơ số 2 của 𝑝𝑖.

Entropy đạt giá trị tối đa khi các xác suất 𝑝𝑖​ đều nhau, tức là khi không có giá trị nào có xác suất xảy ra cao hơn giá trị khác, cho thấy mức độ không chắc chắn lớn nhất. Trong trường hợp đó, nếu 𝑥 có thể nhận 𝑛 giá trị khác nhau với xác suất đều nhau, mỗi 𝑝𝑖 = ​, entropy sẽ là:

Ngược lại, nếu một giá trị có xác suất xảy ra là 1 và tất cả các giá trị khác có xác suất là 0, entropy sẽ là 0, chỉ ra rằng không có sự không chắc chắn nào trong phân phối này.



Hình 1. 9 Đồ thị biểu diễn Entropy

1. *Information Gain (IG)*

Information Gain - IG là một khái niệm quan trọng trong lý thuyết thông tin và học máy, đặc biệt trong các thuật toán như cây quyết định (decision trees). Nó đo lường mức độ giảm bớt sự không chắc chắn (entropy) về một biến ngẫu nhiên khi biết giá trị của một biến khác. IG thường được sử dụng để chọn các thuộc tính phân chia trong việc xây dựng cây quyết định

Information Gain - IG của một thuộc tính A với tập dữ liệu 𝐷 được định nghĩa dựa trên sự giảm entropy khi biết thuộc tính 𝐴. Công thức tổng quát của Information Gain là:

𝐼𝐺(𝐷,𝐴) = 𝐻(𝐷) − 𝐻(𝐷∣𝐴)

Trong đó:

* 𝐻(𝐷) là entropy của tập dữ liệu 𝐷,
* 𝐻(𝐷∣𝐴) là entropy có điều kiện của 𝐷 khi biết thuộc tính 𝐴.

1. **Yêu cầu đề ra**
2. *Tính Entropy, Average Entropy và Information Gain*

**Bước 1**: Người dùng nhập vào tên của một thuộc tính điểm số (ví dụ: Q1).

**Bước 2**: Tính toán các giá trị sau:

* Entropy (H): Đo lường mức độ không chắc chắn hoặc hỗn loạn của một biến ngẫu nhiên.
* Average Entropy (AE): Trung bình của các giá trị Entropy của tất cả các thuộc tính điểm số (Q1 – Q9).
* Information Gain (IG): Đo lường mức độ giảm bớt của Entropy khi biết giá trị của thuộc tính điểm số đó. IG giúp xác định thuộc tính nào mang lại nhiều thông tin nhất khi phân chia tập dữ liệu.

1. *Cài đặt, huấn luyện và đánh giá mô hình Decision Tree*

**Bước 1:** Cài đặt mô hình Decision Tree sử dụng thư viện Scikit-learn.

**Bước 2:** Huấn luyện mô hình với tập dữ liệu dt\_data.csv.

**Bước 3:** Đánh giá hiệu suất của mô hình thông qua các chỉ số như độ chính xác (accuracy).

**Bước 4:** Trực quan hóa cấu trúc cây quyết định

**Bước 5:** Hiển thị cây quyết định sau khi huấn luyện để giúp hiểu rõ hơn về cách mà mô hình ra quyết định dựa trên các thuộc tính điểm số.

1. **Phương pháp giải**
2. **Thuật toán C4.5**
3. *Mô tả thuật toán*

Là một cải tiến của thuật toán ID3, Ross Quinlan đã tạo ra thuật toán cây quyết định C4.5. Trong các ứng dụng học máy và khai phá dữ liệu, đây là một phương pháp được ưa chuộng để tạo ra các cây quyết định. C4.5 giải quyết một số nhược điểm của thuật toán ID3, bao gồm khả năng xử lý các đặc tính liên tục và xu hướng quá khớp với tập dữ liệu huấn luyện.

Một sửa đổi của độ lợi thông tin được gọi là tỷ lệ độ lợi được sử dụng để giải quyết sự thiên lệch đối với các thuộc tính có nhiều giá trị. Nó được tính bằng cách chia độ lợi thông tin cho thông tin nội tại, đây là một thước đo lượng thông tin cần thiết để mô tả các giá trị của một thuộc tính.

Gain Ratio =

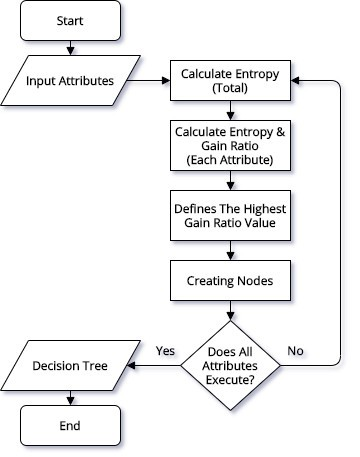
Trong đó, Split Information đại diện cho entropy của chính thuộc tính đó. Thuộc tính có tỷ lệ độ lợi cao nhất sẽ được chọn để chia tách.

Khi xử lý các thuộc tính liên tục, C4.5 sắp xếp các giá trị của thuộc tính trước, sau đó chọn điểm giữa giữa mỗi cặp giá trị liền kề làm điểm chia tách tiềm năng. Tiếp theo, nó xác định điểm chia tách nào có giá trị lớn nhất bằng cách tính toán độ lợi thông tin hoặc tỷ lệ độ lợi cho mỗi điểm.

Bằng cách biến mỗi đường đi từ gốc đến lá thành một quy tắc, C4.5 cũng có thể tạo ra các quy tắc từ cây quyết định. Các dự đoán dựa trên dữ liệu mới có thể được tạo ra bằng cách sử dụng các quy tắc này.

C4.5 là một kỹ thuật hiệu quả để tạo ra các cây quyết định có thể tạo ra các quy tắc từ cây và xử lý cả thuộc tính rời rạc và liên tục. Độ chính xác của mô hình được tăng lên và tránh được hiện tượng quá khớp nhờ vào việc sử dụng tỷ lệ độ lợi và cắt tỉa lỗi giảm dần. Tuy nhiên, nó vẫn có thể bị ảnh hưởng bởi dữ liệu nhiễu và có thể không hoạt động hiệu quả trên các bộ dữ liệu có nhiều đặc trưng.

1. *Flow chart*

**

Hình 1. 10 Flow chart của mô hình thuật toán C4.5

1. **Thuật toán CART**
2. *Mô tả thuật toán*

CART là một thuật toán cây quyết định dùng cho cả phân loại và hồi quy. Thuật toán này hoạt động bằng cách tìm các điểm chia nhỏ nhất để giảm thiểu Gini impurity, thước đo khả năng phân loại sai một phần tử được chọn ngẫu nhiên. Với phân loại, CART tính toán Gini impurity cho mỗi điểm chia và chọn điểm có impurity thấp nhất.

Gini(D) = 1-

Trong đó:

* : là tần suất tương đối của nhãn thứ i trong tập dữ liệu D.

Với nhiệm vụ hồi quy, CART tối thiểu hóa phương sai của biến mục tiêu trong mỗi tập con, sử dụng Mean Squared Error (MSE) để đánh giá các điểm chia:

MSE(D) =

Trong đó:

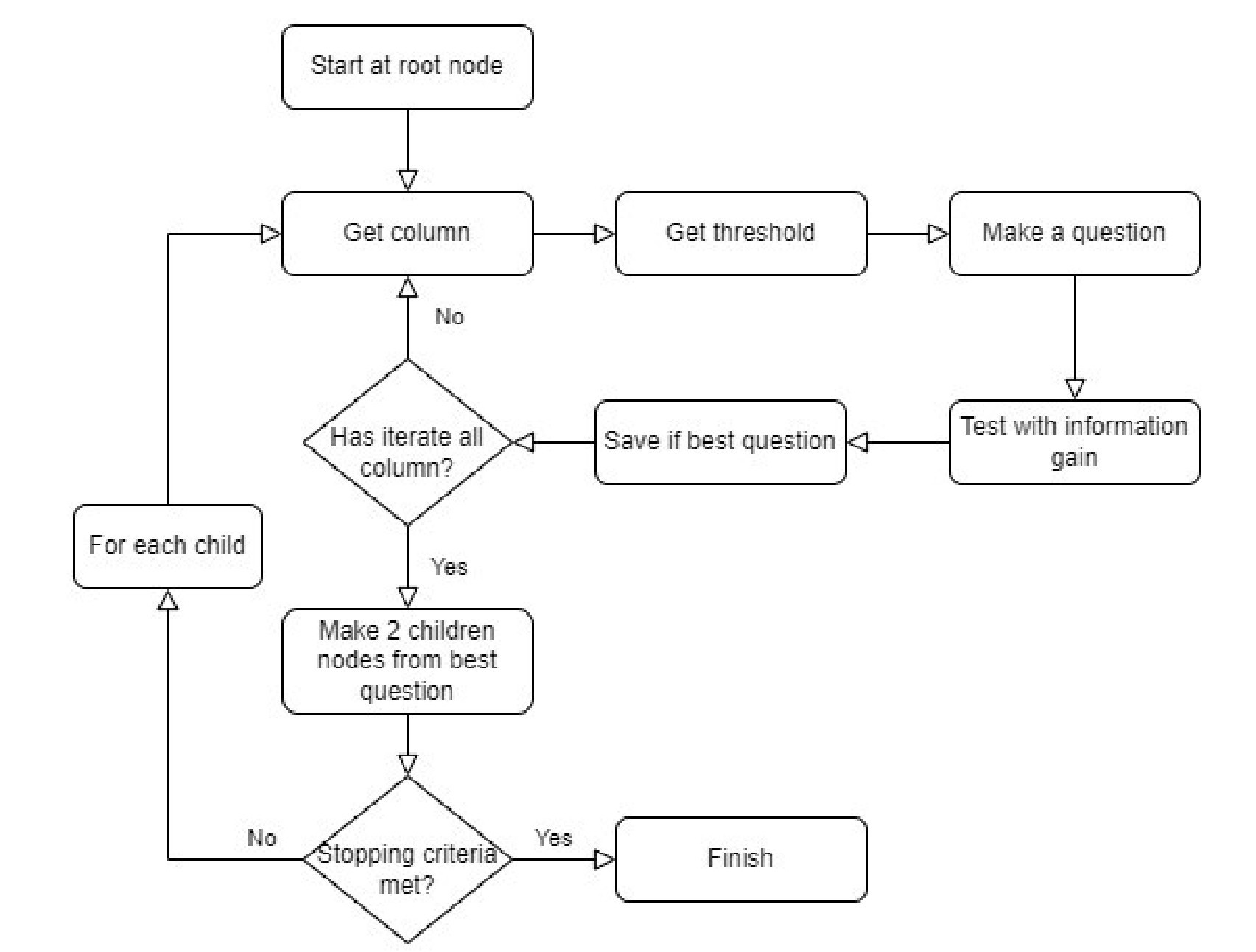
* : là giá trị mục tiêu
* : là giá trị trung bình của các giá trị mục tiêu trong tập dữ liệu D.
* : là số lượng điểm dữ liệu trong D.

CART sử dụng chiến lược tham lam (greedy) để chia nhỏ tập dữ liệu, chọn điểm chia làm giảm thiểu Gini impurity hoặc tối đa hóa độ lợi thông tin ở mỗi bước.

Cost-Complexity Pruning: để giảm thiểu hiện tượng quá khớp, CART sử dụng cắt tỉa độ phức tạp chi phí (cost-complexity pruning), tối thiểu hóa tổng chi phí bằng cách thêm tham số độ phức tạp vào thước đo độ tạp chất.

Cấu trúc của cây: mỗi nút nội bộ trong cây nhị phân của CART có chính xác hai nút con, làm đơn giản hóa quá trình chia và giải thích cây kết quả.

1. *Flow chart*



Hình 1. 11 Flow chart của mô hình thuật toán CART

1. **Thực nghiệm**

**A screenshot of a computer

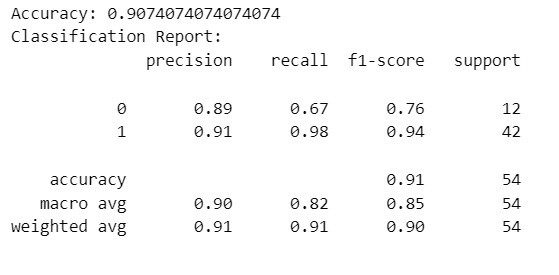
Description automatically generated**

Hình 1. 12 Báo cáo đánh giá của model sử dụng cây quyết định với thuật toán C4.5

**A diagram of a computer algorithm

Description automatically generated with medium confidence**

Hình 1. 13 Hình ảnh trực quan của cây quyết định kết quả với thuật toán C4.5

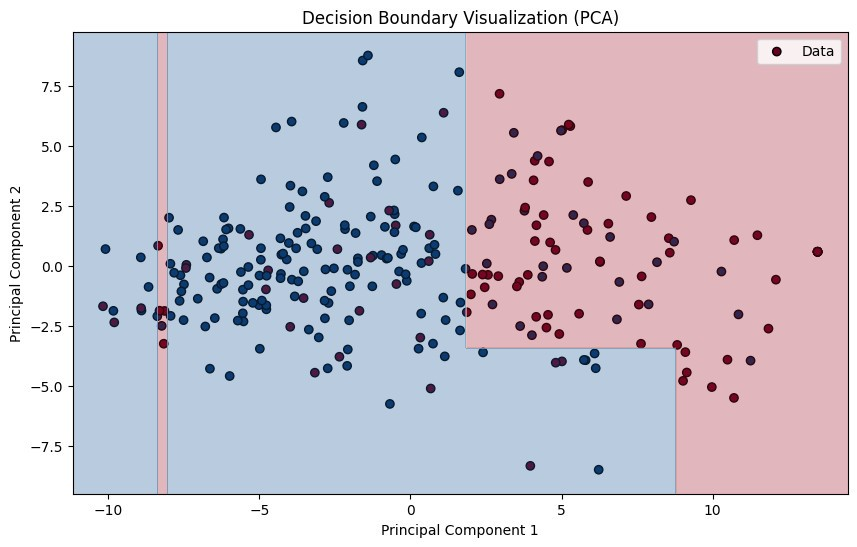


Hình 1. 14 Báo cáo đánh giá của model sử dụng cây quyết định với thuật toán CART

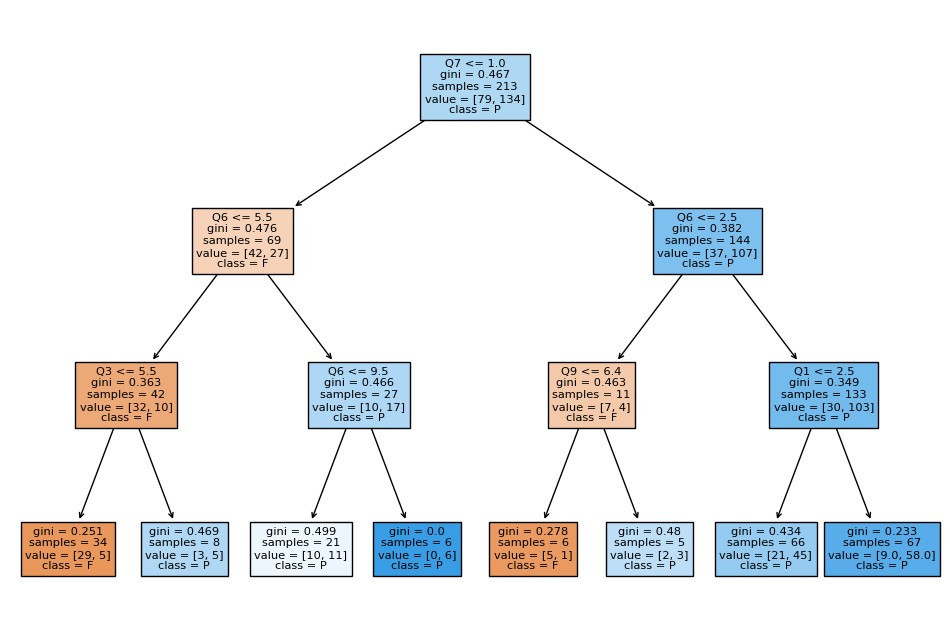
A diagram of a mathematical equation

Description automatically generated with medium confidence

Hình 1. 15 Hình ảnh trực quan của cây quyết định kết quả với thuật toán CART

****

Hình 1. 16 Biểu đồ phân tán dữ liệu trong không gian hai chiều với đường ranh giới quyết định

****

Hình 1. 17 Hình ảnh trực quan cấu trúc cây quyết định

1. **Bảng đánh giá**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Thuộc tính** | **Q1** | **Q2** | **Q3** | **Q4** | **Q5** | **Q6** | **Q7** | **Q8** | **Q9** |
| **Entropy** | 2.524 | 4.132 | 3.137 | 2.924 | 3.016 | 3.202 | 2.090 | 3.496 | 3.229 |
| **Average entropy** | 0.864 | 0.761 | 0.846 | 0.842 | 0.848 | 0.814 | 0.815 | 0.827 | 0.787 |
| **Information gain** | 0.057 | 0.160 | 0.075 | 0.097 | 0.073 | 0.107 | 0.106 | 0.094 | 0.134 |

# TÀI LIỆU THAM KHẢO

**Tiếng Việt**

1. Nguyễn, V. "Bài 34: Decision Trees (1): ID3," Machine Learning cơ bản, 14 Jan, 2018. [Online]. Available: https://machinelearningcoban.com/2018/01/14/id3/. Accessed: 17 May,2024.
2. N. V. Hòa, "Chương 4: Tìm kiếm đối kháng - trò chơi," Khoa Công nghệ Thông tin, Đại học An Giang, Bài giảng.

**Tiếng Anh**

1. GeeksforGeeks,"Decision Tree Algorithms, "GeeksforGeeks, 11 Nov, 2023. [Online]. Available: https://www.geeksforgeeks.org/decision-tree-algorithms/ ,Accessed: May 15, 2024.
2. D. B. Fogel, "Figure 1: Flow Chart for Hybrid Cuckoo Search to Solve N-queen Problem," ResearchGate, Mar. 2019. [Online]. Available: https://www.researchgate.net/figure/Figure-1-Flow-Chart-for-Hybrid-Cuckoo-Search-to-Solve-N-queen-Problem-42-Proposed\_fig1\_331585490 , Accessed: May 12,2024.
3. GeeksforGeeks, "N-Queen Problem | Backtracking-3," GeeksforGeeks, Dec. 19, 2018. [Online]. Available: https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/, Accessed: May 12, 2024.
4. S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

# BẢNG TỰ ĐÁNH GIÁ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MSSV | Họ tên | Email | Phân công công việc | Đánh giá thành viên | Đánh giá mức độ hoàn thành từng câu |
| 52200166 | Lương Chí Trung | 52200166@student.tdtu.edu.vn | Xây dựng thuật toán Decision Tree, N-Queen with CNFs | 100% | 100% |
| 52200148 | Hà Trọng Nguyễn | 52200148@student.tdtu.edu.vn | Xây dựng thuật toán Tic Tac Toe, Decision Tree | 100% | 100% |
| 52200141 | Đặng Nhật Toàn | 52200141@student.tdtu.edu.vn | N-Queen with CNs, trinh bày báo cáo | 100% | 100% |
| 52200144 | Đỗ Thị Kiều Thanh | 52200144@student.tdtu.edu.vn | Trình bày báo cáo, xây dựng thuật Toán Tic Tac Toe | 100% | 100% |
| 52200149 | Hồ Thu Yến Ngọc | 52200149@student.tdtu.edu.vn | Trình bày báo cáo, xây dựng thuật toán N-Queen with CNFs | 100% | 100% |