TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KỲ
NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Người hướng dẫn: **ThS. NGUYỄN THÀNH AN**

Người thực hiện: **LÝ TUẤN AN - 52000620**

GIẢN HOÀNG HUY - 52200147

LÝ TIỂU LONG - 52200168

LÊ HỒNG QUANG - 52200156

HUỲNH HOÀI NAM - 52200151

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

BÁO CÁO CUỐI KỲ
NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Người hướng dẫn: **ThS. NGUYỄN THÀNH AN**

Người thực hiện: **LÝ TUẤN AN - 52000620**

GIẢN HOÀNG HUY - 52200147

LÝ TIỂU LONG - 52200168

LÊ HỒNG QUANG - 52200156

HUỲNH HOÀI NAM - 52200151

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Đầu tiên, chúng tôi xin gửi lời cảm ơn chân thành đến ***Ban giám hiệu trường Đại Học Tôn Đức Thắng*** vì đã tạo điều kiện tối ưu về cơ sở vật chất và hệ thống thư viện hiện đại, đa dạng các loại tài liệu, sách báo giúp chúng tôi có thể tìm kiếm, nghiên cứu thông tin một cách thuận tiện nhất để hoàn thành bài báo cáo này.

Chúng tôi xin chân thành cảm ơn **ThS. Nguyễn Thành An**, giảng viên bộ môn *Nhập môn Trí tuệ nhân tạo*, đã truyền đạt kiến thức một cách tận tình và chi tiết, giúp chúng tôi đủ nền tảng để vận dụng vào việc viết bài báo cáo này.

BÁO CÁO ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là sản phẩm báo cáo của riêng nhóm chúng tôi và được sự hướng dẫn của **ThS. Nguyễn Thành An**. Các nội dung nghiên cứu, kết quả trong báo cáo 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 các 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 chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung bài báo cáo của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng 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)

Lý Tuấn An

Lý Tiểu Long

Giản Hoàng Huy

Lê Hồng Quang

Huỳnh Hoài Nam

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 **3 chương**:

Chương 1: 8x8 Tic-Tac-Toe

Chương 2: N-Queens with CNFs

Chương 3: Decision Tree

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

CHƯƠNG 1 – 8x8 TIC TAC TOE

1.1Vấn đề

8x8 Tic-Tac-Toe

- Cài đặt một chương trình cho phép người dùng chơi Tic-Tac-Toe với máy tính trên bàn cờ có 8 x 8 ô, trong đó bên nào có 4 quân thẳng hàng ngang hoặc dọc hoặc chéo sẽ giành chiến thắng.
- Trò chơi hoạt động trên màn hình console. Người chơi chọn ô bằng cách nhập toạ độtừ bàn phím. Sinh viên có thể cập nhật giao diện bàn cờ bằng cách xoá rồi vẽ lại sau mỗi lượt.
- Thuật toán dùng cho máy tính là alpha-beta pruning.
- Sinh viên tổ chức
  - Lớp Problem để quản lý bài toán
  - Lớp SearchStrategy với hàm alpha_beta_search(p: Problem) → action

  để máy tính ra nước cờ.
- Có thể cài đặt thêm các lớp khác đối tượng để hỗ trợ.

1.2 Cài đặt chương trình

- Môi trường:
  - Bảng Tic Tac Toe 8x8: Một bảng kích thước 8x8 được chia thành 64 ô vuông.
  - Người chơi: Hai người chơi, thường được gọi là "X" và "O", lần lượt đánh dấu ô trên bảng.
- Đặc điểm của Trò Chơi:
  - Loại Trò Chơi: Tic Tac Toe 8x8 là một trò chơi có thông tin hoàn hảo (Perfect information) và hoàn toàn quyết định (Deterministic). Mỗi tình huống trò chơi có thể được biết trước và không có yếu tố may mắn nào được tính đến.

- Hành động của Người Chơi: Cả hai người chơi lần lượt đánh dấu một ô trên bảng cho đến khi có người chiến thắng hoặc bảng đầy.
- Mục tiêu của Người Chơi: Mục tiêu của mỗi người chơi là đạt được một chuỗi liên tiếp của các ký hiệu của họ trên hàng ngang, hàng dọc hoặc đường chéo của bảng.
- Kết thúc Trò Chơi: Trò chơi kết thúc khi một trong hai người chơi đạt được một chuỗi liên tiếp 4 ký hiệu của mình, hoặc khi bảng đã đầy mà không có ai chiến thắng.
- Thông tin của Trò Chơi: Cả hai người chơi đều có thông tin đầy đủ về trạng thái hiện tại của trò chơi, bao gồm vị trí của tất cả các ký hiệu trên bảng.
- Chiến lược và Tính Toán:

- Chiến lược: Mỗi người chơi cần phải phát triển một chiến lược để cố gắng đạt được mục tiêu của mình trong mọi tình huống trò chơi.
  - Tính Toán: Mỗi nước đi của mỗi người chơi cần phải được cân nhắc kỹ lưỡng để đảm bảo rằng họ đang tiến hành nước đi tốt nhất có thể dựa trên trạng thái hiện tại của trò chơi và dự đoán về nước đi của đối thủ. Điều này thường đòi hỏi sử dụng các thuật toán tìm kiếm và đánh giá chiến lược, như thuật toán Minimax và các biến thể của nó.
  - Trong ngữ cảnh của game theory, Tic Tac Toe 8x8 có thể được xem như một trò chơi zero-sum với thông tin hoàn hảo, nơi mỗi người chơi luôn cố gắng tối đa hóa lợi ích của mình.

1.3 Mã giả

- problem.py

  class Board:

  - __init__(size): Khởi tạo đối tượng là một mảng 2 chiều có kích thước là size x size.

- __str__(): Trực quan hóa bàn cờ Tic-Tac-Toe trên màn hình console.
- winner(state): kiểm tra điều kiện thắng là 4 quân liên tiếp.

class Problem

- __init__(size = 8): Khởi tạo đối tượng là bàn cờ có kích thước 8x8 và người chơi là người và máy.
- actions(state: Board): Trả về những nước đi còn trống trên bàn cờ.
- result(action, state; Board): Trả về lỗi nếu người đánh chọn nước nằm ngoài hoặc những nước không còn trống trên bàn cờ. Hoặc trả về một bàn cờ mới với những nước mà người chơi và máy đánh.
- next_player_in_state(state: Board): Trả về quân được đánh ở nước tiếp theo.
- terminal(): Trả về True nếu trò chơi đã dừng, trả về False khi trò chơi vẫn còn tiếp tục.
- utility(state: Board): Trả về 1 khi X thắng, -1 khi O thắng, 0 khi hòa.
- make_move(action): thực hiện nước đi lên bàn cờ ứng với hành động truyền vào. Nước đi ứng với current_player
- is_valid_move(action): kiểm tra xem nước đi có hợp lệ hay không.
- switch_player(): Chuyển đổi lượt chơi giữa X và O.
- evaluate(state: Board, action): Trả về điểm số tổng quan giữa 2 người chơi X và O để tìm ra nước đi tối ưu nhất. Số càng dương thì X có lợi, số càng âm thì O có lợi.
- evaluate_player(state, current_player, action): Trả về điểm số ứng với người chơi và hành động được truyền vào.
- rank_of_place(x, y): Trả về điểm số tại các vòng của bàn cờ, càng vô trung tâm thì càng nhiều điểm hơn.

- evalute_streak_and_score(state, x, y, player, dx, dy, visited): Nhận vào vị trí x,y và hướng đi của chuỗi (ngang, dọc, chéo chính, chéo phụ) và trả về tổng số chuỗi của đối thủ từ và số điểm tương ứng.
- attack(board, x, y, player): Nhận vào vị trí x,y và trả về số điểm ở ô tương ứng nếu player truyền vào có khả năng tạo chuỗi khi đánh vào ô đó.

- search.py

  class AlphaBetaSearch

  - alpha_beta_search(p: Problem): Trả về nước đi tốt nhất để có thể thắng người chơi.
  - max_value(problem: Problem, board: Board, action, alpha, beta, depth): Trả về giá trị minimax.
  - min_value(problem: Problem, board: Board, action, alpha, beta, depth): Trả về giá trị minimax.

- test.py

  main(): Hàm để test thử trò chơi.

mã giả của AlphaBetaSearch

FUNCTION alpha_beta_search(problem) **returns** best_action

    actions ← ACTIONS(problem.board.state)

    best_action ← NONE

    alpha = -inf

    beta = +inf

    if problem.AI_player is X_SYMBOL **then** v ← -inf

    elsev ← inf

    for each action in actions

        if p.AI_player is X_SYMBOL **then**

```
            min_v ← MIN_VALUE(p, RESULT(action, GET_BOARD()),
        action, alpha, beta, depth)
            if min_v > v then v ← min_v, best_action ← action
            else max_v ← MAX_VALUE(p, RESULT(action,
GET_BOARD()), action, alpha, beta, depth)
            if max_v < v then v ← max_v, best_action ← action

FUNCTION max_value(problem, board, _action, alpha, beta, depth) returns v
    if TERMINAL(board) or depth == 0 then return EVALUTE(board, _action)
    v ← -inf
    for each action in ACTIONS(board)
        v = MAX(v, MIN_VALUE(problem, RESULT(action, board), action,
alpha, beta, depth - 1))
        if v >= beta then return v
        alpha = max(alpha, v)

FUNCTION min_value(problem, board, _action, alpha, beta, depth) returns v
    if TERMINAL(board) or depth == 0 then return EVALUTE(board, _action)
    v ← inf
    for each action in ACTIONS(board)
        v ← min(v, MAX_VALUE(problem, RESULT(action, board), action,
alpha, beta, depth - 1))
        if v <= alpha then return v
        beta = MIN(beta, v)
```

1.4 Chiến lược tính điểm cho hàm Evalute

Phòng Thủ:

- Chặn Chuỗi Liên Tiếp:
  - Mục tiêu: Ngăn chặn đối thủ tạo ra một chuỗi liên tiếp của các ký hiệu của mình trên bảng.
  - Chiến lược: Chặn Chuỗi 1, 2, 3:
    - 1: Chặn một chuỗi liên tiếp bằng cách đặt ký hiệu của mình ở một trong các ô tiếp theo để ngăn chặn đối thủ hoàn thành chuỗi.
    - 2: Chặn một chuỗi liên tiếp bằng cách đặt ký hiệu của mình ở ô giữa của chuỗi để phá vỡ nó thành hai phần.
    - 3: Chặn một chuỗi liên tiếp bằng cách đặt ký hiệu của mình ở ô cuối cùng của chuỗi để "bít" chuỗi.
- Chặn Ô ở Giữa:
  - Mục tiêu: Ngăn chặn đối thủ tạo ra một chuỗi liên tiếp bằng cách chặn ô giữa của chuỗi.
  - Chiến lược: Đặt ký hiệu của mình ở ô giữa của chuỗi của đối thủ để phá vỡ nó thành hai phần.

Tấn Công:

- Tạo Chuỗi Liên Tiếp:
  - Mục tiêu: Tạo ra một chuỗi liên tiếp của các ký hiệu của mình trên bảng để đạt được chiến thắng.
  - Chiến lược: Tạo Chuỗi 1, 2, 3:
    - 1: Tạo một chuỗi liên tiếp bằng cách đặt ký hiệu của mình ở một trong các ô tiếp theo của chuỗi.
    - 2: Tạo một chuỗi liên tiếp bằng cách đặt ký hiệu của mình ở ô giữa của chuỗi.

    - 3: Tạo một chuỗi liên tiếp bằng cách đặt ký hiệu của mình ở ô cuối cùng của chuỗi.

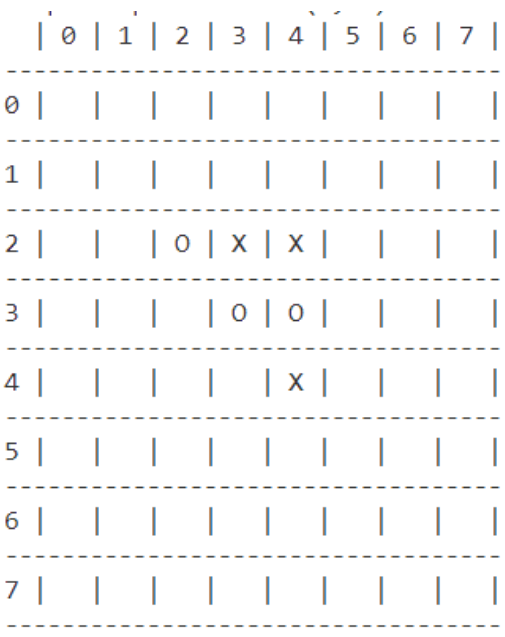
- Tạo Chuỗi Ô Giữa:
    - Mục tiêu: Tạo ra một chuỗi liên tiếp bằng cách kết nối 2 phần cạnh bên.
    - Chiến lược: Đặt ký hiệu của mình ở ô giữa của chuỗi của mình để tạo thành một chuỗi liên tiếp.

Cấp Độ Trên Bảng: Bảng 8x8 được chia nhỏ thành các cấp độ, từ cấp độ 1 đến cấp độ 4, để giúp xác định vị trí chiến lược của mỗi nước đi một cách cụ thể và rõ ràng hơn.

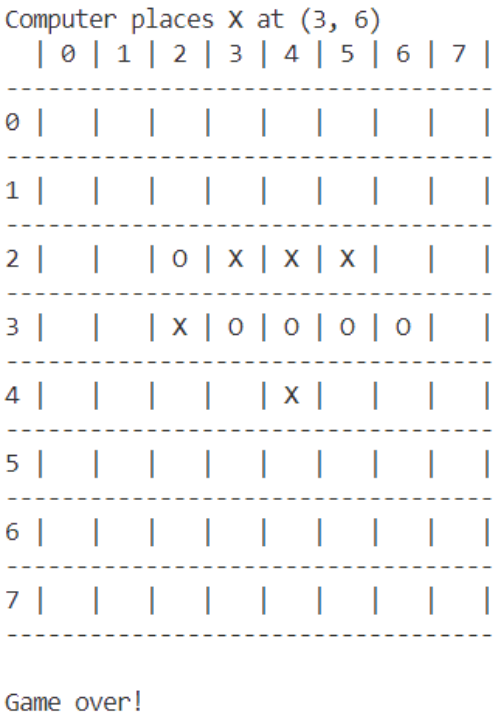
Tóm Lược:

- Trong trò chơi Tic Tac Toe 8x8, cả hai người chơi đều cần phải xem xét cả chiến lược phòng thủ và tấn công. Chiến lược phòng thủ nhằm mục tiêu ngăn chặn đối thủ tạo ra các chuỗi liên tiếp, trong khi chiến lược tấn công nhằm mục tiêu tạo ra các chuỗi liên tiếp của chính mình. Các cấp độ trên bảng được sử dụng để xác định vị trí chiến lược cụ thể cho mỗi nước đi.
- Điểm số sẽ được cộng dựa trên việc thỏa mãn điều kiện của từng chiến lược và từng cấp độ trên bảng. Nếu một điều kiện được thỏa mãn, điểm số tương ứng sẽ được cộng vào. Điều này giúp quyết định nước đi hiệu quả trong mỗi tình huống của trò chơi.

1.5 Kết quả thực hiện chương trình



Hình 1: Đánh cờ trên console.



Hình 2: Khi trò chơi kết thúc.

CHƯƠNG 2 – N-QUEEN WITH CNFs

2.1 Vấn đề

N-Queens with CNFs:

- Cài đặt chương trình đặt N quân hậu lên bàn cờ có kích thước N x N. Số $N \geq 4$ do người dùng nhập vào từ bàn phím.
- Sinh viên gán vào mỗi ô trên bàn cờ một số nguyên dương đại diện cho các biến logic nếu như biến mang giá trị True thì ô có hậu, nếu biến mang giá trị False thì ô trống.
- Xác định và biểu 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.
- Chương trình hoàn chỉnh hoạt động như sau:
    - Cho người dùng nhập vào số nguyên dương $N \geq 4$
    - Khởi tạo biến, phát sinh ràng buộc, chuyển đổi thành CNFs
    - Tìm bộ giá trị thoả mãn các mệnh đề CNFs bằng Glucose3
    - Nếu có đáp án thì vẽ bàn cờ kết quả lên màn hình console, ngược lại thì in ra
    - Dòng chữ cho biết không có đáp án.
- Gợi ý:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Bảng 1: Bảng gợi ý yêu cầu.

- Với bàn cờ 4 x 4, gán các số nguyên dương vào từng ô, mỗi số ứng với một
- biến logic
- Một ràng buộc ví dụ: "ô 1 có hậu nếu và chỉ nếu các ô 2, 3, 4 không có hậu.
- Mệnh đề: $1 \leftrightarrow \neg 2 \Lambda \neg 3 \Lambda \neg 4$. $\neg$ là phép not, Λ là phép and, $\leftrightarrow$ là kéo theo hai chiều.
- Chuyển đổi thành dạng CNFs theo quy tắc

$$p \leftrightarrow q \iff \begin{cases} p \rightarrow q \\ q \rightarrow p \end{cases} \iff \begin{cases} \neg p \vee q \\ \neg q \vee p \end{cases}$$

Hình 3: Quy tắc chuyển đổi thành dạng CNFs.

2.2 Cài đặt chương trình

class NQueenSolver:

- __init__(self): Khởi tạo 1 danh sách rỗng để lưu trữ kết quả.
- flatten(n, x, y): Hàm chuyển tọa độ x, y thành 1 chỉ số nằm trong khoảng 0 đến 63.
- buildCNF(n): Trả về danh sách các mệnh đề ở dạng CNFs.
- check_solve(n, CNF_n, visualize): Kiểm tra xem các điều kiện CNF có hợp lệ hay không.
- check_solve(n, CNF_n, visualize): Kiểm tra các mệnh đề CNF.
- solve(): kiểm tra và in ra kết quả ma trận bàn cờ.

2.3 Mô hình hóa không gian N-Queens

- Trạng thái: Một bàn cờ với kích thước NxN với N >= 4 do người dùng nhập vào từ bàn phím.
- Trạng thái đích: Đặt N quân hậu lên bàn cờ NxN với các điều kiện ràng buộc:

- Mỗi hàng chỉ có một quân hậu.
- Mỗi cột chỉ có một quân hậu.
- Không có hai quân hậu nào trên cùng một đường chéo chính.
- Không có hai quân hậu nào trên cùng một đường chéo phụ.

- Mục tiêu: Đặt N quân hậu lên bàn cờ kích thước NxN sao thỏa mãn các điều kiện ràng buộc được biểu diễn dưới dạng công thức chuẩn tắc Conjunctive Normal Form (CNF).

2.4 Mã giả

class NQueensSolver:

- __init__()
- flatten(n, x, y)
- buildCNF(n)
- check_solve(n, CNF_n, visualize)
- solve()

FUNCTION flatten(n, x, y) return a number in [1, NxN]

```
return x * n + y + 1
```

FUNCTION buildCNF(n) return list

```
CNFset_n ← a empty list
for i=1 to n do
    for j=1 to n do
        for k=j+1 to n do
            CNFset_n ← INSERT([-flatten(n, j, i), -flatten(n, k, i)])
    for j=1 to n do
        CNFset_n ← INSERT(flatten(n, j, i))
for i=1 to n do
```

```
        for j=1 to n do
            for k=j+1 to n do
                CNFset_n ← INSERT([-flatten(n, i, j), -flatten(n, i, k)])
for i=-(n-1) to n do
        for j=1 to n do
            if i+j>= 0 then
                for k=j+1 to n do
                    if i+k>=0 and i+k<n then
                        CNFset_n ← INSERT([-flatten(n, j, i+j), -
flatten(n, k, i+k)])
for i=0 to 2*n-1 do
        for j=1 to n do
            if i-j>=0 and i-j <n then
                for k=j+1 to n do
                    if i-k>=0 and i-k <n then
                        CNFset_n ← INSERT([-flatten(n, j, i-j),
                        flatten(n, k, i-k)])
```

2.5 Kết quả thực hiện chương trình

```
[['.' '.' '.' '.' 'Q' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' 'Q' '.' '.' '.']
 ['.' '.' '.' 'Q' '.' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' '.' '.' '.' 'Q']
 ['.' '.' 'Q' '.' '.' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' 'Q' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' '.' '.' 'Q' '.']
 ['.' 'Q' '.' '.' '.' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' '.' 'Q' '.' '.']
 ['Q' '.' '.' '.' '.' '.' '.' '.' '.' '.']]
```

Hình 4: Kết quả thực hiện chương trình N-Queens với N=10.

```
[['.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' 'Q' '.']
 ['.' '.' '.' 'Q' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' 'Q' '.' '.' '.' '.' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' '.' '.' '.' 'Q' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' 'Q' '.' '.']
 ['.' '.' '.' '.' '.' '.' '.' '.' '.' '.' 'Q' '.' '.' '.' '.']
 ['.' '.' 'Q' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' 'Q']
 ['.' 'Q' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' '.' '.' 'Q' '.' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' 'Q' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.']
 ['Q' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' '.' 'Q' '.' '.' '.' '.' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' '.' '.' '.' '.' '.' 'Q' '.' '.' '.']
 ['.' '.' '.' '.' '.' '.' 'Q' '.' '.' '.' '.' '.' '.' '.' '.']]
```

Hình 5: Kết quả thực hiện chương trình N-Queens với N=15.

CHƯƠNG 3 – DECISION TREE

3.1 Vấn đề

- Sinh viên tìm hiểu hai thư viện Pandas và Decision Tree (Scikit-learn) để thực hiện đồ án.
- Cho tập dữ liệu dt_data.csv chứa các cột dữ liệu như sau:
  - # → chỉ số dòng
  - Rank → xếp hạng
  - Q1 – Q9 → 9 thuộc tính điểm số
- Sinh viên thực hiện hai yêu cầu:
  - Viết chương trình cho phép người dùng nhập vào tên của một thuộc tính điểm số. Tính và in ra màn hình các giá trị Entropy (H), Average Entropy (AE) và Information Gain (IG) của thuộc tính đó.
  - Cài đặt, huấn luyện và đánh giá mô hình Decision Tree với tập dữ liệu được cho. Trực quan hoá cấu trúc cây quyết định kết quả.

3.2 Giải quyết vấn đề

3.2.1 Chương trình tính các giá trị Entropy (H), Average Entropy (AE) và Information Gain (IG)

- Cơ sở lý thuyết:
  - Entropy là thước đo độ không chắc chắn của một biến ngẫu nhiên V với các giá trị vk. Với vk là một lớp trong V.
  - Information Gain dựa trên sự giảm của hàm Entropy khi tập dữ liệu được phân chia trên một thuộc tính. Ta sử dụng thuộc tính trả về Information Gain cao nhất để làm nút gốc.
- Các công thức cần thiết cho việc giải quyết bài toán:

- Công thức tính giá trị Entropy (H):

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$$

Hình 6: Công tưhcs tính Entropy (H).

- Công thức tính giá trị Average Entropy (AE):

$$AE_{Alternate} = P(Alt = T) \times H(Alt = T) + P(Alt = F) \times H(Alt = F)$$

Hình 7: Công thức tính Average Entropy (AE).

- Công thức tính giá trị Information Gain (IG):

$$IG(Alternate, S) = H(S) - AE_{Alternate} = 1 - 1 = 0$$

Hình 8: Công thức tính giá trị Information Gain (IG).

- Cài đặt chương trình
- Lớp Calculate:
  - __init__(self, name_score): Khởi tạo đối tượng với tên thuộc tính điểm được
  - data_processing(self, name_score, data): Hàm dùng để lưu trữ dữ bộ dữ liệu cần thiết cho việc tính toán các giá trị.
  - calculate_entropy(self, quantity_score): Hàm dùng để tính giá trị entropy.
  - calculate_average_entropy(self, quantity_score, quantity_rank): Hàm được dùng để tính giá trị average entropy.
  - calculate_information_gain(self, quantity_score, quantity_rank, S): Hàm được dùng để tính toán giá trị information gain.
- Lớp main:
  - __init__(self, data): Khởi tạo đối tượng chứa bộ dữ liệu tổng
  - solve(self): Hàm dùng để giải quyết vấn đề

- Mã giả:

FUNCTION calculate_entropy(self, quantity_score): **return**s H

H ← 0

total ← sample space of quantity_score

for each value in VALUE(quantity_score):

H ← H - (value / total) * log2(value / total)

FUNCTION calculate_average_entropy(self, quantity_score, quantity_rank): **returns** AE

AE ← 0

total ← sample space of quantity_score

for each value, item in (VALUE(quantity_score), quantity_rank)

AE ← AE + (value / total) * calculate_entropy(item)

FUNCTION calculate_information_gain(self, quantity_score, quantity_rank, S): **returns** IG

IG ← calculate_entropy(S) - calculate_average_entropy(quantity_score, quantity_rank)

Kết quả thực hiện:

```
Q1
Entropy value: 2.5248962387450886
Average Entropy value: 0.8643297071195146
Information Gain value: 0.05766599597492594
```

Hình 9: Kết quả của Q1.

```
Q2
Entropy value: 4.132418142909147
Average Entropy value: 0.7616877890146895
Information Gain value: 0.1603079140797511
```

Hình 10: Kết quả của Q2.

3.2.2 Cài đặt, huấn luyện và đánh giá mô hình Decision Tree. Trực quan hóa cấu trúc cây quyết định

- Nội dung này sẽ sử dụng thư viện Decision Tree (Scikit-learn) để hỗ trợ thực hiện.
- Quá trình xử lý vấn đề
  - Thiết lập dữ liệu: Loại bỏ các dòng không cần thiết như là số dòng, thiết lập lại dữ liệu của trường rank với giá trị tương ứng.
  - Chuẩn bị dữ liệu: Tách các biến độc lập (features) là các thuộc tính điểm và biến phụ thuộc (target) là thuộc tính rank trong bộ dữ liệu. Và lưu vào các biến X_selected và y tương ứng.
  - Phân tách dữ liệu thành hai tập con: Ta sẽ sử dụng hàm train_test_split() của thư viện Scikit-learn để phân tách bộ dữ liệu thành hai tập con. Một tập sẽ được dùng để huấn luyện mộ hình (train) và một tập sẽ được dùng để đánh giá hiệu suất của mô hình (test). Dữ liệu được phân tách theo tỉ lệ 0.2 tức là 80% dữ liệu cho tập train và 20% dữ liệu cho tập test và dữ liệu sẽ được xáo trộn trước khi chia.
  - Huấn luyện mô hình: Xây dựng mô hình cây quyết định có độ sâu là 7. Sử dụng hàm fit để huấn luyện mô hình vừa tạo với dữ liệu là tập huấn luyện. Sau đó sử dụng hàm predict() để thực hiện dự đoán nhãn cho tập kiểm tra và tập huấn luyện bằng cây quyết định đã được huấn luyện trước đó.
  - Đánh giá mô hình: Đánh giá độ chính xác của mô hình đã được huấn luyện thông qua giá trị accuracy của cả tập huấn luyện và tập kiểm tra. Đồng thời đánh giá mô hình thông qua ma trận nhầm lẫn. Và thông qua các hàm được cung cấp sẵn trong thư viện ta dễ dàng có thể tính toán được giá trị accuracy và ma trận nhầm lẫn của mô hình trên.

```
Decision Tree: Accuracy on training Data: 0.9531
Decision Tree: Accuracy on test Data: 0.7222
```

Hình 11: Độ chính xác của mô hình.

```
Confusion Matrix for Training Data:
 [[ 64   8]
 [  2 139]]
Confusion Matrix for Test Data:
 [[10  8]
 [ 7 29]]
```

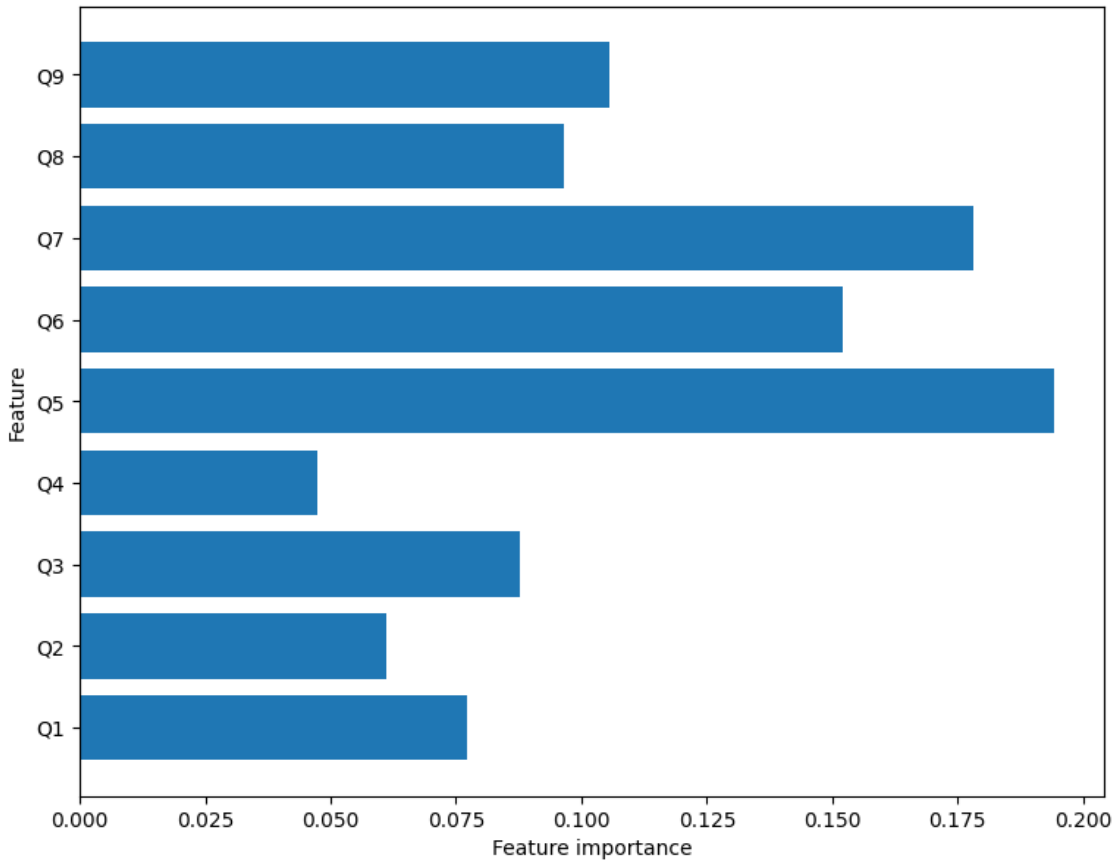
Hình 12: Ma trận nhầm lẫn của mô hình.

- Trực quan hóa dữ liệu: Vẽ biểu đồ thanh ngang để đánh giá sự quan trọng của các biến độc lập (feature) trong cây quyết định. Và vẽ cây quyết định lên màn hình.
- Thực nghiệm: Sau khi huấn luyện ta sẽ cung cấp cho mô hình một bộ dữ liệu mẫu và nhờ mô hình dự đoán kết quả thông qua bộ dữ liệu mẫu đó.

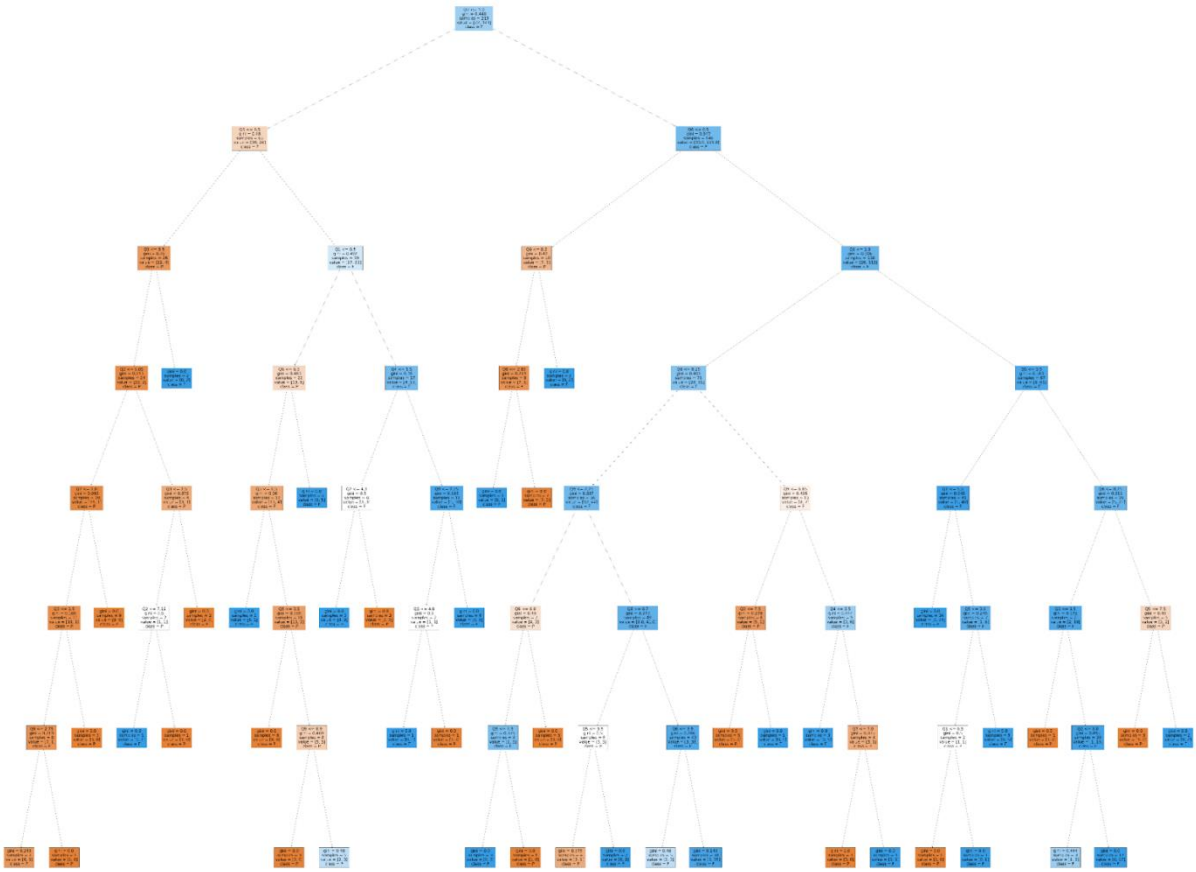
```
Input [[0, 6, 5, 0, 6, 5, 2, 7.1, 5.5]]
Result predict
P
```

Hình 13: Kết quả thực nghiệm của mô hình.

- Hình ảnh trực quan hóa:



Hình 14: Biểu đồ đánh giá sự quan trọng của các biến độc lập.



Hình 15: Decision Tree.

PHÂN CÔNG CÔNG VIỆC

Thành viên thực hiện	MSSV, Email	Phần phân công việc	Mức độ hoàn thành
Lý Tuấn An	52000620, 52000620@student.tdtu.edu.vn	Câu 1, câu 3, viết báo cáo kết quả thực hiện.	100%
Giản Hoàng Huy	52200147, 52200147@student.tdtu.edu.vn	Câu 2, viết báo cáo kết quả thực hiện.	100%
Lý Tiểu Long	52200168, 52200168@student.tdtu.edu.vn	Câu 1, viết báo cáo kết quả thực hiện.	100%
Huỳnh Hoài Nam	52200151, 52200151@student.tdtu.edu.vn	Câu 2, Viết báo cáo kết quả thực hiện.	100%
Lê Hồng Quang	52200156, 52200156@student.tdtu.edu.vn	Câu 3, viết báo cáo kết quả thực hiện.	100%

Bảng 2: Bảng phân công công việc.

THUẬN LỢI VÀ KHÓ KHĂN

Thuận lợi

- Đề bài cung cấp hướng dẫn chi tiết cho từng câu hỏi và tài liệu tham khảo cụ thể như các trang web và thư viện cần sử dụng (ví dụ: Glucose3, Pandas, Scikit-learn).
- Việc phân chia làm việc theo nhóm giúp giảm khối lượng công việc cho mỗi cá nhân, đồng thời có thể tận dụng đa dạng kỹ năng và kiến thức từ các thành viên khác nhau.
- Các công cụ và thư viện như Python, Pandas, Scikit-learn, và Glucose3 hỗ trợ mạnh mẽ cho việc xử lý dữ liệu, xây dựng mô hình và giải quyết các bài toán về trí tuệ nhân tạo.
- Có nhiều nguồn tài liệu tham khảo về các bài toán Tic-Tac-Toe, N-Queens, Decision Tree cùng với những kiến thức được giảng dạy ở trên lớp và sự đồng hành của thầy và những người bạn.

Khó khăn

- Độ phức tạp của các thuật toán như Alpha-beta pruning trong Tic-Tac-Toe và chuyển đổi các mệnh đề logic sang CNF trong N-Queens yêu cầu hiểu biết sâu về lý thuyết và kỹ năng lập trình cao.
- Phải làm quen với thư viện mới như thư viện Glucose3 hay Scikit-learn, việc học và áp dụng chúng vào bài toán có thể tốn nhiều thời gian.
- Làm việc nhóm đòi hỏi kỹ năng quản lý dự án và phân công công việc rõ ràng để đảm bảo tiến độ và chất lượng. Sự khác biệt về trình độ và khả năng giữa các thành viên có thể gây ra khó khăn trong việc hoàn thành nhiệm vụ.
- Tích hợp các mã nguồn từ các thành viên khác nhau và kiểm thử toàn bộ chương trình để đảm bảo không có lỗi là một thách thức lớn, đặc biệt với các bài toán phức tạp.

- Viết báo cáo yêu cầu khả năng diễn đạt và trình bày các kết quả phức tạp một cách cô đọng và dễ hiểu. Việc tránh nhúng mã nguồn thô cũng là một yêu cầu khó khăn đối với những ai không quen với việc viết báo cáo kỹ thuật.
- Các bài toán yêu cầu kỹ năng lập trình khá cao, từ việc quản lý các lớp và đối tượng trong Python cho đến việc cài đặt các thuật toán phức tạp, có thể gây khó khăn cho những sinh viên chưa có nhiều kinh nghiệm lập trình.

BẢNG ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

Các yêu cầu trong bài	Mức độ hoàn thành yêu cầu
Yêu cầu 1: 8x8 Tic-Tac-Toe.	100%
Yêu cầu 2: N-Queens with CNFs.	100%
Yêu cầu 3: Decision Trees	100%

Bảng 3: Bảng đánh giá mức độ hoàn thành.

TÀI LIỆU THAM KHẢO

[1] Từ Minh Phương, *Giáo trình Nhập môn trí tuệ nhân tạo,* Hà Nội, 2014, 22.

[2] GS. TSKH Hoàng Kiếm, ThS. Đình Nguyễn Anh Dũng, *Giáo Trình Nhập Môn Trí Tuệ Nhân Tạo,* NXB Đại học Quốc gia 2005, Đại học Quốc gia TP.HCM, Trường Đại học Công nghệ thông tin.

[3] J. R. Quinlan, **C4.5: Programs for Machine Learning**, 1st ed. San Mateo, CA: Morgan Kaufmann, 1993.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone, **Classification and Regression Trees**, 1st ed. Belmont, CA: Wadsworth, 1984.

[5] W. Ertel, *Introduction to Artificial Intelligence*, 2nd ed. Springer, 2018.