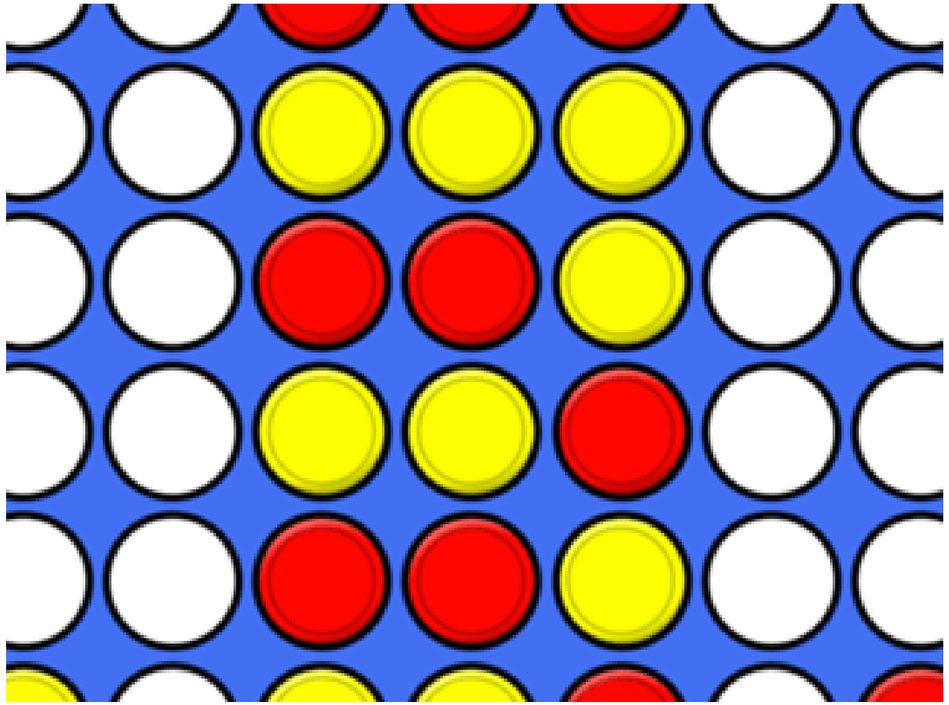
BÁO CÁO
LAB_04



Về trò chơi Connect4

- Lưới: 6 hàng × 7 cột treo đứng
- Mục tiêu: Tạo đường thẳng 4 đĩa cùng màu (ngang, dọc, chéo)
- Người chơi: Đỏ (MAX/Player 1) và Vàng (MIN/Player -1)
- Đã được giải toán học (1988): Người chơi đầu luôn thắng nếu chơi tối ưu

Mục tiêu học tập

- Triển khai thuật toán Minimax và Alpha-Beta Pruning
- Thiết kế hàm đánh giá Heuristic hiệu quả cho trạng thái trung gian
- So sánh hiệu suất các chiến lược tác tử khác nhau
- Đánh giá sự đánh đổi giữa chất lượng quyết định và tốc độ tính toán

TASK 1: ĐỊNH NGHĨA BÀI TOÁN

Thành phần	Ý nghĩa
Trạng thái (State)	Mô tả toàn bộ bàn cờ hiện tại (ma trận 6×7).
Hành động (Action)	Người chơi chọn **cột 0–6** để thả đĩa.
Mô hình chuyển đổi (Transition Model)	Xác định **trạng thái kế tiếp** sau khi đĩa được thả (rơi xuống vị trí trống thấp nhất).
Trạng thái kết thúc (Terminal State)	Khi một người chơi có **4 đĩa liên tiếp** hoặc **bảng đầy** (hòa).
Utility (Giá trị)	Đánh giá kết quả của trạng thái cuối: thắng (+1), thua (-1), hòa (0).

TASK 1: ĐỊNH NGHĨA BÀI TOÁN

Phân Tích Độ Phức Tạp

a. Kích thước Không gian Trạng thái (State Space Size)

- Ước tính (Upper Bound): $3^{R \times C} = 3^{6 \times 7} = 3^{42} \approx 1.5 \times 10^{20}$ trạng thái.
- Giải thích: Mặc dù số lượng trạng thái thực tế khả dĩ (reachable states) nhỏ hơn do luật chơi, không gian vẫn quá lớn để sử dụng tìm kiếm vét cạn (brute-force search).

b. Kích thước Cây Trò chơi (Game Tree Size)

- Ước tính: Với hệ số phân nhánh (branching factor, b) ≈ 7 và độ sâu tối đa (depth, d) ≤ 42, kích thước cây có thể lên tới $7^{42} \approx 1.6 \times 10^{35}$ node.
- Kết luận: Kích thước khổng lồ này khẳng định rằng thuật toán Minimax thô sơ là không khả thi. Các kỹ thuật tối ưu hóa như Alpha-Beta Pruning và giới hạn độ sâu là bắt buộc.

TASK 2: GAME ENVIRONMENT & RANDOM AGENT

Triển khai Cơ sở: Random Agent

- Mục đích: Kiểm tra tính năng của môi trường game và làm đối thủ benchmark cơ sở.
- Chiến lược: Tác tử random_agent chọn một hành động (cột) ngẫu nhiên từ danh sách các hành động hợp lệ.

Thiết kế hàm đánh giá Eval_function

- Ý tưởng Cốt lõi: Đánh giá trạng thái bằng cách định lượng các chuỗi đĩa liên tiếp có tiềm năng mở rộng thành chuỗi 4.
- Tính toán: Quét toàn bộ bảng theo 4 hướng (ngang, dọc, chéo chính, chéo phụ).
- Trọng số: Gán trọng số cao cho chuỗi 3 (mối đe dọa thắng gần) và thấp hơn cho chuỗi 2 (cơ hội phát triển).
- Giá trị Heuristic: Là hiệu số giữa tổng điểm Heuristic của Player 1 và Player -1.

TASK 3: TRIỂN KHAI MINIMAX VỚI ALPHA-BETA PRUNING

Thuật Toán Alpha-Beta

- **α (Alpha)**: Giá trị tối đa mà MAX có thể đảm bảo trên đường đi hiện tại
- **β (Beta):** Giá trị tối thiểu mà MIN có thể đảm bảo trên đường đi hiện tại
- Điều kiện cắt tỉa: Dừng tìm kiếm khi α ≥ β

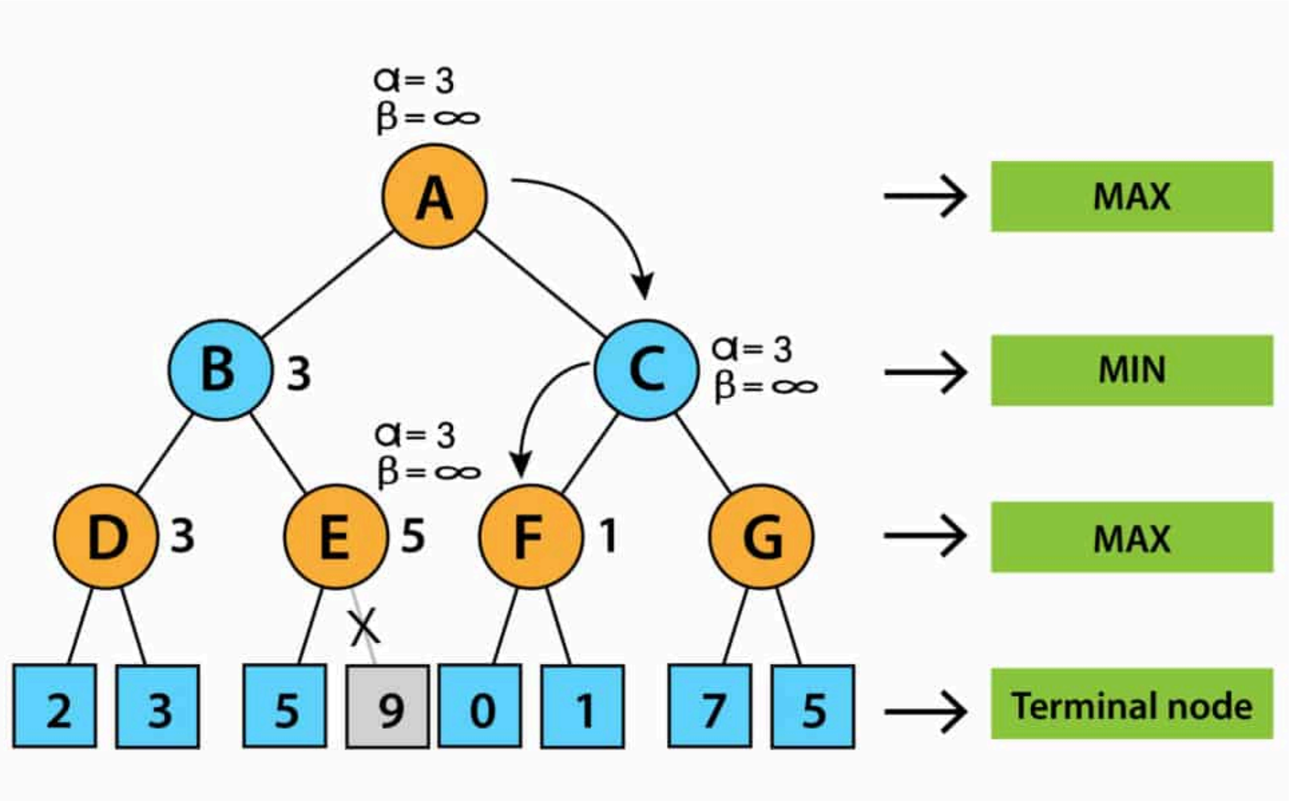
Logic MAX và MIN

MAX Player (max_value_alpha_beta):

- Tìm giá trị lớn nhất: v = max(v, min_value(...))
- Cắt tỉa Beta: Nếu v ≥ β → return v
- Cập nhật: α = max(α, v)

MIN Player (min_value_alpha_beta):

- Tìm giá trị nhỏ nhất: v = min(v, max_value(...))
- Cắt tỉa Alpha: Nếu v ≤ α → return v
- Cập nhật: β = min(β, v)



TASK 4: HEURISTIC ALPHA-BETA TREE SEARCH

```python
env = Connect4Env(6,7)
agent_deep = MinimaxAgent(max_depth=4, eval_fn=heuristic_eval)
agent_shallow = MinimaxAgent(max_depth=2, eval_fn=heuristic_eval)
winner = play_game(agent_deep, agent_shallow, env, verbose=False)
print('Winner (deep vs shallow):', winner)
```

```
Winner (deep vs shallow): 1
```

Kỹ Thuật Tối Ưu Hóa

- Move Ordering: Sắp xếp nước đi để cắt tỉa hiệu quả hơn
- Heuristic cải tiến: Pattern recognition chính xác hơn
- Best First Move: Tối ưu nước đi đầu tiên

Hướng Phát Triển Nâng Cao

- Monte Carlo Tree Search (MCTS)
- Deep Learning cho hàm đánh giá
- Opening book và Endgame database