Artificial-Intelligence

Giới thiệu

🕴️ BÁO CÁO CÁ NHÂN 🕴️ Thông tin sinh viên

Họ và tên: Tạ Hoàng Đạt

MSSV: 23110090

Lớp: Tri tue nhan tao_ Nhom 04CLC

Môn học: Trí tuệ nhân tạo

Đề tài: Giải bài toán 8 quân hậu bằng nhiều thuật toán tìm kiếm

Giảng viên hướng dẫn: TS.Phan Thị Huyền Trang

🎯 Mục tiêu

Hiểu rõ cơ chế hoạt động của các thuật toán tìm kiếm trong AI.

So sánh hiệu quả của từng thuật toán khi áp dụng cho bài toán 8 quân hậu.

Xây dựng chương trình minh họa trực quan bằng Pygame.

⚙️ Công cụ và môi trường phát triển Thành phần Mô tả Ngôn ngữ Python 3.12 Thư viện pygame, heapq, collections, tkinter, random, time, sys IDE Visual Studio 2022 / VS Code Hệ điều hành Windows 10 💡 Giới thiệu bài toán

Bài toán 8 Queens (8 quân hậu) yêu cầu đặt 8 quân hậu lên bàn cờ 8×8 sao cho không quân nào tấn công được nhau (tức là không có 2 quân cùng hàng, cùng cột hoặc cùng đường chéo).

Chương trình cài đặt nhiều thuật toán khác nhau để tìm kiếm lời giải và hiển thị trực quan quá trình tìm kiếm trên giao diện đồ họa.

🔍 Các thuật toán được cài đặt Nhóm thuật toán Cụ thể Tìm kiếm có hệ thống DFS, BFS, UCS, DLS, IDS Tìm kiếm heuristic Greedy Best-First, A*, Hill Climbing Tối ưu hóa Beam Search, Genetic Algorithm Ràng buộc Backtracking, Forward Checking Biến thể Partial Observation

Mỗi thuật toán đều có hàm riêng trong file 8Queens_23110090.py, ví dụ:

def dfs_8_queens_steps(): def bfs_8_queens_steps(): def astar_8_queens_steps(): ...

🧩 Mô tả giao diện

Giao diện sử dụng thư viện Pygame, hiển thị:

Bảng cờ bên trái: diễn tiến từng bước thuật toán.

Bảng cờ bên phải: lời giải cuối cùng (nếu tìm được).

Các nút bấm chọn thuật toán (DFS, BFS, A*, Hill, Genetic, v.v...).

Hai nút điều khiển đặc biệt:

Steps: in ra các bước trên terminal.

Skip: tua nhanh đến kết quả cuối.

▶️ Hướng dẫn chạy chương trình

Cài đặt các thư viện cần thiết:

pip install pygame

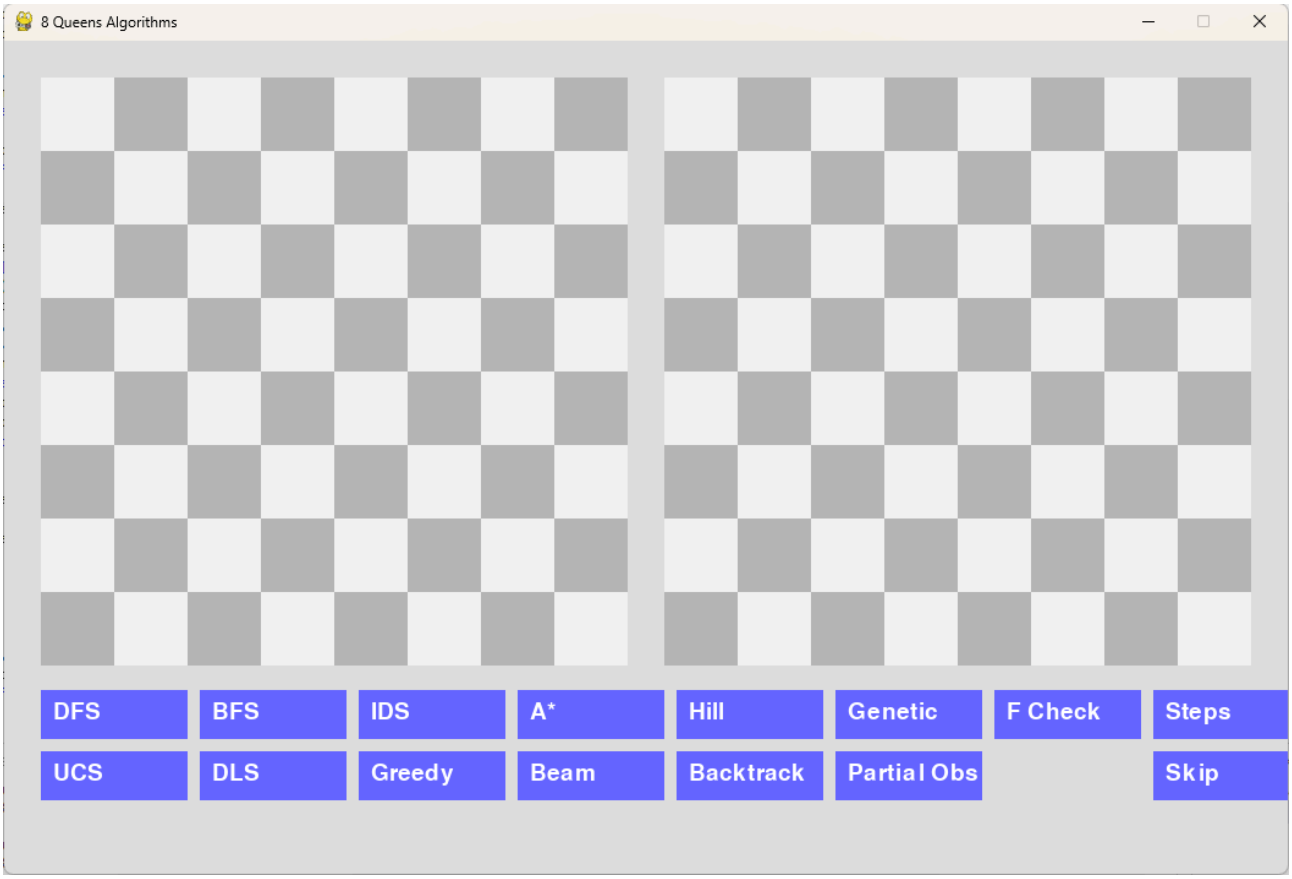
Chạy chương trình:

python 8Queens_23110090.py

Chọn thuật toán bạn muốn chạy (bấm nút trên giao diện).

Quan sát quá trình đặt quân hậu.

📸 Ảnh minh họa giao diện



📊 ĐÁNH GIÁ VÀ SO SÁNH CÁC THUẬT TOÁN 🔹 1. Nhóm thuật toán tìm kiếm có hệ thống

Thuật toán	Đặc điểm	Ưu điểm	Nhược điểm
DFS (Depth-First Search)	Duyệt sâu đến cùng, dùng ngăn xếp	Dễ cài đặt, tốn ít bộ nhớ	Có thể rơi vào nhánh sai, không đảm bảo tìm lời giải tối ưu
BFS (Breadth-First Search)	Duyệt theo từng tầng, dùng hàng đợi	Tìm được lời giải tối ưu (nếu chi phí bằng nhau)	Rất tốn bộ nhớ do lưu nhiều trạng thái
UCS (Uniform Cost Search)	Mở rộng trạng thái có chi phí thấp nhất	Đảm bảo tối ưu với chi phí khác nhau	Tốc độ chậm hơn BFS khi nhiều trạng thái
DLS (Depth-Limited Search)	DFS nhưng giới hạn độ sâu	Tránh duyệt vô hạn	Có thể bỏ sót lời giải nếu độ sâu quá nhỏ
IDS (Iterative Deepening Search)	Kết hợp BFS và DFS (tăng dần độ sâu)	Đảm bảo tìm lời giải tối ưu và tiết kiệm bộ nhớ	Tốn thời gian do lặp lại nhiều tầng tìm kiếm

- 2. Nhóm thuật toán heuristic (dựa trên hàm đánh giá)

Thuật toán	Hàm đánh giá	Ưu điểm	Nhược điểm
Greedy Best-First Search	Chọn trạng thái có `h(n)` nhỏ nhất	Nhanh, hướng đích	Không đảm bảo tối ưu, dễ rơi vào cục bộ
A*	Kết hợp `f(n) = g(n) + h(n)`	Cân bằng giữa chi phí và heuristic, tối ưu nếu `h` tốt	Tốn bộ nhớ khi không gian tìm kiếm lớn
Hill Climbing	Cải thiện dần theo hướng giảm `h(n)`	Đơn giản, ít bộ nhớ	Dễ mắc kẹt tại cực trị địa phương, không đảm bảo lời giải

- 3. Nhóm thuật toán tối ưu hóa và tiến hóa

Thuật toán	Cơ chế	Ưu điểm	Nhược điểm
Beam Search	Giữ lại `k` trạng thái tốt nhất ở mỗi bước	Giảm không gian tìm kiếm, nhanh	Có thể bỏ lỡ lời giải đúng nếu loại bỏ sớm
Genetic Algorithm	Dựa trên tiến hóa tự nhiên (lai ghép, đột biến)	Khám phá không gian lớn, có thể thoát cục bộ	Kết quả không ổn định, phụ thuộc tham số

- 4. Nhóm thuật toán ràng buộc

Thuật toán	Cơ chế	Ưu điểm	Nhược điểm
Backtracking	Dò từng lựa chọn, quay lui khi sai	Đảm bảo tìm được lời giải đúng	Tốc độ chậm khi không có heuristic
Forward Checking	Loại bỏ sớm giá trị không hợp lệ trong miền	Giảm số nhánh cần duyệt, nhanh hơn backtracking	Cài đặt phức tạp hơn

- 5. Thuật toán quan sát một phần (Partial Observation)

Dành cho trường hợp đã biết trước một phần vị trí quân hậu.

Giúp giảm không gian tìm kiếm, nhưng không linh hoạt nếu dữ liệu đầu vào bị sai.

⚖️ Tổng kết hiệu năng

Thuật toán	Độ phức tạp thời gian (ước lượng)	Hiệu quả thực tế	Khả năng tìm lời giải
DFS	O(b^d)	Nhanh, dễ rơi vào nhánh sai	Có
BFS	O(b^d)	Chậm, tốn bộ nhớ	Có
UCS	O(b^d)	Trung bình	Có
DLS	O(b^l)	Tùy giới hạn độ sâu	Có thể không
IDS	O(b^d)	Tốt, ổn định	Có
Greedy	O(b·logb)	Rất nhanh	Có thể không
A*	O(b^d)	Cân bằng, hiệu quả	Có
Hill Climbing	$O(n^2)$	Nhanh, không ổn định	Không luôn
Beam	O(k·b)	Nhanh, thiếu tối ưu	Có thể
Genetic	O(g·p)	Chậm hơn nhưng đa dạng	Có thể
Backtracking	O(N!)	Chính xác, chậm	Có
Forward Checking	< O(N!)	Nhanh hơn Backtracking	Có

🧠 Kết luận chung

Thuật toán A* và Forward Checking cho kết quả ổn định và hiệu quả nhất trong bài toán 8 quân hậu.

Greedy và Hill Climbing chạy nhanh nhưng có thể dừng ở trạng thái không tối ưu.

Genetic Algorithm thú vị vì tính ngẫu nhiên, tuy nhiên khó kiểm soát kết quả.

Backtracking tuy chậm nhưng đảm bảo lời giải chính xác tuyệt đối.