Bao_cao_ca_nhan_AI_23110170_8Queens

Môn học: Trí tuệ nhân tạo (Artificial Intelligence)
Giảng viên hướng dẫn: Phan Thị Huyền Trang
Sinh viên thực hiện: Trần Văn Tưởng
MSSV: 23110170
Lớp: ARIN330585_04CLC
Ngày nộp: 16/10/2025

1. Giới thiệu bài toán

Bài toán **8 quân hậu (8-Queens Problem)** là một bài toán kinh điển trong lĩnh vực **Trí tuệ nhân tạo (AI)** và **tìm kiếm ràng buộc (CSP – Constraint Satisfaction Problem)**.

Mục tiêu: Đặt 8 quân hậu lên bàn cờ 8x8 sao cho **không có quân nào ăn được nhau**, tức là không trùng **hàng, cột hoặc đường chéo**.

2. Công cụ và môi trường

• **Ngôn ngữ:** Python 3.x

• **Thư viện:** Tkinter (giao diện đồ họa), heapq, random, collections, math

• **IDE:** VSCode / PyCharm

• **Hệ điều hành:** Windows / Linux (chạy tốt cả hai)

3. Mô tả chương trình

Chương trình mô phỏng bài toán 8 quân hậu bằng giao diện đồ họa. Người dùng có thể chọn thuật toán khác nhau để giải và quan sát trực quan quá trình đặt hậu.

Các thuật toán đã triển khai:

1. BFS (Breadth-First Search)

2. DFS (Depth-First Search)

3. UCS (Uniform Cost Search)

4. DLS (Depth Limited Search)

5. IDS (Iterative Deepening Search)

6. Greedy Search

7. A* Search

8. Hill Climbing

9. Simulated Annealing

10. Beam Search

11. Genetic Algorithm

12. AND/OR Graph Search

13. Belief Search

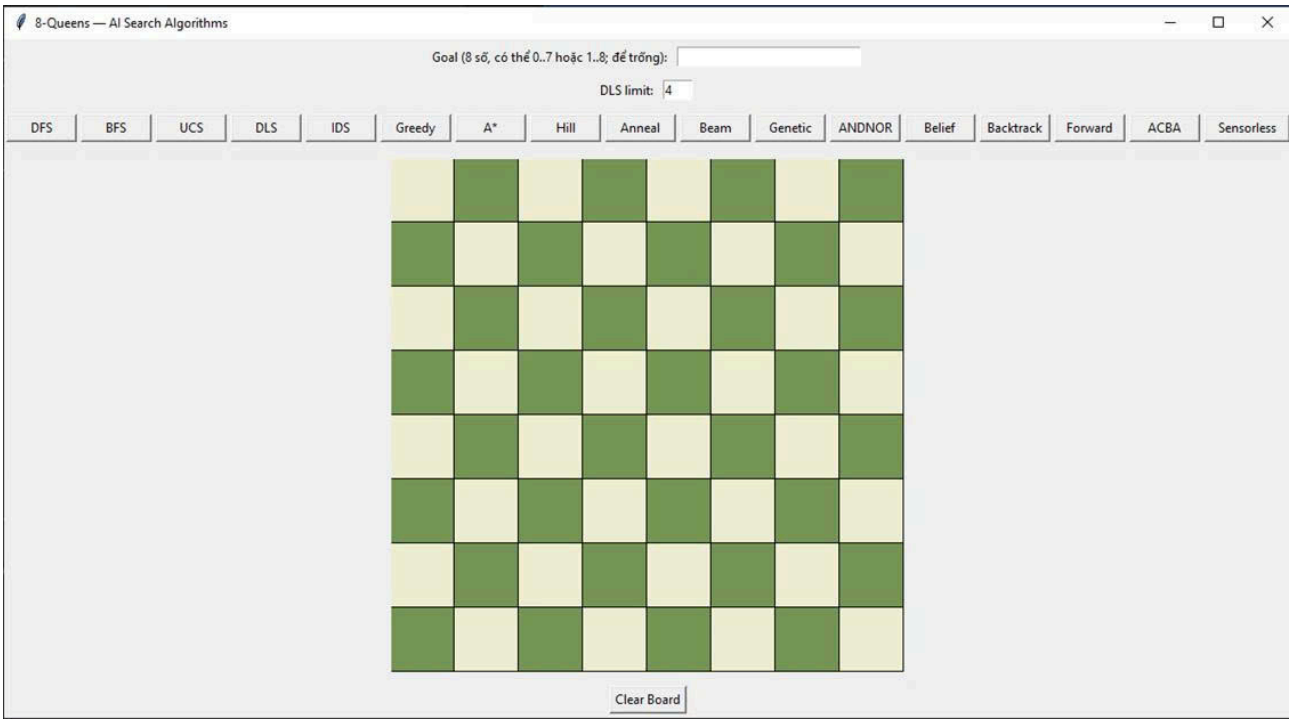
14. Backtracking

15. Forward Checking

16. AC-3 (Arc Consistency Algorithm)

17. Sensorless Belief Progression (nâng cao)

4. Giao diện chương trình



5. Cách sử dụng

1. Chạy file Python chính (main.py).

2. Giao diện hiển thị bàn cờ 8x8.

3. Nhấn chuột lên bàn cờ để đặt quân hậu ban đầu.

4. Nhập "Goal" nếu muốn kiểm tra trạng thái đích cụ thể.

5. Chọn thuật toán cần chạy → quan sát trực quan quá trình giải.

6. Nhấn Clear Board để làm lại.

7. Chọn Sensorless để xem tiến trình belief state (nâng cao).

6. Kết quả và đánh giá

• Chương trình chạy thành công, hiển thị rõ từng bước đặt hậu.

• Các thuật toán tìm kiếm cho kết quả hợp lệ, trực quan.

• Giao diện dễ sử dụng, hỗ trợ nhiều loại thuật toán khác nhau.

• Các thuật toán heuristic (A*, Greedy, Hill, Anneal, Genetic) cho tốc độ nhanh hơn so với DFS/BFS.

• So sánh hiệu quả các thuật toán

a. DFS (Depth-First Search)

• **Tìm được lời giải:** Có (thường).

• **Đặc điểm:** Thăm sâu, tốn ít bộ nhớ, có thể tìm lời giải nhanh nhưng dễ bị lạc sâu hoặc không tối ưu.

• **Khi nên dùng:** Khi cần tìm nhanh một lời giải mà không quan tâm đến tối ưu.

b. BFS (Breadth-First Search)

• **Tìm được lời giải:** Có.

• **Đặc điểm:** Tìm theo lớp, đảm bảo lời giải ngắn nhất nếu chi phí các bước bằng nhau; tốn nhiều bộ nhớ.

• **Khi nên dùng:** Khi muốn tìm lời giải ngắn nhất và máy đủ bộ nhớ.

c. UCS (Uniform-Cost Search)

• **Tìm được lời giải:** Có.

• **Đặc điểm:** Giống BFS khi chi phí bằng nhau, nhưng xử lý tốt khi mỗi bước có chi phí khác nhau.

• **Khi nên dùng:** Khi các bước có chi phí không đồng nhất.

d. DLS (Depth-Limited Search)

• **Tìm được lời giải:** Có (tùy giới hạn).

• **Đặc điểm:** DFS có giới hạn độ sâu, tránh lạc sâu vô hạn nhưng có thể bỏ sót.

• **Khi nên dùng:** Khi biết hoặc ước lượng được độ sâu cần tìm.

e. IDS (Iterative Deepening Search)

• **Tìm được lời giải:** Có.

• **Đặc điểm:** Kết hợp ưu điểm của BFS (tối thiểu bước) và DFS (bộ nhớ thấp), lặp DLS với giới hạn tăng dần.

• **Khi nên dùng:** Khi muốn cân bằng giữa bộ nhớ và thời gian.

f. Greedy Search

• **Tìm được lời giải:** Có (nhiều khi).

• **Đặc điểm:** Dùng heuristic (số cặp hậu xung đột), chạy nhanh nhưng có thể không tối ưu hoặc thất bại.

• **Khi nên dùng:** Khi cần tốc độ và heuristic đủ tốt.

g. A*

• **Tìm được lời giải:** Có (thường).

• **Đặc điểm:** Kết hợp chi phí thực (g) và heuristic (h) để chọn đường đi tối ưu.

• **Khi nên dùng:** Khi cần lời giải tối ưu và có heuristic phù hợp.

h. Hill Climbing

• **Tìm được lời giải:** Có / Dễ mắc kẹt.

• **Đặc điểm:** Tối ưu cục bộ; rất nhanh nhưng có thể dừng ở cực trị địa phương.

• **Khi nên dùng:** Khi muốn thử nghiệm nhanh hoặc kết hợp với restart để cải thiện.

i. Simulated Annealing

• **Tìm được lời giải:** Có (thường).

• **Đặc điểm:** Giống hill climbing nhưng có thể chấp nhận bước tệ trong giai đoạn đầu để thoát cực trị địa phương.

• **Khi nên dùng:** Khi hill climbing bị kẹt, cần giải pháp tốt hơn trong thời gian hợp lý.

j. Beam Search

• **Tìm được lời giải:** Có (tùy beam width).

• **Đặc điểm:** Giữ top-k trạng thái tốt nhất mỗi bước; giảm bộ nhớ nhưng có thể bỏ sót lời giải tối ưu.

• **Khi nên dùng:** Khi muốn cân bằng giữa hiệu quả và tốc độ.

k. Genetic Algorithm

• **Tìm được lời giải:** Có (thường).

• **Đặc điểm:** Dựa trên tiến hóa — quần thể, lai ghép, đột biến; không đảm bảo tối ưu tuyệt đối.

• **Khi nên dùng:** Khi không gian tìm kiếm lớn, cần phương pháp tiến hóa ngẫu nhiên.

l. AND-OR Search

• **Tìm được lời giải:** Có (tùy).

• **Đặc điểm:** Phù hợp cho bài toán có cấu trúc phân nhánh phức tạp.

• **Khi nên dùng:** Trong bài toán nâng cao hoặc có cây lựa chọn AND/OR.

m. Belief Search

• **Tìm được lời giải:** Có (cho tập belief).

• **Đặc điểm:** Xét nhiều trạng thái khả dĩ cùng lúc (sensorless idea).

• **Khi nên dùng:** Khi bài toán có yếu tố bất định hoặc thiếu cảm biến (sensorless).

n. Backtracking (CSP cơ bản)

• **Tìm được lời giải:** Có.

• **Đặc điểm:** Dễ hiểu, hiệu quả cho bài toán ràng buộc nếu có loại bỏ nhánh sớm (pruning).

• **Khi nên dùng:** Khi giải CSP cơ bản như 8-Queens.

o. Forward Checking

• **Tìm được lời giải:** Có (hiệu quả hơn Backtrack).

• **Đặc điểm:** Gỡ bỏ giá trị không hợp lệ trong các biến chưa gán → giảm nhánh duyệt.

• **Khi nên dùng:** Khi cần tăng hiệu quả so với quay lui thuần túy.

p. AC-3 (Arc-Consistency / ACBA)

• **Tìm được lời giải:** Có (đa số).

• **Đặc điểm:** Duy trì tính nhất quán cung, giúp prune mạnh hơn khi kết hợp Backtracking.

• **Khi nên dùng:** Khi muốn kiểm tra ràng buộc trước khi gán biến trong CSP.

q. Sensorless Belief Progression

• **Tìm được lời giải:** Có (tùy).

• **Đặc điểm:** Sinh progression của tập belief; minh họa tiến trình cập nhật trạng thái trong bài toán sensorless.

• **Khi nên dùng:** Trong phần mở rộng / minh họa AI nâng cao (Belief State Progression).

7. Đặc điểm, ưu và nhược điểm của các thuật toán

7.1. DFS (Depth-First Search)

• **Đặc điểm:** Duyệt sâu nhất trước, sử dụng ngăn xếp.

• **Ưu điểm:** Đơn giản, tốn ít bộ nhớ.

• **Nhược điểm:** Có thể rơi vào vòng lặp vô hạn, không tìm được lời giải tối ưu.

7.2. BFS (Breadth-First Search)

• **Đặc điểm:** Duyệt theo tầng, dùng hàng đợi.

• **Ưu điểm:** Luôn tìm được lời giải tối ưu (nếu chi phí bằng nhau).

• **Nhược điểm:** Tốn rất nhiều bộ nhớ.

7.3. UCS (Uniform-Cost Search)

• **Đặc điểm:** Mở rộng nút có chi phí thấp nhất trước.

• **Ưu điểm:** Đảm bảo tìm được lời giải tối ưu.

• **Nhược điểm:** Chậm khi có nhiều trạng thái đồng chi phí.

7.4. DLS (Depth-Limited Search)

• **Đặc điểm:** Giới hạn độ sâu của DFS.

• **Ưu điểm:** Tránh lạc sâu vô hạn.

• **Nhược điểm:** Dễ bỏ sót lời giải nếu limit nhỏ.

7.5. IDS (Iterative Deepening Search)

• **Đặc điểm:** Lặp DLS với giới hạn tăng dần.

• **Ưu điểm:** Tối ưu về bộ nhớ như DFS, nhưng vẫn tìm được lời giải tối ưu như BFS.

• **Nhược điểm:** Phải duyệt lại nhiều nút.

7.6. Greedy Search

• **Đặc điểm:** Dựa vào heuristic để chọn hướng đi "có vẻ tốt nhất".

• **Ưu điểm:** Rất nhanh.

• **Nhược điểm:** Không đảm bảo tối ưu, dễ mắc kẹt.

7.7. A*

• **Đặc điểm:** Kết hợp chi phí thật (g) và heuristic (h).

• **Ưu điểm:** Tối ưu và đầy đủ nếu heuristic hợp lệ.

• **Nhược điểm:** Cần bộ nhớ và tính toán lớn.

7.8. Hill Climbing

• **Đặc điểm:** Tìm nghiệm tốt hơn ở lân cận.

• **Ưu điểm:** Nhanh, dễ triển khai.

• **Nhược điểm:** Dễ dừng ở cực trị địa phương.

7.9. Simulated Annealing

• **Đặc điểm:** Cho phép nhận nghiệm xấu lúc đầu để thoát cực trị.

• **Ưu điểm:** Giải quyết hạn chế của Hill Climbing.

• **Nhược điểm:** Cần tinh chỉnh tham số "nhiệt độ".

7.10. Beam Search

• **Đặc điểm:** Duy trì k trạng thái tốt nhất.

• **Ưu điểm:** Giảm chi phí bộ nhớ.

• **Nhược điểm:** Có thể bỏ sót lời giải tối ưu.

7.11. Genetic Algorithm

• **Đặc điểm:** Dựa trên tiến hóa – chọn lọc, lai ghép, đột biến.

• **Ưu điểm:** Tìm lời giải gần tối ưu nhanh trên không gian lớn.

• **Nhược điểm:** Không đảm bảo hội tụ, phụ thuộc tham số.

7.12. AND-OR Search

• **Đặc điểm:** Áp dụng cho bài toán có cấu trúc phân nhánh logic.

• **Ưu điểm:** Giải được bài toán có điều kiện phức tạp.

• **Nhược điểm:** Khó cài đặt, chi phí lớn.

7.13. Belief Search

• **Đặc điểm:** Duyệt theo tập trạng thái khả dĩ.

• **Ưu điểm:** Xử lý được bài toán sensorless hoặc có bất định.

• **Nhược điểm:** Số trạng thái tăng nhanh.

7.14. Backtracking

• **Đặc điểm:** Thử – sai, quay lui khi không hợp lệ.

• **Ưu điểm:** Hiệu quả cho bài toán ràng buộc nhỏ.

• **Nhược điểm:** Dễ bị bùng nổ tổ hợp nếu không prune.

7.15. Forward Checking

• **Đặc điểm:** Cập nhật miền giá trị còn lại khi gán biến.

• **Ưu điểm:** Giảm đáng kể số nhánh duyệt.

• **Nhược điểm:** Cần lưu trữ thêm thông tin miền giá trị.

7.16. AC-3 (Arc Consistency)

• **Đặc điểm:** Duy trì tính nhất quán cung giữa các biến.

• **Ưu điểm:** Loại bỏ sớm các giá trị vô nghĩa, tăng hiệu quả CSP.

• **Nhược điểm:** Tốn thời gian xử lý khi số biến lớn.

7.17. Sensorless Belief Progression

• Đặc điểm: Duyệt theo các tập belief không có cảm biến.

• Ưu điểm: Minh họa trực quan cho bài toán nâng cao.

• Nhược điểm: Chỉ dùng cho mô phỏng / phần nâng cao.

8. Video demo

(https://drive.google.com/file/d/1Q5K-mzkxph1eAhakW17_XhKbjHKf0oBb/view?usp=sharing)

9. Kết luận

• Đã cài đặt và chạy thành công bài toán 8 quân hậu với nhiều thuật toán tìm kiếm AI.

• Học được cách áp dụng thuật toán heuristic và CSP vào bài toán thực tế.

• Giao diện giúp quan sát và so sánh hiệu quả trực quan giữa các phương pháp.

10. Tài liệu tham khảo

1. Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach (3rd ed.). Pearson Education.
2. Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). Pearson Education.
3. Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press.
4. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.
5. Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly Media.
6. Lapan, M. (2020). Deep Reinforcement Learning Hands-On. Packt Publishing.
7. Kong, Q. (2021). Python Programming and Numerical Methods: A Guide for Engineers and Scientists. Academic Press.
8. Slide giảng dạy môn Trí tuệ Nhân tạo – Khoa CNTT, Trường ĐH Sư phạm Kỹ thuật TP.HCM (2024).

9. Python Software Foundation. (2024). Python 3.x Documentation. Retrieved from https://docs.python.org/3/
10. Tkinter GUI Documentation. (2024). Retrieved from https://docs.python.org/3/library/tkinter.html