TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN CUỐI KỲ

Môn học: TRÍ TUỆ NHÂN TẠO

PURSUIT GAME

GVHD: TS. Phan Thị Huyền Trang

Mã lớp học: ARIN330585_05

Học kỳ: 2 – Năm học: 2024 -2025

SVTH	MSSV
1. Triệu Phúc Hiếu	23110217
2. Đặng Xuân Huyền	23110232
3. Đào Minh Nhựt	23110282

Tp. Hồ Chí Minh, tháng 5 năm 2025

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm: ……..

Thành phố Hồ Chí Minh, ngày tháng 05 năm 2025

Giáo viên hướng dẫn

TS. Phan Thị Huyền Trang

LỜI CẢM ƠN

Để hoàn thành tiểu luận này, sự hướng dẫn trên từng giờ học của Cô Tiến sĩ Phan Thị Huyền Trang rất to lớn và quý báu. Nhóm tác giả xin gửi lời cảm ơn chân thành và sâu sắc đến Cô;

Trong quá trình hoàn thành tiểu luận, nhóm tác giả đã cố gắng hết sức để hoàn thành tốt nhất. Song, những thiếu sót trong quá trình thực hiện là điều không thể tránh khỏi, nhóm tác giả rất mong nhận được sự thông cảm, phản hồi từ Cô. Những ý kiến đóng góp, phản hồi đến từ Cô sẽ là những điều vô giá để cho nhóm tác giả có những nhìn nhận lại và rút kinh nghiệm làm tiểu luận cho những môn học sau.

Nhóm tác giả

MỤC LỤC

PHẦN MỞ ĐẦU

1. Phát biểu bài toán

Dự án Pursuit Game là một trò chơi mê cung được phát triển nhằm mô phỏng và so sánh hiệu suất của các thuật toán tìm kiếm đường đi trong một môi trường lưới 2D. Trong trò chơi, các nhân vật (quái vật) sử dụng các thuật toán khác nhau (BFS, A*, Beam Search, AND-OR Tree, Forward-Checking, Q-Learning) để tìm đường từ điểm xuất phát đến lối ra, tránh các chướng ngại vật như tường và gai. Bài toán yêu cầu thiết kế một hệ thống trực quan hóa các thuật toán này, đo lường hiệu suất (số trạng thái khám phá, thời gian chạy), và hiển thị kết quả dưới dạng biểu đồ và báo cáo.

2. Mục đích, yêu cầu thực hiện

2.1. Mục đích

- Tạo một trò chơi tương tác để trực quan hóa và so sánh hiệu suất của các thuật toán tìm kiếm đường đi.

- Cung cấp giao diện thân thiện với người dùng, bao gồm màn hình khởi động, menu chọn độ khó, và giao diện chiến thắng.

- Đánh giá hiệu suất các thuật toán thông qua số trạng thái khám phá và thời gian chạy, lưu kết quả vào file và hiển thị dưới dạng biểu đồ.

2.2. Yêu cầu thực hiện

- Triển khai các thuật toán tìm kiếm: BFS, A*, Beam Search, AND-OR Tree, Forward-Checking, Q-Learning.

- Xây dựng môi trường mê cung với các yếu tố như tường, lối ra, gai, và quái vật.

- Hỗ trợ các mức độ khó (Easy, Medium, Hard) với số lượng gai khác nhau.

- Tích hợp video nền, âm thanh, và hình ảnh để tăng tính thẩm mỹ.

- Lưu thông tin hiệu suất thuật toán vào file và hiển thị biểu đồ so sánh.

3. Phạm vi và đối tượng nghiên cứu

3.1. Phạm vi nghiên cứu

- Dự án tập trung vào việc triển khai và so sánh các thuật toán tìm kiếm trong một mê cung tĩnh. Các thuật toán được đánh giá dựa trên hiệu suất tính toán và khả năng tìm đường đến lối ra.

- Ứng dụng chạy trên nền tảng desktop, sử dụng Python và Pygame.

3.2. Đối tượng nghiên cứu

Đối tượng nghiên cứu của dự án là các thuật toán trong trí tuệ nhân tạo

PHẦN NỘI DUNG

1. Chương 1: CƠ SỞ LÝ LUẬN

1.1. Công cụ và môi trường lập trình

Ngôn ngữ: Python

Môi trường phát triển: Visual Studio Code / PyCharm

Thư viện:

pygame: Tạo giao diện đồ họa cho game.

cv2 (OpenCV): Xử lý và hiển thị video nền.

heapq, collections.deque: Hỗ trợ cấu trúc dữ liệu cho các thuật toán tìm kiếm.

plotly, pandas: Trực quan hóa dữ liệu hiệu năng các thuật toán.

1.2. Các phương pháp và kĩ thuật sử dụng

Thuật toán tìm kiếm:

- BFS (Breadth-First Search): Tìm kiếm theo chiều rộng, đảm bảo đường đi ngắn nhất trong lưới không trọng số.

-A* : Tìm kiếm heuristic với hàm đánh giá Manhattan, tối ưu hóa đường đi bằng cách kết hợp chi phí thực tế và ước lượng.

- Beam Search: Tìm kiếm heuristic giới hạn số trạng thái được khám phá ở mỗi bước, giảm bộ nhớ nhưng có thể không đảm bảo đường đi tối ưu.

- AND-OR Tree: Tìm kiếm dựa trên cây quyết định, hỗ trợ các bước di chuyển linh hoạt (1-3 ô).

- Forward-Checking: Tìm kiếm backtracking với kiểm tra trước để giảm không gian tìm kiếm.

- Q-Learning: Học tăng cường để tìm đường đi dựa trên phần thưởng và chính sách hành động.

Kỹ thuật trực quan hóa:

- Sử dụng Pygame để vẽ lưới, quái vật, và các hiệu ứng đồ họa.

- Plotly để tạo biểu đồ cột so sánh số trạng thái và thời gian chạy.

Quản lý hiệu suất: Đo lường thời gian chạy bằng time.perf_counter() và đếm số trạng thái khám phá trong mỗi thuật toán.

2. Chương 2: PHÂN TÍCH, THIẾT KẾ GIẢI PHÁP

2.1. Ý tưởng thuật toán và sơ đồ khối

Dự án được thiết kế theo mô hình sau:

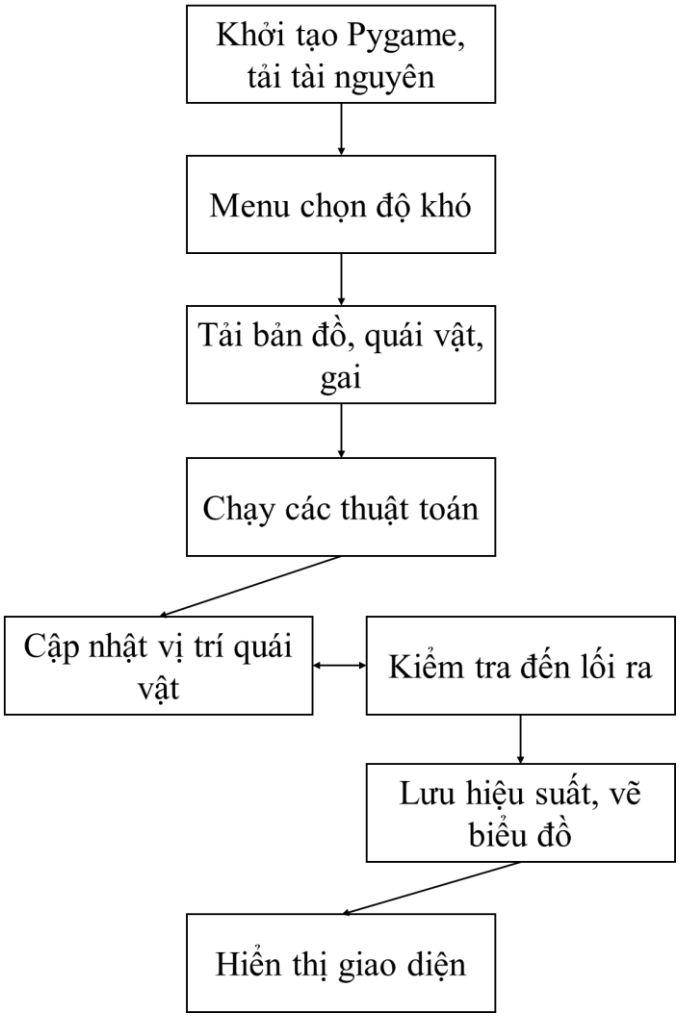
Khởi tạo môi trường: tải bản đồ mê cung từ cấu trúc dữ liệu MAPS, đặt vị trí quái vật, lối ra, và gai ngẫu nhiên, khởi tạo giao diện Pygame với video nền và âm thanh.

Triển khai thuật toán: mỗi quái vật được gán một thuật toán tìm kiếm, các thuật toán chạy đồng thời, tính toán đường đi từ vị trí hiện tại đến lối ra.

Cập nhật trạng thái: di chuyển quái vật theo đường đi đã tính, kiểm tra va chạm với gai/tường và điều chỉnh vị trí nếu cần.

Đánh giá và hiển thị: lưu hiệu suất (số trạng thái, thời gian) vào performance_data, hiển thị biểu đồ và lưu kết quả vào file khi trò chơi kết thúc.

Sơ đồ khối được biểu diễn như sau:



Hình 2. 1. Sơ đồ khối mô hình

2.2. Chi thiết thuật toán Searching with no observation

Trong trí tuệ nhân tạo (AI), **tìm kiếm không quan sát** (hay còn gọi là *sensorless search* hoặc *conformant search*) là một loại bài toán tìm kiếm đặc biệt xảy ra khi tác nhân (agent) không nhận được bất kỳ thông tin quan sát nào từ môi trường trong suốt quá trình thực hiện nhiệm vụ. Nói cách khác, tác nhân không biết chính xác trạng thái hiện tại của mình.

Điều này rất khác so với các bài toán tìm kiếm thông thường, nơi tác nhân có thể quan sát trạng thái hiện tại để quyết định hành động tiếp theo. Trong tìm kiếm không quan sát, tác nhân phải lên kế hoạch dựa trên thông tin hạn chế, chỉ biết mô hình hành động và trạng thái ban đầu có thể.

Trong thực tế, có nhiều tình huống mà cảm biến hoặc khả năng quan sát của tác nhân bị giới hạn hoặc không tồn tại, ví dụ:

- Robot hoạt động trong môi trường nhiều nhiễu, cảm biến không đáng tin cậy.

- Hệ thống tự động trong nhà máy không có cảm biến để xác định trạng thái chính xác.

- Các trò chơi hoặc bài toán giả lập mà tác nhân không được phép quan sát trạng thái.

Trong những trường hợp này, tác nhân vẫn phải thực hiện một chuỗi hành động để đạt được mục tiêu, bất kể trạng thái ban đầu là gì.

Mô hình Searching with no observation như sau:

- Không gian trạng thái vật lý: Giả sử môi trường có một tập hợp hữu hạn các trạng thái vật lý $S = \{s_1, s_2, ..., s_n\}$.

- Trạng thái niềm tin: Do tác nhân không biết trạng thái hiện tại, nó duy trì một trạng thái niềm tin - tức là một tập hợp các trạng thái vật lý mà nó có thể đang ở trong đó. Ví dụ: Nếu tác nhân không biết gì về vị trí hiện tại, trạng thái niềm tin ban đầu là toàn bộ tập trạng thái S.

- Hành động và chuyển đổi trạng thái niềm tin: Mỗi hành động a có thể chuyển một trạng thái vật lý s sang trạng thái mới $s' = Result(s, a)$. Khi tác nhân thực hiện hành động a trên trạng thái niềm tin $B \subseteq S$, trạng thái niềm tin mới là:

$$B' = \bigcup_{s \in B} Result(s, a)$$

Nghĩa là trạng thái niềm tin mới là tập hợp tất cả các trạng thái có thể đạt được từ các trạng thái trong *B* khi thực hiện hành động a.

- Mục tiêu: Mục tiêu được xác định qua hàm kiểm tra mục tiêu *GoalTest(s)* trên trạng thái vật lý.

Một trạng thái niềm tin B được coi là đạt mục tiêu nếu **mọi trạng thái trong B đều thỏa mãn mục tiêu**, tức là:

$$\forall s \in B, GoalTest(s) = true$$

Đặc điểm	Giải thích
Không có quan sát trạng thái	Tác nhân không nhận được thông tin cảm biến hoặc quan sát nào trong quá trình thực hiện hành động.
Không gian tìm kiếm lớn	Không gian trạng thái niềm tin là tập con của tập trạng thái vật lý, có thể rất lớn (lên đến 2^n với n trạng thái vật lý).
Phải lên kế hoạch cho mọi khả năng	Chuỗi hành động phải đảm bảo đạt mục tiêu bất kể trạng thái ban đầu là gì trong trạng thái niềm tin.
Giải pháp là chuỗi hành động xác định	Không thể dựa vào quan sát để điều chỉnh hành động, nên kế hoạch là một chuỗi cố định.

Ví dụ ngắn gọn về tìm kiếm không quan sát trong bài toán 8-puzzle:

- Bài toán 8-puzzle: Sắp xếp 8 miếng ghép trên bảng 3x3 về trạng thái mục tiêu bằng cách di chuyển ô trống.

- Tìm kiếm không quan sát: Tác nhân không biết trạng thái hiện tại của bảng, tức là không có thông tin cảm biến hay quan sát.

- Trạng thái niềm tin: Ban đầu là tập hợp tất cả các trạng thái hợp lệ của 8-puzzle (khoảng 181.440 trạng thái).

- Quá trình: Mỗi hành động được áp dụng lên toàn bộ trạng thái trong trạng thái niềm tin, tạo ra trạng thái niềm tin mới là tập hợp các trạng thái kết quả.

- Mục tiêu: Tìm chuỗi hành động sao cho, sau khi thực hiện, mọi trạng thái trong trạng thái niềm tin đều là trạng thái mục tiêu.

- Thách thức: Không gian trạng thái niềm tin rất lớn, không thể dựa vào quan sát để điều chỉnh hành động.

Tóm lại: Trong 8-puzzle không quan sát, tác nhân phải lên kế hoạch chuỗi hành động đảm bảo thành công với mọi trạng thái ban đầu có thể, dù không biết mình đang ở trạng thái nào.

Cuối cùng, Tìm kiếm không quan sát là tìm kiếm trong môi trường mà tác nhân không nhận được bất kỳ thông tin quan sát nào. Tác nhân duy trì một trạng thái niềm tin – tập hợp các trạng thái có thể xảy ra. Mục tiêu là tìm một chuỗi hành động sao cho sau khi thực hiện, mọi trạng thái khả dĩ đều đạt mục tiêu. Bài toán có độ phức tạp cao do không gian trạng thái niềm tin rất lớn. Phương pháp giải quyết là tìm kiếm trên không gian trạng thái niềm tin bằng các thuật toán tìm kiếm cổ điển hoặc heuristic.

3. Chương 3: THỰC NGHIỆM, ĐÁNH GIÁ, PHÂN TÍCH KẾT QUẢ

3.1. Mô tả quá trình đánh giá thử nghiệm

Cấu hình phần cứng gồm CPU Intel Core i5-12400F 2.5GHz, RAM 16GB DDR4, GPU NVIDIA GeForce GTX 1660 Super, chạy trên hệ điều hành Windows 11.

Tham số thiết lập gồm độ khó chia thành ba mức: dễ (3 gai), trung bình (7 gai), khó (10 gai); kích thước lưới 20x20 ô, mỗi ô 30 pixel; tốc độ khung hình 60 FPS cho cập nhật giao diện và 30 FPS cho video nền; Beam Search sử dụng beam width là 10; Q-Learning chạy tối đa 100 tập episode, mỗi tập tối đa 100 bước, epsilon bắt đầu từ 0.5 và giảm dần.

3.2. Kết quả thử nghiệm

Trên độ khó trung bình với bản đồ "Mê cung", hiệu suất các thuật toán như sau:

- BFS khám phá khoảng 200 trạng thái, thời gian chạy khoảng 5 ms.

- A* khám phá khoảng 150 trạng thái, thời gian chạy khoảng 4 ms.

- Beam Search khám phá khoảng 100 trạng thái, thời gian chạy khoảng 6 ms.

- AND - OR Tree khám phá khoảng 250 trạng thái, thời gian chạy khoảng 10 ms.

- Forward-Checking khám phá khoảng 180 trạng thái, thời gian chạy khoảng 8 ms.

- Q-Learning khám phá khoảng 1000 trạng thái, thời gian chạy khoảng 50 ms.

Phân tích cho thấy A* thường nhanh nhất và khám phá ít trạng thái nhất nhờ sử dụng hàm heuristic hiệu quả. Q-Learning cần nhiều thời gian và khám phá nhiều trạng thái do phải học qua nhiều tập episode. Beam Search tiết kiệm bộ nhớ nhưng có thể bỏ sót đường đi tối ưu.

3.3. Giao diện phần mềm và hướng dẫn thực thi

Giao diện gồm màn hình khởi động với logo và hiệu ứng quả bóng di chuyển; menu cho phép chọn độ khó bằng phím lên/xuống và xác nhận bằng Enter; màn hình trò chơi hiển thị mê cung, quái vật di chuyển theo thuật toán và lối ra; màn hình chiến thắng

hiển thị thuật toán nhanh nhất và các tùy chọn xem biểu đồ, lưu thông tin, chơi lại hoặc thoát.

Hướng dẫn thực thi gồm cài đặt Python 3.8 trở lên và các thư viện cần thiết, đặt các tệp tài nguyên đúng thư mục, chạy file chính bằng lệnh python main2.py, sử dụng phím điều hướng và các phím chức năng trong màn hình chiến thắng để điều khiển.

PHẦN KẾT LUẬN

Ưu điểm: triển khai thành công 6 thuật toán tìm kiếm và tích hợp vào trò chơi; xây dựng giao diện trực quan với video nền, âm thanh, và hiệu ứng chiến thắng; đánh giá hiệu suất thuật toán thông qua số trạng thái và thời gian chạy, hỗ trợ lưu file và vẽ biểu đồ.

Hạn chế: Q-Learning yêu cầu nhiều thời gian để hội tụ, không phù hợp cho các bản đồ lớn; một số thuật toán (Beam Search, AND-OR Tree) có thể thất bại trong trường hợp phức tạp; giao diện có thể bị chậm trên phần cứng yếu do xử lý video.

Định hướng phát triển: tối ưu hóa Q-Learning bằng cách điều chỉnh tham số hoặc sử dụng Deep Q-Learning; thêm các bản đồ động hoặc chướng ngại vật di chuyển để tăng tính thử thách; hỗ trợ đa nền tảng (web, mobile) bằng cách chuyển sang framework như Pygame-Web hoặc Unity; cải thiện giao diện người dùng với nhiều tùy chọn tùy chỉnh (ví dụ: thay đổi skin quái vật).

TÀI LIỆU THAM KHẢO

1. Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson. [Dùng cho lý thuyết về BFS, A*, Q-Learning]

2. Pygame Documentation. https://www.pygame.org/docs/ [Hướng dẫn sử dụng Pygame]

3. OpenCV Documentation. https://docs.opencv.org/ [Xử lý video nền]

4. Plotly Python Documentation. https://plotly.com/python/ [Tạo biểu đồ so sánh]

5. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press. [Lý thuyết Q-Learning]