ĐỒ ÁN CUỐI KỲ MÔN INTRODUCTION TO AI

Đề tài 1: Tối ưu lịch trình công việc cá nhân bằng thuật toán heuristic

Mô tả bài toán:

Trong một ngày làm việc, bạn có nhiều công việc cần hoàn thành, mỗi việc có khoảng thời gian thực hiện dự kiến, mức độ ưu tiên và có thể có phụ thuộc với các công việc khác. Mục tiêu là xây dựng một chương trình giúp sắp xếp lịch trình công việc sao cho:

- Không có công việc bị trùng thời gian,
- Ưu tiên hoàn thành các công việc quan trọng trước,
- Tuân thủ các ràng buộc phụ thuộc giữa các công việc,
- Tối ưu tổng thời gian hoàn thành hoặc giảm thiểu khoảng trống thời gian.

Gợi ý:

- Mô hình bài toán như một bài toán lập lịch (scheduling) có ràng buộc.
- Áp dụng thuật toán heuristic như thuật toán A* hoặc greedy để tìm lịch trình tối ưu hoặc gần tối ưu.
- Heuristic có thể là tổng thời gian trễ, số công việc chưa sắp xếp, hay độ ưu tiên tổng cộng.
- Phát triển giao diện đơn giản cho phép nhập công việc và hiển thị lịch trình.

Yêu cầu tối thiểu:

- Nhập danh sách công việc với thời gian dự kiến, mức độ ưu tiên và phụ thuộc (nếu có).
- Sắp xếp lịch trình không trùng thời gian, ưu tiên công việc quan trọng trước, tuân thủ ràng buộc phụ thuộc.
- Xuất lịch trình ngày hoàn chỉnh.

Gợi ý yêu cầu nâng cao (tùy chọn):

- Hỗ trợ lịch trình đa ngày, công việc kéo dài nhiều ngày.
- Tự động điều chỉnh ưu tiên dựa trên độ trễ (đẩy cao mức ưu tiên nếu công việc bị hoãn lâu).
- Áp dụng thuật toán heuristic kết hợp local search để cải thiện lịch trình.
- Giao diện nhập liệu và hiển thị lịch trình.

Đề tài 2: Tối ưu tuyến đường giao hàng sử dụng thuật toán heuristic

Mô tả bài toán:

Bạn có một kho hàng và danh sách các điểm giao hàng trên bản đồ 2D. Người giao hàng cần đi qua tất cả các điểm này và quay lại kho, sao cho tổng quãng đường đi là nhỏ nhất. Đây là bài toán Travelling Salesman Problem (TSP) kinh điển. Bạn hãy xây dựng chương trình sử dụng thuật toán heuristic để tìm lộ trình tối ưu hoặc gần tối ưu.

Gợi ý:

- Mô hình hóa các điểm bằng tọa độ (x, y).
- Sử dụng thuật toán Greedy (chọn điểm gần nhất), A* hoặc thuật toán tìm kiếm cục bộ (local search) với heuristic khoảng cách Euclidean.
- Cho phép nhập danh sách điểm và xuất kết quả tuyến đường + tổng quãng đường.
- Có thể minh họa tuyến đường trên đồ thị.

Yêu cầu tối thiểu:

- Nhập tọa độ kho và điểm giao hàng (x,y) ít nhất 5 – 10 điểm.
- Tính tuyến đường bắt đầu từ kho, đi qua tất cả điểm, quay lại kho sao cho tổng quãng đường Euclidean nhỏ nhất.
- Xuất thứ tự điểm giao hàng và tổng quãng đường.

Gợi ý yêu cầu nâng cao (tùy chọn):

- Sử dụng dữ liệu thực tế từ API bản đồ: Google, OpenStreetMap.
- Thêm các ràng buộc thời gian hoặc chi phí giao hàng (thời gian bắt đầu, giới hạn thời gian phục vụ, ...)
- Hiển thị đồ thị tuyến đường giao hàng.
- Sử dụng thuật toán nâng cao heuristic/metaheuristic (2-opt hoặc 3-opt, Simulated Annealing, ...) và so sánh kết quả với thuật toán cơ bản.

Đề tài 3: Tìm đường thoát thông minh trong trò chơi mê cung

Mô tả bài toán:

Bạn có bản đồ mê cung dưới dạng ma trận 2D, với các ô trống, tường, điểm bắt đầu (S) và điểm đích (G). Nhiệm vụ là viết chương trình giúp nhân vật hoặc robot tìm đường đi ngắn nhất từ S đến G, sử dụng thuật toán tìm kiếm có heuristic (ví dụ A*). Thuật toán cần tính toán sao cho hiệu quả nhất và in ra đường đi.

Gợi ý:

- Xây dựng ma trận mê cung, xác định các ô đi được và vật cản.
- Cài đặt thuật toán A* với heuristic Manhattan hoặc Euclidean distance.
- In ra đường đi dưới dạng danh sách tọa độ hoặc minh họa trên bản đồ.

Yêu cầu tối thiểu:

- Nhập mê cung ma trận 2D, xác định điểm S và G từ file hoặc hardcoded (ma trận ký tự).
- Dùng thuật toán A* để tìm đường đi từ S đến G sao cho: số bước đi là ít nhất có thể và không được đi qua tường.
- In đường đi trên bản đồ.

- Xử lý trường hợp không có đường thoát.

Gợi ý yêu cầu nâng cao (tùy chọn):

- Mê cung có ô với chi phí di chuyển khác nhau, thuật toán tìm đường tối ưu theo chi phí thay vì bước đi.
- Mê cung động: vật cản có thể thay đổi khi robot di chuyển, thuật toán phải xử lý lại đường đi.
- So sánh hiệu quả các heuristic khác nhau.
- Giao diện hiển thị quá trình tìm đường theo thời gian thực.
- Mở rộng mê cung: thêm các vật phẩm cần thu thập trước khi đến đích, ...

Đề tài 4: Robot hút bụi tìm đường làm sạch căn hộ

Mô tả bài toán:

Mô phỏng robot hút bụi trong căn hộ được chia thành các ô. Mỗi ô có thể sạch hoặc bẩn, robot cần di chuyển để làm sạch toàn bộ các ô bẩn trong thời gian và năng lượng tối thiểu. Môi trường có thể có vật cản. Chương trình cần tìm lộ trình đi qua tất cả ô bẩn bằng thuật toán heuristic.

Gợi ý:

- Mô hình lưới 2D gồm: robot, vật cản, ô bụi bẩn, ô sạch.
- Sử dụng thuật toán tìm đường để tìm đường đến từng ô bẩn.
- Kết hợp heuristic để chọn ô bẩn tiếp theo (ví dụ chọn ô gần nhất hoặc tổng khoảng cách ước tính).
- Xuất kết quả là lộ trình di chuyển và số bước.

Yêu cầu tối thiểu:

- Nhập bản đồ căn hộ với robot, bụi bẩn, vật cản từ file hoặc hardcoded (ma trận ký tự).
- Robot phải tìm đường làm sạch hết các ô bụi, sử dụng thuật toán A* và heuristic chọn điểm bụi gần nhất.
- Robot không được đi xuyên tường và chỉ có thể di chuyển 4 hướng (trên, dưới, trái phải)
- Xuất lộ trình di chuyển và tổng số bước đi (có thể cập nhật bản đồ là chỗ nào đã đi qua).

Gợi ý yêu cầu nâng cao (tùy chọn):

- Robot có giới hạn năng lượng (chỉ đi được N bước trước khí hết pin và phải quay lại trạm sạc C để nạp lại), phải lên kế hoạch vừa làm sạch vừa quay lại chỗ sạc đúng lúc.
- Bản đồ căn hộ nhiều phòng, robot phải di chuyển qua các cửa ra vào.
- Mô phỏng giao diện hiển thị robot di chuyển và trạng thái làm sạch.
- Bụi có thể quay trở lại sao một khoảng thời gian (tính theo số bước di chuyển của robot) và robot phải quay lại hút bụi.
- Hỗ trợ bản đồ phức tạp với các loại vật cản đa dạng.

Đề tài 5: Giải bài toán sudoku sử dụng heuristic

Mô tả bài toán:

Sudoku là trò chơi điền số vào bảng 9x9 sao cho mỗi hàng, cột, và vùng 3x3 không trùng số từ 1 đến 9. Bạn hãy xây dựng chương trình giải sudoku sử dụng thuật toán backtracking kết hợp heuristic để tối ưu tốc độ giải. Heuristic có thể là chọn ô trống có ít khả năng điền nhất trước.

Gợi ý:

- Cài đặt thuật toán backtracking cơ bản.
- Thêm heuristic: luôn chọn ô có ít số khả thi nhất để điền tiếp.
- Kiểm tra tính hợp lệ sau mỗi bước điền.
- Xuất ra lời giải hoặc báo không có lời giải nếu bài sudoku cho trước không hợp lệ.

Yêu cầu tối thiểu:

- Nhập bảng Sudoku 9x9 chưa hoàn chỉnh.
- Giải sudoku bằng thuật toán backtracking kết hợp heuristic chọn ô có ít khả năng điền giá trị hợp lệ nhất (Minimum Remaining Values - MRV). Kiểm tra ràng buộc trong sudoku.
- Xuất lời giải hoặc báo không có lời giải.

Gợi ý yêu cầu nâng cao (tùy chọn):

- Hỗ trợ giải biến thể sudoku (16x16, sudoku X với ràng buộc đường chéo).
- Xây dựng giao diện nhập liệu và hiển thị lời giải.
- Hiển thị số lần thử, thời gian giải.
- Tự động phát hiện sudoku không có lời giải.
- Tạo đề tự động và đảm bảo có lời giải duy nhất.