**BÁO CÁO CÁ NHÂN CÁC THUẬT TOÁN**

**Huỳnh Tấn Vinh\_23110365**

**Breadth-First Search (BFS)**

**Khái niệm**: BFS là thuật toán tìm kiếm theo chiều rộng, khám phá tất cả các trạng thái ở cùng một độ sâu trước khi chuyển sang độ sâu tiếp theo. Nó sử dụng hàng đợi để quản lý các trạng thái cần khám phá.

**Chức năng**: Tìm đường đi ngắn nhất từ trạng thái ban đầu đến trạng thái mục tiêu trong không gian tìm kiếm không có trọng số.

**Ưu điểm**:

Đảm bảo tìm được đường đi ngắn nhất nếu tồn tại.

Hoàn chỉnh (luôn tìm được giải pháp nếu có).

Dễ triển khai và phù hợp với các bài toán có không gian trạng thái nhỏ.

**Depth-First Search (DFS)**

**Khái niệm**: DFS là thuật toán tìm kiếm theo chiều sâu, khám phá một nhánh đến độ sâu tối đa trước khi quay lại và thử nhánh khác. Nó sử dụng ngăn xếp để lưu trạng thái.

**Chức năng**: Tìm đường đi từ trạng thái ban đầu đến mục tiêu, thường không đảm bảo đường đi ngắn nhất.

**Ưu điểm**:

Sử dụng ít bộ nhớ hơn BFS vì chỉ lưu một đường đi tại một thời điểm.

Hiệu quả trong không gian tìm kiếm sâu và hẹp.

Dễ triển khai và có thể tìm giải pháp- Phù hợp với các bài toán như tìm đường trong mê cung.

**Uniform Cost Search (UCS)**

**Khái niệm**: UCS là thuật toán tìm kiếm mở rộng trạng thái có chi phí thấp nhất trước, sử dụng hàng đợi ưu tiên để quản lý các trạng thái dựa trên chi phí đường đi.

**Chức năng**: Tìm đường đi có chi phí tối thiểu từ trạng thái ban đầu đến mục tiêu trong không gian tìm kiếm có trọng số.

**Ưu điểm**:

Đảm bảo tìm được đường đi tối ưu về chi phí.

Hoàn chỉnh và tối ưu trong các bài toán có chi phí không âm.

Phù hợp với các bài toán như định tuyến mạng hoặc tìm đường đi ngắn nhất.

**Iterative Deepening Search (IDS)**

**Khái niệm**: IDS kết hợp ưu điểm của BFS và DFS, thực hiện tìm kiếm theo chiều sâu với giới hạn độ sâu tăng dần cho đến khi tìm thấy mục tiêu.

**Chức năng**: Tìm đường đi từ trạng thái ban đầu đến mục tiêu, đảm bảo đường đi ngắn nhất với bộ nhớ thấp.

**Ưu điểm**:

Sử dụng ít bộ nhớ hơn BFS (giống DFS).

Đảm bảo tìm được đường đi ngắn nhất như BFS.

Phù hợp với các bài toán có không gian trạng thái lớn nhưng cần giải pháp tối ưu.

**Greedy Search**

**Khái niệm**: Greedy Search chọn trạng thái lân cận có giá trị heuristic thấp nhất (ước lượng khoảng cách đến mục tiêu) mà không xem xét chi phí đường đi.

**Chức năng**: Tìm đường đi nhanh chóng đến mục tiêu dựa trên heuristic, không đảm bảo tối ưu.

**Ưu điểm**:

Nhanh và hiệu quả trong các bài toán cần tốc độ hơn độ chính xác.

Yêu cầu bộ nhớ thấp hơn A\*.

Phù hợp với các bài toán như tìm kiếm gần đúng hoặc định hướng robot.

**A\* Search**

**Khái niệm**: A\* kết hợp chi phí đường đi (g) và heuristic (h) để ưu tiên trạng thái có tổng chi phí ước lượng thấp nhất (f = g + h).

**Chức năng**: Tìm đường đi ngắn nhất từ trạng thái ban đầu đến mục tiêu trong không gian tìm kiếm có trọng số, đảm bảo tối ưu nếu heuristic hợp lệ.

**Ưu điểm**:

Tối ưu và hoàn chỉnh với heuristic không vượt quá chi phí thực tế.

Hiệu quả hơn BFS và UCS trong không gian lớn.

Phù hợp với các bài toán như lập kế hoạch đường đi hoặc trò chơi.

**Iterative Deepening A\* (IDA\*)**

**Khái niệm**: IDA\* là phiên bản tiết kiệm bộ nhớ của A\*, sử dụng tìm kiếm theo chiều sâu với ngưỡng dựa trên hàm chi phí f = g + h, tăng ngưỡng khi cần.

**Chức năng**: Tìm đường đi tối ưu với bộ nhớ thấp hơn A\* trong không gian tìm kiếm lớn.

**Ưu điểm**:

Sử dụng ít bộ nhớ hơn A\* (giống DFS).

Đảm bảo đường đi tối ưu như A\*.

Phù hợp với các bài toán như 8-puzzle hoặc các không gian trạng thái phức tạp.

**Simple Hill Climbing**

**Khái niệm**: Simple Hill Climbing chọn trạng thái lân cận đầu tiên cải thiện giá trị heuristic so với trạng thái hiện tại.

**Chức năng**: Tìm lời giải gần tối ưu bằng cách leo lên "đồi" của hàm heuristic.

**Ưu điểm**:

Đơn giản và nhanh chóng.

Yêu cầu bộ nhớ thấp.

Phù hợp với các bài toán tối ưu hóa cục bộ hoặc tìm kiếm gần đúng.

**Steepest Ascent Hill Climbing**

**Khái niệm**: Steepest Ascent Hill Climbing chọn trạng thái lân cận có giá trị heuristic tốt nhất (thấp nhất) trong tất cả các lân cận.

**Chức năng**: Tìm lời giải gần tối ưu bằng cách chọn bước cải thiện lớn nhất.

**Ưu điểm**:

Hiệu quả hơn Simple Hill Climbing trong việc tránh cực trị cục bộ nhỏ.

Dễ triển khai và nhanh.

Phù hợp với các bài toán như tối ưu hóa hàm liên tục.

**Stochastic Hill Climbing**

**Khái niệm**: Stochastic Hill Climbing chọn ngẫu nhiên một trạng thái lân cận có heuristic tốt hơn, thay vì chọn trạng thái tốt nhất.

**Chức năng**: Tìm lời giải gần tối ưu, giảm nguy cơ bị kẹt ở cực trị cục bộ.

**Ưu điểm**:

Tăng khả năng thoát khỏi cực trị cục bộ so với Hill Climbing thông thường.

Nhanh và đơn giản.

Phù hợp với các bài toán có nhiều cực trị cục bộ.

**Simulated Annealing**

**Khái niệm**: Simulated Annealing mô phỏng quá trình làm nguội kim loại, cho phép chấp nhận trạng thái kém hơn với xác suất giảm dần theo thời gian.

**Chức năng**: Tìm lời giải gần tối ưu, tránh bị kẹt ở cực trị cục bộ.

**Ưu điểm**:

Hiệu quả trong việc thoát khỏi cực trị cục bộ.

Linh hoạt với các bài toán tối ưu hóa phức tạp.

Phù hợp với các bài toán như tối ưu hóa lịch trình hoặc thiết kế mạch.

**Beam Search**

**Khái niệm**: Beam Search giữ lại một số lượng cố định (beam width) các trạng thái tốt nhất dựa trên heuristic ở mỗi bước, loại bỏ các trạng thái kém hơn.

**Chức năng**: Tìm đường đi gần tối ưu với bộ nhớ giới hạn.

**Ưu điểm**:

Tiết kiệm bộ nhớ so với BFS hoặc A\*.

Nhanh và hiệu quả trong không gian lớn.

Phù hợp với các bài toán như dịch máy hoặc nhận dạng giọng nói.

**AND-OR Search**

**Khái niệm**: AND-OR Search xử lý các bài toán có nhiều cách để đạt mục tiêu (OR) và các bước bắt buộc (AND), thường dùng trong không gian tìm kiếm phức tạp.

**Chức năng**: Tìm lời giải trong các bài toán có cấu trúc logic phức tạp.

**Ưu điểm**:

Xử lý tốt các bài toán có ràng buộc logic.

Linh hoạt trong các bài toán như lập kế hoạch hoặc suy luận logic.

Có thể kết hợp với heuristic để tăng hiệu quả.

**Belief State Search**

**Khái niệm**: Belief State Search xử lý các bài toán có bất định, duy trì một tập hợp các trạng thái có thể xảy ra (belief states) và cập nhật dựa trên hành động.

**Chức năng**: Tìm đường đi trong môi trường không chắc chắn hoặc quan sát một phần.

**Ưu điểm**:

Phù hợp với các bài toán có thông tin không đầy đủ, như điều hướng robot.

Linh hoạt và mạnh mẽ trong môi trường động.

Có thể kết hợp với các thuật toán khác để tăng hiệu quả.

**Backtracking Search**

**Khái niệm**: Backtracking Search khám phá không gian trạng thái bằng cách thử từng nhánh và quay lại khi gặp ngõ cụt.

**Chức năng**: Tìm lời giải trong các bài toán ràng buộc hoặc tổ hợp.

**Ưu điểm**:

Đơn giản và dễ triển khai.

Hiệu quả trong các bài toán như Sudoku hoặc bài toán N-Queens.

Có thể kết hợp với heuristic để giảm không gian tìm kiếm.

**Genetic Algorithm**

**Khái niệm**: Genetic Algorithm mô phỏng quá trình tiến hóa, sử dụng quần thể các lời giải, lai ghép, đột biến, và chọn lọc để tìm lời giải tốt nhất.

**Chức năng**: Tìm lời giải gần tối ưu trong các bài toán tối ưu hóa phức tạp.

**Ưu điểm**:

Hiệu quả trong không gian tìm kiếm lớn và phức tạp.

Có khả năng thoát khỏi cực trị cục bộ.

Phù hợp với các bài toán như tối ưu hóa tham số hoặc thiết kế kỹ thuật.

**AC-3 Search**

**Khái niệm**: AC-3 (Arc Consistency) là thuật toán trong bài toán thỏa mãn ràng buộc (CSP), loại bỏ các giá trị không thỏa mãn ràng buộc để thu hẹp miền giá trị.

**Chức năng**: Giải các bài toán CSP như lập lịch hoặc bài toán tô màu đồ thị.

**Ưu điểm**:

Hiệu quả trong việc giảm không gian tìm kiếm trước khi tìm kiếm chính thức.

Dễ kết hợp với các thuật toán tìm kiếm khác.

Phù hợp với các bài toán có ràng buộc phức tạp.