TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÀI TẬP MÔN PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN
HOMEWORK #01: DFS/BFS/UCS for Sokoban

GV hướng dẫn: Lương Ngọc Hoàng

Người thực hiện: Lê Quý Nhân. MSSV: 22520999

TPHCM, ngày 20 tháng 3 năm 2024

1 Giới Thiệu:

Sokoban là trò chơi trí tuệ phổ biến, mô phỏng việt di chuyển thùng hàng trong kho. Mục tiêu là di chuyển tất cả các thùng hàng đến vị trí đích đã chỉ định.

2 Mô Hình Hóa Sokoban:

Trò chơi được khởi tạo với rất nhiều *map* và các *"level"*. Các kí hiệu tượng trưng cho các vật thể: "#" là ***tường***, "B" là ***thùng***, "." là ***đích đến***, " " là ***khoảng trống*** có thể đi được và cuối cùng "&" là ***vị trí khởi tạo ban đầu*** của Sokoban

Trạng thái kết thúc là trạng thái khi các ***thùng*** (B) trùng với ***đích đến*** (.) thì trò chơi kết thúc.

Không gian trạng thái là tập hợp tất cả các trạng thái có thể xảy ra trong trò chơi.

Có 4 hành động hợp lệ trong Sokoban đó là: *đi lên*, *đi xuống*, *qua phải* và *qua trái*.

Tuy nhiên để cho chi tiết hơn, có tận 8 hành động hợp lệ là: *đi lên*, *đi xuống*, *qua phải* và *qua trái*, *đẩy thùng lên*, *đẩy thùng xuống*, *đẩy thùng qua trái*, *đẩy thùng qua phải*.

Hàm tiến triển (successor function) là một hàm giúp ta cập nhật lại trạng thái hiện tại khi thực hiện một hành động hợp lệ.

3 DFS, BFS, UCS

3.1 Tổng quan về 3 loại Uninformed Search

DFS (Depth-First Search):
Duyệt theo chiều sâu, ưu tiên khám phá các đỉnh con trước. Sử dụng cấu trúc dữ liệu Stack để lưu trữ các đỉnh cần khám phá.

BFS (Breadth-First Search):
Duyệt theo chiều rộng, khám phá tất cả các đỉnh cùng độ sâu trước. Sử dụng cấu trúc dữ liệu Queue để lưu trữ các đỉnh cần khám phá.

UCS (Uniform Cost Search):

Duyệt theo chi phí đồng nhất, ưu tiên đường đi có chi phí thấp nhất. Sử dụng cấu trúc dữ liệu Priority Queue để lưu trữ các đỉnh cần khám phá, sắp xếp theo chi phí đường đi.

3.2 Bảng thống kê bước đi

Level	DFS	BFS	UCS
1	79	12	12
2	24	9	9
3	403	15	15
4	27	7	7
5	55	20	20
6	Lớn	19	19
7	Lớn	21	21
8	Lớn	97	97
9	Lớn	8	9
10	Lớn	33	33
11	Lớn	34	34
12	Lớn	23	23
13	Lớn	31	31
14	Lớn	23	23
15	Lớn	105	105
16	Lớn	34	34

Ta có thể thấy rằng **BFS** và **UCS** có thể tìm ra đường đi ngắn nhất.
Với không gian trạng thái nhỏ, **DFS** khám phá rất nhanh nhưng lại dễ rơi vào ngõ cụt, với **BFS** thì có thể tìm đường đi ngắn nhất nhanh nhất nhưng lại có giới hạn về bộ nhớ, còn **UCS** luôn có thể tìm ra đường đi tối ưu (không phải hoàn toàn là ngắn nhất) nhưng tốc độ có thể chậm hơn **BFS**.
Với không gian trạng thái lớn, **DFS** có thể áp dụng được nhưng dễ rơi vào vòng lặp vô hạn, **BFS** sẽ tốn rất nhiều bộ nhớ nên ít khi khả dụng, **UCS** có thể tìm ra đường đi tối ưu mặc dù có thể tốn thời gian và bộ nhớ.

Vậy ta có thể nói UCS là thuật toán tối ưu hơn DFS và BFS vì nó trung hòa được vấn đề của cả hai thuật toán.

Tình huống	Thuật toán phù hợp
Không gian trạng thái nhỏ	DFS, BFS, UCS
Tìm đường đi ngắn nhất nhanh nhất	BFS
Tìm đường đi tối ưu	UCS
Không gian trạng thái lớn	UCS