ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP 1

MÔN: TRÍ TUỆ NHÂN TẠO - CS106.O22

THUẬT TOÁN DFS, BFS, UCS ĐỂ CHƠI GAME SOKOBAN

Giảng viên hướng dẫn: **TS. Lương Ngọc Hoàng**

Sinh viên thực hiện: Tăng Gia Hân - 22520394

TP. Hồ Chí Minh, tháng 3, năm 2024

MỤC LỤC

I. GIỚI THIỆU BÀI TOÁN

Game Sokoban là một trò chơi logic có tính năng giải đố và là một trong những trò chơi cổ điển và phổ biến trong thể loại này. Trong trò chơi này, người chơi điều khiển một người giữ kho và cố gắng đẩy các hộp đến các vị trí dự kiến. Thiết kế các bước di chuyển tối thiểu và làm thế nào để tránh ngõ cụt là hai yếu tố bắt buộc quyết định độ khó của trò chơi.

II. MÔ HÌNH HÓA BÀI TOÁN

Trò chơi sokoban có thể được coi là một bài toán tìm kiếm, chúng ta chủ yếu tìm kiếm các hộp và vị trí lưu trữ. Vì vậy, nó có trạng thái bắt đầu và trạng thái kết thúc hợp lệ. Trạng thái bắt đầu là trạng thái được đưa ra bởi nhà phát triển trò chơi gốc, bao gồm vị trí của người chơi, tường và các hộp trên bản đồ. Trạng thái kết thúc là trạng thái khi tất cả các hộp được chuyển đến vị trí lưu trữ thích hợp.

Để mô hình hóa trò chơi này, có một ký hiệu tiêu chuẩn được sử dụng để phân biệt độc lập và rõ ràng giữa các đối tượng trong trò chơi. Đầu vào cho mô hình bao gồm các ký tự sau: “#”, “ ”, “B”, “&”, “.” và “X”. Mỗi kí tự là một thuộc tính đặc biệt của trò chơi được gán cho nó: “#” là một bức tường, “ ” là một không gian trống, “B” là một hộp, “&” là người chơi, “.” là một vị trí mục tiêu và “X” là một hộp được đặt trên một mục tiêu.

Không gian trạng thái là tập hợp tất cả các trạng thái có thể của trò chơi, bao gồm vị trí của người chơi và các hộp trên bản đồ. Các hành động hợp lệ bao gồm di chuyển của người chơi theo chiều ngang hoặc dọc (bốn hướng - Lên, Xuống, Trái và Phải). Người chơi có thể đẩy tối đa một hộp vào một không gian trống không phải là tường hoặc hộp khác. Người chơi cũng không được phép kéo các hộp. Số lượng hộp và vị trí mục tiêu bằng nhau và người chơi thành công khi tất cả các hộp đều ở vị trí lưu trữ mục tiêu.

Hàm tiến triển (successor function): hàm này nhận vào vị trí hiện tại của người chơi và hộp trên bản đồ, sau đó sinh ra các hành động hợp lệ mà người chơi có thể thực hiện. Các hành động bao gồm di chuyển (Move - 'M') hoặc đẩy hộp (Push - 'P') theo các hướng lên,

xuống, trái, phải. Hàm sẽ trả về danh sách các cặp hành động và hướng di chuyển tương ứng.

III. CÁC THUẬT TOÁN TÌM KIẾM CHO SOKOBAN

3.1 Breadth First Search (BFS)

Breadth First Search (BFS) là một giải thuật vét cạn, dùng để duyệt qua tất cả các trạng thái di chuyển có thể xảy ra khi áp dụng vào Sokoban để tìm tới trạng thái mục tiêu. Chúng ta sẽ từ trạng thái ban đầu, sinh ra tối đa 4 trạng thái với các nước đi của người chơi tiếp theo có thể đi. Từ đó tạo thành một cấu trúc cây. Cuối cùng dùng BFS để vét cạn cây chắc chắn sẽ tìm được trạng thái mục tiêu.

Level	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Steps	12	9	15	7	20	19	21	97	8	33	34	23	31	23	105	34	-	-
Time(s)	0.07	0.01	0.15	0.01	170.66	0.01	0.76	0.18	0.01	0.02	0.02	0.08	0.13	2.43	0.24	20.77	-	-

Bảng 1: Bảng thống kê độ dài đường đi và thời gian thực thi của BFS

Khi áp dụng thuật toán BFS vào trò chơi Sokoban với bảng thống kê kết quả như trên, chúng ta có thể nhận xét như sau:

- BFS thường cho ra đường đi ngắn nhất trên các bản đồ Sokoban đơn giản và kích thước nhỏ.
- Trong trường hợp của level 17, thuật toán không đưa ra được lời giải. Điều này có thể được giải thích bởi sự phức tạp của bản đồ, khi có nhiều hộp cần di chuyển và các đường đi có thể bị chặn.
- Trong trường hợp của level 18, thời gian thực thi của BFS trở nên rất lớn và có thể gây ra vấn đề về hiệu suất hoặc thậm chí không thể hoàn thành trong thời gian hợp lý. Việc áp dụng BFS vào Sokoban ở level này có thể dễ dàng dẫn đến việc sử dụng quá nhiều bộ nhớ và vượt quá giới hạn bộ nhớ có sẵn.

3.2 Depth First Search (DFS)

Tương tự như BFS, Depth First Search (DFS) bắt đầu tìm kiếm tại một nút gốc và thay vì duyệt qua tất cả các trạng thái di chuyển có thể xảy ra theo chiều rộng, nó duyệt theo chiều sâu. Khi duyệt, tiếp tục đi sâu vào một nhánh của cây cho đến khi không thể đi tiếp nữa hoặc tìm thấy trạng thái mục tiêu. Khi tìm thấy trạng thái mục tiêu, thuật toán DFS kết thúc và trả về đường đi từ trạng thái ban đầu đến trạng thái mục tiêu. Trong thực tế, DFS dẫn đến tìm kiếm nhanh hơn nhiều so với BFS. Tuy nhiên, nó thường tạo ra một giải pháp dài có chứa nhiều bước di chuyển.

Level	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Steps	79	24	403	27	-	55	707	323	74	37	36	109	185	865	291	-	-	-
Time(s)	0.05	0.01	0.19	0.00	-	0.01	0.48	0.07	0.24	0.02	0.02	0.12	0.16	3.43	0.15	-	-	-

Bảng 2: Bảng thống kê độ dài đường đi và thời gian thực thi của DFS

Khi áp dụng thuật toán DFS vào trò chơi Sokoban với bảng thống kê kết quả như trên, chúng ta có thể nhận xét như sau:

- Số bước đi thường khá lớn so với BFS, cho thấy DFS có thể tạo ra đường đi không phải là tối ưu.
- Thời gian thực thi của DFS thường không quá lớn, cho thấy thuật toán có hiệu suất tốt trên các bản đồ Sokoban đơn giản và kích thước nhỏ.
- Tuy nhiên, DFS có thể gặp vấn đề khi đối mặt với các bản đồ phức tạp hoặc có quá nhiều trạng thái trạng thái khả thi, như trong trường hợp của level 5, 16, 17 và 18, khi không có kết quả hoặc cần nhiều thời gian hơn để tính toán.

3.3 Uniform Cost Search (UCS)

UCS là một biến thể của BFS, nhưng thay vì sử dụng chi phí cố định cho mỗi bước di chuyển, UCS sử dụng chi phí thực tế của từng bước. Trong trò chơi Sokoban, chi phí có thể được định nghĩa là số bước đi để di chuyển một hộp mà không gây ra việc đẩy. Sử dụng một hàng đợi ưu tiên để lưu trữ các trạng thái của trò chơi. Mỗi phần tử trong hàng đợi ưu tiên là một danh sách, đại diện cho một trạng thái của trò chơi. Độ ưu tiên của mỗi phần tử được xác định bằng hàm chi phí. Tại mỗi bước, thuật toán UCS lấy ra trạng thái đầu tiên từ hàng đợi ưu tiên và mở rộng nó bằng cách xem xét các hành động có thể thực hiện được từ trạng thái hiện tại. Sau mỗi bước mở rộng, UCS kiểm tra xem trạng thái hiện tại có phải là trạng thái mục tiêu hay không. Nếu là trạng thái mục tiêu, thuật toán dừng lại và trả về đường đi ngắn nhất đã tìm thấy.

Level	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Steps	12	9	15	7	20	19	21	99	8	33	34	23	31	23	105	34	-	-
Time(s)	0.04	0.01	0.08	0.01	49.45	0.02	0.48	0.18	0.01	0.02	0.02	0.07	0.15	2.51	0.25	13.06	-	-

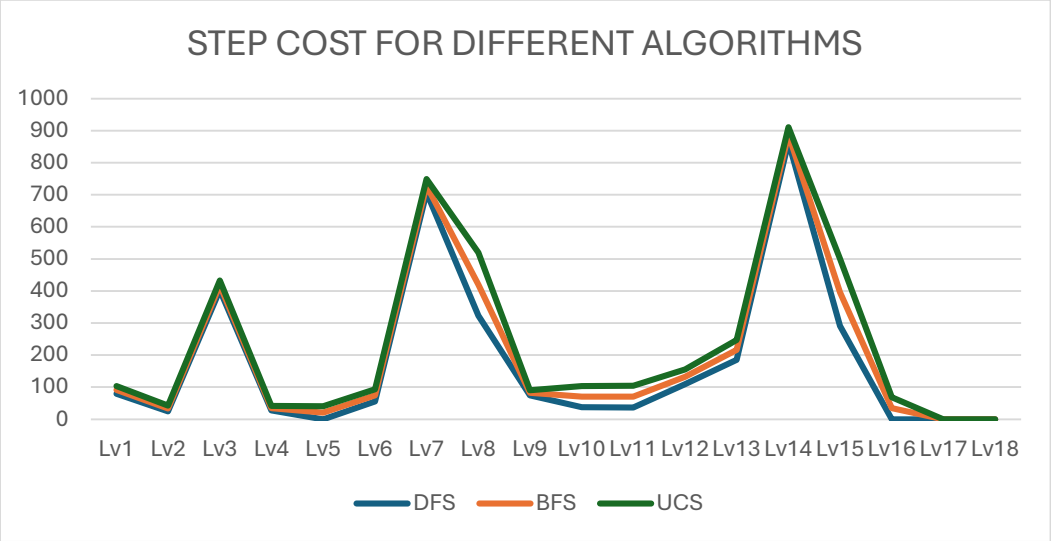
Bảng 3: Bảng thống kê độ dài đường đi và thời gian thực thi của UCS

Khi áp dụng thuật toán UCS vào trò chơi Sokoban với bảng thống kê kết quả như trên, chúng ta có thể nhận xét như sau:

- UCS đã tìm ra đường đi thành công trên hầu hết các level, nhưng không có kết quả cho level 17 và 18.
- Số bước đi của UCS thường tương đương hoặc gần giống với BFS, cho thấy UCS cũng có khả năng tạo ra các đường đi gần tối ưu.
- Thời gian thực thi của UCS tương đối nhỏ so với BFS trên các bản đồ Sokoban đơn giản và kích thước nhỏ.
- Tuy nhiên, tương tự như DFS, UCS cũng có thể gặp vấn đề khi đối mặt với các bản đồ phức tạp hoặc có quá nhiều trạng thái trạng thái khả thi, như trong trường hợp của level 17 và 18.

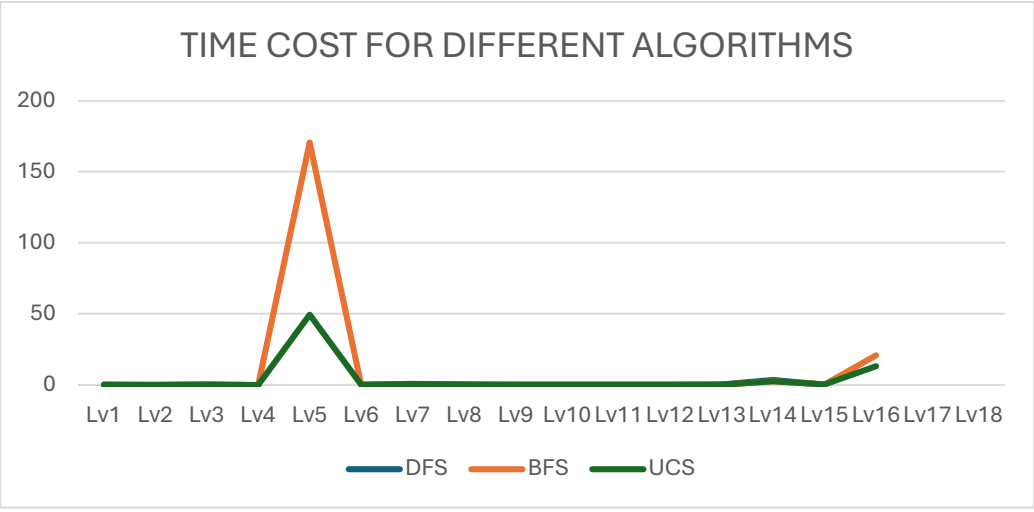
IV. PHÂN TÍCH HIỆU XUẤT

Trong quá trình áp dụng các thuật toán DFS, BFS và UCS vào game Sokoban, chúng ta sử dụng hai chỉ số chính để đánh giá hiệu suất của mỗi thuật toán: tổng số bước đi và thời gian thực thi. Đầu tiên, chúng ta sẽ so sánh số bước đi mà mỗi thuật toán tìm ra được. Sau đó, ta sẽ đối chiếu thời gian mà mỗi thuật toán tốn để tìm kiếm lời giải.



Hình 1: Biểu đồ độ dài đường đi của các thuật toán DFS BFS UCS

Biểu đồ 1 này biểu diễn số bước đi tìm được bởi mỗi thuật toán trên các level của trò chơi Sokoban. Từ biểu đồ, ta có thể thấy rằng BFS và UCS thường cho ra giải pháp có số bước ít nhất, trong khi DFS thường có số bước đi lớn hơn so với hai thuật toán còn lại. Điều này cho thấy BFS và UCS có khả năng tạo ra giải pháp gần tối ưu hơn, trong khi DFS có thể tạo ra giải pháp không tối ưu và dài hơn.



Hình 2: Biểu đồ chi phí thời gian cho các thuật toán DFS BFS UCS

Biểu đồ 2 này so sánh thời gian thực hiện của mỗi thuật toán trong quá trình tìm kiếm lời giải trên các level của trò chơi Sokoban. Từ biểu đồ, ta thấy rằng thời gian thực hiện của DFS thường thấp hơn so với BFS và UCS, đặc biệt là trên các bản đồ phức tạp và lớn.

V. KẾT LUẬN

Trong quá trình nghiên cứu và áp dụng các thuật toán DFS, BFS và UCS vào trò chơi Sokoban, chúng ta đã thấy rằng mỗi thuật toán có những ưu và nhược điểm riêng. Tuy nhiên, khi đánh giá hiệu suất của từng thuật toán trên các level của trò chơi, có thể kết luận như sau:

- BFS và UCS thường cho ra giải pháp gần tối ưu với số bước đi ít nhất trên hầu hết các level của trò chơi Sokoban. Điều này cho thấy rằng BFS và UCS thích hợp cho việc tìm kiếm giải pháp tối ưu trong trò chơi này.
- DFS có thể tạo ra giải pháp không tối ưu và có số bước đi lớn hơn so với hai thuật toán còn lại. Tuy nhiên, DFS lại có thời gian thực hiện thấp hơn, đặc biệt là trên các bản đồ phức tạp.

Về mặt khó khăn trong giải quyết, có thể nhận thấy rằng bản đồ khó giải nhất là bản đồ có index là 18. Bản đồ này phức tạp với nhiều vật cản và các hộp cần di chuyển, đồng thời có một không gian trống lớn giữa các hộp và mục tiêu, điều này tạo ra nhiều tùy chọn di chuyển. Trạng thái ban đầu đã tạo ra một môi trường khó khăn, với nhiều hộp đặt ở các vị trí khác nhau và cần phải di chuyển đến các vị trí mục tiêu phức tạp. Điều này làm tăng độ phức tạp của bài toán và khiến cho các thuật toán tìm kiếm lời giải mất nhiều thời gian hoặc thậm chí không tìm được giải pháp trong một khoảng thời gian hợp lý.

Tổng kết lại, dựa trên kết quả thử nghiệm và phân tích, thuật toán tốt nhất cho trò chơi Sokoban là BFS hoặc UCS, tùy thuộc vào yêu cầu về thời gian thực hiện và số bước đi tối ưu. Đồng thời, cần chú ý đến độ phức tạp của các bản đồ để đánh giá và lựa chọn thuật toán phù hợp.