BÁO CÁO: ÁP DỤNG THUẬT TOÁN TÌM KIẾM DFS, BFS VÀ UCS VÀO GAME SOKOBAN

Giảng viên: Lương Ngọc Hoàng

I. Giới thiệu

Sokoban là trò chơi dạng câu đố trong đó người chơi phải đẩy một số khối vuông vượt qua chướng ngại vật để đến đích.

Một số lưu ý của trò chơi:

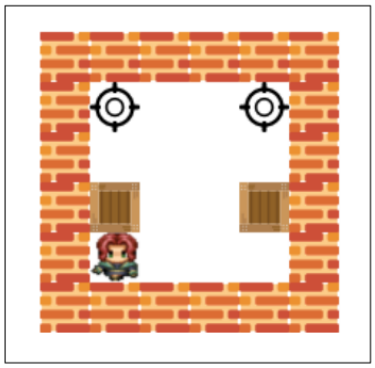
- Trò chơi có dạng bảng ô vuông. Có một số khối vuông được đẩy đến đích (số ô đích đúng bằng số khối vuông). Chỉ có thể đẩy từng khối vuông một, và không thể kéo, cũng như không thể đẩy một dãy hai hay nhiều khối.
- Khối vuông bị dính tường nếu như nó bị đẩy sát vào tường mà hai bên tường đều là góc. Vì không thể kéo khối lại được nên coi như khối này bị mất, nó không thể được đưa đến đích trừ phi đích nằm đúng trên cạnh tường đó. Dính tường là một trường hợp cần tránh khi chơi.

-Trích Wikipedia

Mục tiêu của ta là sẽ thực thi một số thuật toán tìm kiếm đường đi gồm: Depth First Search (DFS), Breadth First Search (BFS), Uniform Cost Search (USC) để xem cách các giải thuật này sẽ giải trò chơi và so sánh độ tối ưu của các giải thuật.

II. Mô hình hóa

- Trạng thái bắt đầu (startingState): Vị trí đầu tiên của nhân vật và các thùng.



Hình 1: Trạng thái mở đầu của game (Level 2)

- Trạng thái kết thúc (endState): là trạng thái khi mà tất cả các thùng được di chuyển đến vị trí mục tiêu.
- Không gian trạng thái game (gameState): tất cả các vị trí của các hộp và các vị trí mà nhân vật có thể đi trên bản đồ.
- Hàm tiến triển:
    - PosOfPlayer(gameState): trả về vị trí của nhân vật
    - PosOfBoxes(gameState): trả về vị trí của các thùng.
    - PosOfWalls(gameState): trả về vị trí của tường.
    - PosOfGoals(gameState): trả về vị trí của các đích.
    - isEndState(gameState): trả về true nếu toàn bộ thùng đã ở vị trí mục tiêu.
    - isLegalAction(action, posPlayer, posBox): nhận đầu vào là bước đi tiếp theo của nhân vật, vị trí hiện tại của nhân vật và các thùng; trả về true nếu bước đi hợp lệ.
    - legalActions(posPlayer, posBox): nhận đầu vào là vị trí hiện tại của nhân vật và các thùng, trả về tất cả các bước đi hợp lệ.
    - updateState(posPlayer, posBox, action): nhận đầu vào là vị trí hiện tại của nhân vật và các thùng, bước đi tiếp theo của nhân vật; trả về vị trí mới của nhân vật và các thùng.
    - isFailed(posBox): nhận đầu vào là vị trí của các thùng, trả về True nếu vị trí hợp lệ.
    - cost(actions): nhận đầu vào là các hành động của nhân vật, trả về số bước đi không phải bước đẩy thùng.

III. Đánh giá:

Ta có bảng thống kê dưới đây, thông tin trong bảng bao gồm thời gian để tìm ra giải thuật và số bước đi (cost) của các thuật toán Depth First Search (DFS), Breadth First Search (BFS), Uniform Cost Search (UCS) lần lượt theo thứ tự (thời gian - runtime để tìm ra giải thuật có tính tương đối tùy theo bộ xử lý của máy)

Giải thuật	DFS		BFS		UCS	
Level	Steps	Time	Steps	Time	Steps	Time
1					12	0.178
2	24	0.016	9	0.019	9	0.014
3	403	0.62	15	0.495	15	0.256
4	27	0.14	7	0.021	7	0.009
5					20	196.6562
6	55	0.46	19	0.045	19	0.034
7	707	1.6291	21	2.6212	21	1.7601
8	323	0.228	97	0.618	97	0.674
9	74	0.785	8	0.036	8	0.039
10	37	0.059	33	0.056	33	0.053
11	36	0.075	34	0.081	34	0.064
12	109	0.462	23	0.273	23	0.284
13	185	0.536	31	0.536	31	0.581
14	865	11.7537	23	8.2235	23	9.5155
15	291	0.497	105	0.9051	105	0.8921
16			34	63.2646	34	50.4579

Nhận xét:

- Nhìn chung, các giải thuật BFS, DFS, UCS đều có thể tìm ra giải pháp cho các bài toán với điều kiện bài toán có lời giải với số bước đi không quá lớn. Và tùy vào điều kiện của từng bài toán mà các giải thuật sẽ có độ tối tối ưu khác nhau.
- Với bài toán level 5, thoạt nhìn bài toán có vẻ đơn giản với con người – nhìn vào là nghĩ ra ngay bước đi đến mục tiêu, song việc có nhiều ô trống (space) được map cung cấp dẫn đến có nhiều bước đi hợp lệ (legalActions) làm agent bị distract và mất thời gian thử hết tất cả các bước có thể đi để đến goal.

- Dễ thấy số bước đi của DFS thường lớn hơn 2 thuật toán còn lại, dù cùng tìm ra được giải thuật, song thời gian để tìm ra thuật toán của DFS có khi nhỏ hơn (lv 2,4,7,8,11,15) cũng có khi lại lớn hơn so với BFS.
  ⇨ Lý giải: Điều này là do tính chất đào "sâu" của DFS, cách hoạt động của DFS đó là duyệt đi xa nhất theo nhánh cho đến khi tìm thấy mục tiêu hoặc lùi về từng đỉnh, dẫn tới số bước đi (cost) lớn bất thường so với 2 thuật toán còn lại (BFS duyệt theo chiều rộng và UCS duyệt theo nhánh có chi phí thấp nhất) và thời gian tìm ra giải thuật mang tính "may rủi" khi vô tình đào đến node mục tiêu.
- Chi phí của giải thuật tìm ra bởi UCS và BFS ngang nhau, song thời gian tìm ra giải thuật của UCS nhìn chung nhỏ hơn hoặc ngang với BFS. Đặc biệt trong bài toán 5 thì UCS có thể tìm ra giải thuật bằng với bộ xử lý và lưu trữ thông tin tính toán của máy.
  ⇨ Lý giải: UCS duyệt theo nhánh có chi phí nhỏ nhất (trong bài toán là các bước không đẩy thùng), cách hoạt động như vậy giúp giải thuật tập trung vào những bước có chi phí thấp (focus vào những hướng đi có di chuyển thùng) dẫn đến dễ dàng đi đến mục tiêu hơn.

Kết luận: trong 3 thuật toán tìm kiếm, thuật toán UCS tối ưu nhất.