ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÀI TẬP 1

TRÍ TUỆ NHÂN TẠO

Sinh viên: Đỗ Quang Lực - 23520902

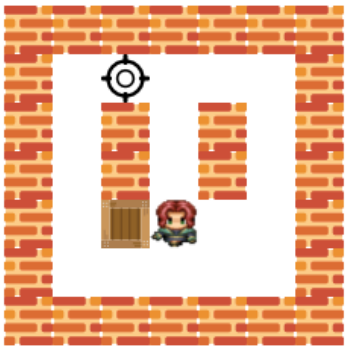
Giáo viên hướng dẫn: Lương Ngọc Hoàng

Ngày 23 tháng 3 năm 2025

Mục lục

1 Giới thiệu về Sokoban

Sokoban là một trò chơi giải đố kinh điển xuất hiện từ năm 1982, trong đó người chơi điều khiển một nhân vật (thường là một người đẩy hàng) để di chuyển các thùng vào các vị trí đích trên bản đồ.



Hình 1: Game Sokoban

Luật chơi cơ bản:

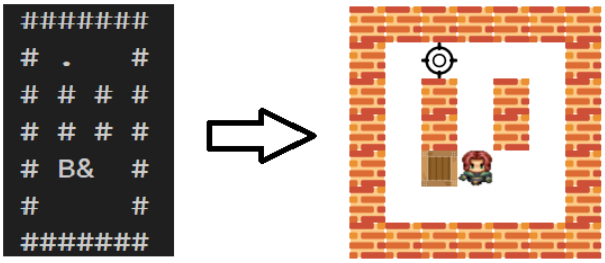
- Người chơi chỉ có thể đẩy hộp, không thể kéo.
- Nếu một hộp bị đẩy vào góc (không có đường để đẩy ra) thì nó sẽ bị kẹt vĩnh viễn.
- Mục tiêu là đưa tất cả các hộp vào đúng vị trí đích.

1.1 Yêu cầu của bài tập

Hãy cài đặt thuật toán BFS và UCS để chơi game Sokoban, viết báo cáo trình bày lại việc áp dụng thuật toán tìm kiếm DFS, BFS, UCS cho Sokoban.

2 Mô hình hóa Sokoban

Để dễ lập trình hơn, trò chơi Sokoban được mô hình hóa như sau:



Hình 2: Mô hình hóa Sokoban

Bản đồ của trò chơi được biểu diễn dưới dạng mảng hai chiều, mỗi phần tử của mảng chứa giá trị (có thể là số hoặc ký tự) để thể hiện các thành phần của bản đồ, ví dụ như:

- ' ' (ký tự khoảng trắng): Ô trống
- '#': Tường
- '&': Nhân vật
- '.': Ô mục tiêu
- 'B': Hộp
- 'X': Hộp ở trên ô mục tiêu

2.1 Trạng thái ban đầu, trạng thái kết thúc

Xét trên từng màn chơi ứng với từng bản đồ, một trạng thái của trò chơi bao gồm:

- $posPlayer$: là vị trí của nhân vật nằm trên bản đồ
- $posBoxes$: là vị trí của các hộp nằm trên bản đồ.

Ta thường biểu diễn trạng thái dưới dạng tuple: $(posPlayer, posBoxes)$.

- **Trạng thái ban đầu của trò chơi:** là vị trí ban đầu của nhân vật và vị trí của các hộp: $(beginPosPlayer, beginPosBoxes)$.
- **Trạng thái kết thúc của trò chơi:** là trạng thái mà tất cả vị trí của các hộp(posBoxes) trùng với tất cả vị trí của các ô mục tiêu (posGoals).

2.2 Không gian trạng thái

Là tập hợp các trạng thái có thể có trong quá trình chơi, từ **trạng thái ban đầu** cho đến **trạng thái kết thúc**.

2.3 Các hành động hợp lệ

Là những hành động mà người chơi có thể thực hiện từ trạng thái hiện tại mà không vi phạm luật của trò chơi.
Đối với trò chơi Sokoban, người chơi có thể thực hiện hai loại hành động chính:

- **Di chuyển:** đi đến một ô trống mà không đẩy hộp.
- **Đẩy hộp:** Đẩy một hộp đến một ô trống nếu hộp đó có thể di chuyển.

Trong file solver.py có hàm legalActions(posPlayer, posBox) sẽ đưa ra các hành động hợp lệ cho nhân vật ở vị trí posPlayer và tập hợp vị trí các hộp ở posBox.

2.4 Hàm tiến triển (Successor Function)

Là một hàm nhận vào trạng thái hiện tại của trò chơi, hành động cần thực hiện rồi trả về trạng thái mới sau khi thực hiện hành động đó (bao gồm việc cập nhật vị trí của nhân vật và các hộp trên bản đồ).
Trong file solver.py, hàm updateState(posPlayer, posBox, action) chính là hàm tiến triển, giúp trả về trạng thái mới của posPlayer, posBox sau khi thực hiện action.

2.5 Cài đặt chương trình

Tất cả source code đều được lưu trữ trên link github: BT1 - DFS/BFS/UCS for Sokoban.

3 Bảng thống kê

Dưới đây là bảng thống kê về thời gian chạy cũng như số bước di chuyển của cả 3 thuật toán DFS, BFS và UCS.

Màn chơi	DFS		BFS		UCS	
	Bước đi	Thời gian (s)	Bước đi	Thời gian (s)	Bước đi	Thời gian (s)
1	0.05809	79	0.08659	12	0.05859	12
2	0.00252	24	0.00556	9	0.00457	9
3	0.22396	403	0.17571	15	0.08544	15
4	0.00099	27	0.00553	7	0.00304	7
5	...	...	182.1518	20	75.59896	20
6	0.01061	55	0.01113	19	0.01218	19
7	0.55005	707	0.87170	21	0.58656	21
8	0.07660	323	0.20155	97	0.23214	97
9	0.27159	74	0.00856	9	0.00758	9
10	0.01205	37	0.01545	33	0.01653	33
11	0.01602	36	0.01615	24	0.01807	24
12	0.13907	109	0.08774	23	0.09802	23
13	0.18910	185	0.15961	31	0.21101	31
14	3.98178	865	2.87621	23	3.41972	23
15	0.16626	291	0.27285	105	0.33353	105
16	3274.62036	86	22.5776	34	17.67599	34
17	0.51311	877	...	...	...	...
18	...	...	...	...	...	...

Bảng 1: So sánh DFS, BFS và UCS

Chú thích: những ô ... là những ô mà chạy thuật toán không tìm ra được lời giải.

4 Nhận xét

4.1 Về thuật toán và độ dài đường đi

- Thuật toán DFS hầu hết đưa ra lời giải tìm kiếm nhanh tuy nhiên độ dài đường đi thì rất dài vì thuật toán tìm kiếm lời giải ở các tầng rất sâu (Màn chơi 14 chỉ cần 23 bước là giải được nhưng thuật toán DFS lại đưa ra lời giải với 865 nước).

- Thuật toán BFS tìm kiếm lời giải ở các tầng thấp trước nên việc tìm kiếm sẽ khó khăn khi lời giải có số bước đi lớn hoặc có nhiều nước đi vô nghĩa. Mặt khác thì lời giải đưa ra chắc chắn sẽ là lời giải với độ dài đường đi ngắn nhất.

- Thuật toán UCS với hàm chi phí là số nước đi không dịch chuyển hộp, thuật toán với mong muốn sẽ ưu tiên tìm kiếm lời giải ít có các nước đi vô nghĩa, độ dài đường đi UCS tìm được cũng sẽ là đường đi với độ dài ngắn nhất và bằng độ dài của BFS.

4.2 Về các bản đồ trò chơi

- Màn 18 là màn khó nhất với kích thước bảng, số lượng lớn, và đáp án cũng ở tầng khá sâu. Cả 3 thuật toán đều không đưa ra lời giải được do vấn đề về bộ nhớ của máy tính.

- Mản 5 là màn gây khó đối với thuật toán DFS, vì có rất nhiều ô trống, số bước hợp lệ cho mỗi trạng thái nhiều hơn các màn khác nên DFS không đưa ra được lời giải. Mặt khác vì độ dài đường đi của lời giải ngắn nên BFS và UCS đều có thể tìm ra lời giải, trong đó UCS đưa ra lời giải nhanh hơn rất nhiều do ưu tiên các lời giải có hàm chi phí thấp.

- Màn 16 là màn cho thấy sự hiệu quả của thuật toán BFS và UCS với những màn chơi có số nước đi của lời giải ngắn, khi chỉ mất khoảng 17-22s là tìm được lời giải, trong khi đó DFS phải mất gần 55 phút.

- Màn 17 là màn cho thấy sự hiệu quả của thuật toán DFS với những màn có số nước đi của lời giải lớn, khi chỉ mất 0.5s để tìm ra lời giải, trong khi đó BFS và UCS không tìm ra được lời giải.

4.3 Kết luận

- Thuật toán DFS sẽ hoạt động tốt hầu hết đối với các màn chơi, đặc biệt với những màn chơi có độ dài đường đi lớn, nếu may mắn đi đúng hướng thì sẽ tìm được lời giải rất nhanh.

- BFS và DFS sẽ hoạt động tốt hơn ở những màn chơi có độ dài đường đi ngắn. Ở một vài màn chơi BFS có đưa ra lời giải nhanh hơn ÚC nhưng không đáng kể, tuy nhiên ở những màn chơi lớn (5, 16) UCS cho thấy sự hiệu quả hơn BFS rất nhiều.