**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

🙤🙧🟍🙥🙦

**Logo

Description automatically generated**

**BÁO CÁO THỰC HÀNH**

**MÔN C106.M21: TRÍ TUỆ NHÂN TẠO**

**LAB: Assignment 1 - BFS/DFS/UCS for Sokoban**

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

**Sinh viên thực hiện:**

**Họ và tên** **MSSV**

1. Trương Thế Tấn 19522180

**TP. Hồ Chí Minh, tháng 3, năm 2022**

1. **Mô hình hóa Sokoban**

Game Sokoban là một mô hình yêu cầu người chơi đẩy các thùng/hộp vào các kho lưu trữ được sắp xếp sẵn. Việc sử dụng AI nhằm phát triển một tác nhân (agent) AI để giải quyết trò chơi Sokoban với các thuật toán DFS, BFS và UCS.

Sokoban mô hình hóa bằng các phân tích game thành các hàm chức năng tương ứng, trong đó các hàm chức năng dựa vào các quy luật di chuyển trong game Sokoban. Các hàm chức năng được kết nối với nhau để được mô hình của hệ thống.

|  |  |
| --- | --- |
| 1. **Xác định trạng thái** | 1. **Xác định vấn đề** |
| * *Hành động (action)*: thiết lập hành động nào có thể thực hiện ở một trạng thái nhất định * *Chi phí (cost)*: chi phí (một số nguyên) để thực hiện một hành động để đạt được một trạng thái nhất định * *Thông tin bản đồ*: vị trí hiện tại của người chơi (agent) và các hộp (boxes). Các ký hiệu định nghĩa các đối tượng trong bản đồ như sau: * ‘#’: tường * ‘&’: người chơi (agent) * ‘.’: Khu vực lưu trữ   Ví dụ:   |  |  | | --- | --- | |  |  | | * *Trạng thái khởi đầu (start state)*: action = “bắt đầu” ([0]), cost = 0, các thông tin bản đồ (vị trí người chơi, hộp, tường,…) * *trạng thái kết thúc (end state)*: tất cả các hộp được chuyển đến kho lưu trữ * isEndState(posBox) * *không gian trạng thái (state space)*: tập hợp tất cả các trạng thái có thể xảy ra. * Ví dụ: người chơi di chuyển lên, xuống, trái, phải, đẩy thùng lên, xuống, trái, phải, đẩy thùng vào kho lưu trữ. * *các hành động hợp lệ:* các hành động có thể xảy ra * lên (u), xuống (d), trái (l), phải (r) * Đẩy thùng lên (U), xuống (D), trái (L), phải (R) * *hàm tiến triển (successor function)*: trạng thái sau khi di chuyển hợp lệ * updateState(posPlayer, posBox, action) |

1. **Áp dụng DFS, BFS, UCS**
2. DFS

frontier : lưu vị trí người chơi và vị trí các hộp

actions : lưu các hành động đã thực hiện

exploredSet : tập hợp các vị trí người chơi và vị trí các hộp đã đi qua

Hình 1. Cơ chế LIFO cho frontier và actions

frontier.pop()

actions.pop()

frontier.append(node+[(newPosPlayer,newPosBox)])

actions.append(node\_action + [action[-1]])

A picture containing diagram

Description automatically generated

Ở mỗi lần lặp sẽ kiểm tra trạng thái kết thúc với hàm isEndState(node[-1][-1]) (node[-1][-1] là vị trí các hộp).

Nếu chưa kết thúc, chương trình sẽ kiểm tra vị trí của người chơi và các hộp đã có trong exploredSet, không đi lại vị trí cũ.

Hàm legalActions trả về các hành động hợp lệ mà người chơi có thể thực hiện. Hàm updateState sẽ cập nhật các vị trí mới dựa vào các hành động được trả về từ hàm legalActions. Hàm isFailed kiểm tra vị trí mới của hộp có hợp lệ không và nếu hợp lệ, thì vị trí mới của người chơi và các hộp sẽ được thêm vào cuối hàng đợi frontier, và hành động sẽ được thêm vào actions.

1. BFS

actions.pop()

frontier.popleft()

actions.appendleft(node\_action + [action[-1]])

frontier.append(node + [(newPosPlayer, newPosBox)])

1. UCS
2. **Thống kế kết quả và Nhận xét**
3. **Thống kê kết quả:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cấu hình máy:** | | + HDH: Window 10, 64bit  + Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz  + RAM: 8.00 GB  + Ổ cứng: SSD | | | | | | | | | | |
| **Bản đồ - Trạng thái khởi đầu** | | **Level 1** | **Level 2** | | **Level 3** | | **Level 4** | | **Level 5** | | **Level 6** | |
|  |  | |  | |  | |  | |  | |
| **DFS** | Time | 0.12s | 0.01s | | 0.41s | | 0.01s | | Memory Error | | 0.02s | |
| Cost | 79 | 24 | | 403 | | 27 | | 55 | |
| **BFS** | Time | 0.23s | 0.01s | | 0.32s | | 0.01s | | 334.17s | | 0.03s | |
| Cost | 12 | 9 | | 15 | | 7 | | 20 | | 19 | |
| **UCS** | Time | 0.12s | 0.01s | | 0.16s | | 0.01s | | 132.17s | | 0.02s | |
| Cost | 12 | 9 | | 15 | | 7 | | 20 | | 19 | |
| **Bản đồ - Trạng thái khởi đầu** | | **Level 7** | | **Level 8** | | **Level 9** | | **Level 10** | | **Level 11** | | **Level 12** |
|  | |  | |  | |  | |  | |  |
| **DFS** | Time | 0.98s | | 0.13s | | 0.5s | | 0.02s | | 0.03s | | 0.28s |
| Cost | 707 | | 323 | | 74 | | 37 | | 36 | | 109 |
| **BFS** | Time | 2s | | 0.36s | | 0.03s | | 0.03s | | 0.03s | | 0.17s |
| Cost | 21 | | 97 | | 8 | | 33 | | 34 | | 23 |
| **UCS** | Time | 1.13s | | 0.39s | | 0.02s | | 0.03s | | 0.04s | | 0.19s |
| Cost | 21 | | 97 | | 8 | | 33 | | 34 | | 23 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bản đồ - Trạng thái khởi đầu** | | **Level 13** | **Level 14** | **Level 15** | **Level 16** |
|  |  |  |  |
| **DFS** | Time | 0.35s | 7.44s | 0.3s |  |
| Cost | 185 | 865 | 291 |  |
| **BFS** | Time | 0.27s | 5.19s | 0.49s | 43.40s |
| Cost | 31 | 23 | 105 | 34 |
| **UCS** | Time | 0.36s | 5.87s | 0.53s | 30.44s |
| Cost | 31 | 23 | 105 | 34 |

1. **Nhận xét:**

* So sánh 3 thuật toán DFS, BFS và UCS ở 2 khía cạnh là Thời gian (Time) và Độ dài đường đi (Cost) có thể nhận thấy:

|  |  |  |  |
| --- | --- | --- | --- |
| Thuật toán | DFS | BFS | UCS |
| Nhận xét | DFS khám phá nhiều trạng thái hơn nên tốn nhiều thời gian để tìm ra lời giải và đường đi không tối ưu. Đặc biệt với bản đồ khó, thuật toán DFS có thể dẫn đến tràn RAM (level 5 và level 16 với cấu hình máy em) | Thời gian và độ dài đường đi đã được cải thiện đáng kể so với DFS. | Độ dài đường đi của UCS tìm ra tương tự với BFS bởi vì mọi hành động di chuyển có chi phí bằng nhau. Nhưng thời gian của phần lớn level nhỏ hơn BFS. |
| Đánh giá | So sánh dựa trên thời gian và độ dài đường đi thì UCS là thuật toán tốt nhất so với DFS và BFS. | | |

* Trong các bản đồ, bản đồ ở level 5 là khó giải hơn cả. Vì một số lý do sau:

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | | * Cả 3 thuật toán cần rất nhiều thời gian để giải. Với DFS (MemoryError), BFS (334.17s) và UCS (132.17s) * Không gian trạng thái ở bản đồ level 5 lớn hơn các bản đồ khác nên 3 thuật toán sẽ cần nhiều thời gian để khám phá đường đi |