VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY

CS106 KHTN

Báo cáo bài tập Sokoban_v2

Sinh Viên :
Đồng Quốc Thắng

Giảng viên :
Lương Ngọc Hoàng

Ngày 29 tháng 3 năm 2025

Mục lục

0.1 Giải thích ý tưởng của hàm Heuristic trong code

```
def heuristic(posPlayer, posBox):
    # print(posPlayer, posBox)
    """A heuristic function to calculate the overall distance between
    ↪  the else boxes and the else goals"""
    distance = 0
    completes = set(posGoals) & set(posBox)
    sortposBox = list(set(posBox).difference(completes))
    sortposGoals = list(set(posGoals).difference(completes))
    for i in range(len(sortposBox)):
        distance += (abs(sortposBox[i][0] - sortposGoals[i][0])) +
        ↪  (abs(sortposBox[i][1] - sortposGoals[i][1]))
    return distance
```

Hàm Heuristic này tính tổng của các khoảng cách Manhattan của các hộp tới các vị trí đích của nó.

Ý tưởng của việc sử dụng manhattan:

- Heuristic này sẽ là một admissible heuristic, tức là nó sẽ không bao giờ dự đoán ra cost lớn hơn cost thực tế. Bởi vì trong hàm heuristic này ta không tính tới chướng ngại vật như các hộp khác hay là tường, và nhân vật chỉ di chuyển được theo 4 hướng(lên, xuống, trái, phải). Nên khoảng cách Manhattan là khoảng cách di chuyển của hộp nhỏ nhất có thể đạt được(khi không có chướng ngại vật nếu có chướng ngại vật thì bước đi thực tế sẽ dài hơn)

- Khoảng cách Manhattan cho khoảng cách hợp lý, không bị quá nhỏ nhưng cũng không quá lớn, ở những map rộng và ít chướng ngại vật như map 5, heuristic bằng khoảng cách manhattan cho ra kết quả rất hợp lý

Tuy nhiên sử dụng khoảng cách manhattan cũng có hạn chế như:

- Khi sử dụng khoảng cách manhattan thì sẽ không tính tới việc có chướng ngại vật cản trở trên đường đi nên manhattan không tốt khi map có nhiều tưởng và không gian mở ít

- Khoảng cách manhattan sẽ tính khoảng cách giữa hộp và đích chứ không quan tâm tới luật của sokoban là chỉ đẩy được chứ không thể kéo được, ngoài ra khi sử dụng manhattan, heuristic này không quan tâm tới vị trí của người chơi đối với hộp

Chương 1

Thống kê kết quả

1.1 A*

Level	Số node mở	Thời gian chạy	Số bước đi
1	208	0.0081	13
2	81	0.0035	9
3	99	0.0054	15
4	52	0.0016	7
5	873	0.0534	22
6	473	0.0085	19
7	1224	0.0403	21
8	5128	0.1419	97
9	61	0.0033	8
10	458	0.0124	33
11	627	0.0139	34
12	1253	0.0297	23
13	3923	0.0892	31
14	16748	0.5455	23
15	5366	0.1748	105
16	2693	0.1920	42
17	-	-	-
18	-	-	-

1.2 UCS

Level	Số node mở	Thời gian chạy	Số bước đi
1	1280	0.0338	12
2	131	0.0036	9
3	792	0.0528	15
4	96	0.0029	7
5	562460	51.4365	20
6	587	0.0096	19
7	13640	0.3286	21
8	5254	0.1350	97
9	107	0.0064	8
10	499	0.0130	33
11	649	0.0126	34
12	2806	0.0566	23
13	5821	0.1109	31
14	65683	1.8673	23
15	6283	0.1850	105
16	108391	10.7183	34
17	-	-	-
18	-	-	-

Chương 2

Nhận xét

2.1 Nhận xét chung về A*

A* nhìn chung cho thời gian chạy rất ngắn so với DFS hay BFS từ bài trước, còn so với UCS, thì A* vẫn cho thời gian chạy tốt hơn rất nhiều. Điển hình ở map thứ 5, A* chỉ cần khoảng 0.05s để chạy xong nhưng UCS lại cần đến 51 giây. Số bước đi của A* cho ra nhìn chung không nhiều hơn UCS là bao nhiêu.

Ngoài ra thuật toán A* trong bài này là sử dụng khoảng cách manhattan, nêu sử dụng một heuristic tốt hơn thì có thể cho kết quả còn tốt hơn nữa

2.2 Nhận xét từng level

1. Level 1: UCS mở 1280 node, trong khi A* chỉ mở 208 node, mở ít node hơn rấT nhiều nhưng kết quả cho 13 bước đi, chỉ nhiều hơn UCS 1 bước

2. Level 2, 3, 4: UCS mở gấp rưỡi số node của A* ở map 2, 4, ở map 3 UCS mở nhiều node hơn gần 8 lần nhưng mà kết quả số bước đi đều bằng nhau, A* ở 3 map này cho số bước đi optimal

3. Level 5: Khác biệt lớn nhất giữa UCS và A* là ở map này, UCS mở 562460 node, nhưng A* chỉ mở 873 node, mở rất ít so với UCS, và từ đó cho thời gian chạy rất thấp, chỉ 0.05 giây so với 51 giây của UCS, nhưng kết quả lại không thua thiệt bao nhiêu, A* cho ra 22 bước đi và UCS cho ra 20 bước đi. Chỉ dài hơn 2 bước

4. Level 6, 8, 9, 10, 11, 13, 15: Tất cả các level này đều có điểm chung là A* cho được kết quả optimal giống như UCS, số node mở không quá ít hơn UCS nhưng luôn ít hơn, ta có thể thấy nếu một heuristic hoạt động được tốt thì nó sẽ cho kết quả tốt tương đương với một thuật toán optimal(nhưng phải mở nhiều node) mà cần phải mở ít node hơn

5. Level 7, 12, 14, 16: Ở những map này A* cho kết quả giống UCS, kết quả tối ưu mà còn có thời gian chạy rất ngắn so với UCS bởi vì A* mở ra ít node hơn nhiều.

6. Level 17: Ở level 17, bài này vẫn giống bài trước là sau khi chạy khoảng 17 giây, đưa ra kết quả trống và không tìm được lời giải

7. Level 18: A* cũng không tìm ra được lời giải cho level 18 sau khi treo máy khá lâu nên em cũng không treo máy nữa vì máy khá nóng nhưng mà vẫn chưa tìm được lời giải.

2.3 Kết luận và giải thích

A* chạy nhanh hơn UCS rất nhiều, luôn mở ít số node hơn và số bước đi optimal ở khá nhiều map, ở những map có số bước nhiều hơn UCS, chỉ nhiều hơn một vài bước, cao nhất là chỉ nhiều hơn 10% số bước nhưng lại chạy nhanh hơn rất nhiều lần.

Lý do cho việc khác biệt này là vì:

- UCS chọn cost ít nhất(số bước đẩy thùng ít nhất) ở mỗi nút để di chuyển, từ đó sẽ đẩy các thùng đi đi đến các state lân cận state ban đầu để vét cạn chứ không có mục đích tìm kiếm cụ thể
- Cost của A* ngoài muốn làm các bước đẩy thùng ít nhất, còn muốn chọn bước đẩy thùng gần tới đích nhất, từ đó sẽ chọn cost hợp lý hơn và đưa thùng về phía đích gần hơn ở mỗi state. Ở đây heuristic này không quan tâm đến vị trí của người chơi và vật cản, một heuristic tốt hơn sẽ còn có thể đưa ra kết quả tốt hơn nữa.
- ở những level cần phải mở sâu hơn như map 5 hoặc map 16, UCS sẽ mở gần như toàn bộ các node có cost ở gần đó, dẫn tới UCS sẽ gần giống với BFS, nhưng A* thì sẽ có xu hướng vừa chọn hướng đi có thể đưa thùng về đích nhanh và ít sô bước nên sẽ tránh được rất nhiều nút không cần thiết. Khi state cần tìm ở mức sâu, mở các nút lân cận bằng UCS hay BFS sẽ tăng số nút trong hàng chờ theo hàm mũ, từ đó gây ra thời gian lâu hơn rất nhiều so với A*.