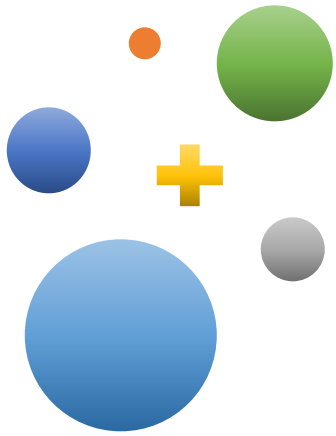
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HOC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Báo cáo đồ án 1

Tìm kiếm heuristic với A*



GVHD: Lê Ngọc Thành

Mục lục:

Báo cáo đồ án 1 1

A. Thành viên nhóm: 3

B. Báo cáo: 3

I. Công việc và mức độ hoàn thành của từng thành viên: 3

1. Thành viên 1: 3

2. Thành viên 2: 3

II. Mức độ hoàn thành của đồ án: 3

III. Những vấn đề chưa thực hiện được: 3

IV. Test cases: 3

1. Test case 1: 3

2. Test case 2: 4

3. Test case 3: 4

4. Test case 4: 6

V. Sơ đồ biểu diễn hệ thống phần mềm: (các hàm chính, công dụng, …): 8

1. Class MapState: 8

2. Class Prob: 9

3. Class Agent: 9

4. Ngoài ra, còn có thêm một số hàm độc lập như: 10

VI. Cấu trúc dữ liệu: 10

VII. Thuật toán chính sử dụng và các cải tiến (nếu có): 10

1. Tìm kiếm A*: 10

2. Các hàm heuristic: 11

C. Tham khảo: 12

References 12

A. Thành viên nhóm:

STT	MSSV	Họ tên	Email
1	1612174	Phùng Tiến Hào	tienhaophung@gmail.com
2	1612269	Võ Quốc Huy	voquochuy304@gmail.com

B. Báo cáo:

I. Công việc và mức độ hoàn thành của từng thành viên:

1. Thành viên 1:

1612174 - Phùng Tiến Hào		
STT	Công việc	Mức độ hoàn thành (%)
1	Cài đặt cấu trúc dữ liệu Priority Queue	100
2	Cài đặt cấu trúc dữ liệu Graph	100
3	Cài đặt hàm input	100
4	Tạo test case	100
5	Format báo cáo	100
6	Review thuật toán A*, hiệu chỉnh code và bổ sung comment	100

2. Thành viên 2:

1612269 - Võ Quốc Huy		
STT	Công việc	Mức độ hoàn thành (%)
1	Cài đặt thuật hàm Heuristic	100
2	Cài đặt thuật toán A*	100
3	Cài đặt hàm output	100
4	Đánh báo cáo	100
5	Kiểm thử phần mềm	100

II. Mức độ hoàn thành của đồ án:

- Hoàn thành 100%

III. Những vấn đề chưa thực hiện được:

- Không có

IV. Test cases:

1. Test case 1:

Input 1	**Output**
7 **0 0** **6 6** **0 0 0 0 0 0 1** **0 0 0 0 0 1 1** **1 1 0 0 0 0 1**	8 (0, 0) (1, 1) (2, 2) (3, 3) (4, 4) (5, 4) (6, 5) (6, 6) S - - - - - o - x - - - o o o o x - - - o - o o x - - -

0 1 1 0 0 0 0 **0 1 1 0 0 1 0** **0 1 0 0 0 1 0** **0 0 0 0 0 0 0**	- o o - x o - - o - - x o - - - - - - x G

2. **Test case 2:**

Input 2	Output
9 **7 3** **0 0** **0 1 0 0 0 0 1 0 0** **0 0 0 1 0 0 0 1 0** **0 0 0 1 0 0 1 0 1** **1 1 0 1 0 1 1 1 1** **0 0 0 1 0 0 0 1 0** **0 1 0 0 0 0 1 0 1** **0 1 1 1 1 0 1 1 1** **0 1 0 0 0 0 1 0 0** **0 0 0 1 1 1 0 1 1**	10 (7,3) (7,4) (6,5) (5,4) (5,3) (4,2) (3,2) (2,1) (1,0) (0,0) G o - - - - o - - x - - o - - - o - - x - o - - o - o o o x o - o o o o - - x o - - - o - - o - x x - o - o - o o o o x o o o - o - S x - o - - - - - o o o - o o

3. **Test case 3:**

Input 3
30 **0 0** **29 29** **0 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0** **0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0** **0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0** **0 1 0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0** **0 0 0 1 1 1 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0** **0 0 0 1 1 1 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1** **0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1** **0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0** **0 0 0 1 1 1 1 0 0 0 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0** **0 0 0 1 1 1 1 0 0 0 0 1 1 0 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0** **0 0 0 1 1 1 1 0 0 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0** **0 0 0 1 1 1 1 0 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0** **0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0** **0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0** **0 0 0 1 1 0 0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0** **0 0 0 0 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0** **0 0 1 1 0 0 0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0** **0 0 1 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0** **1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0**

```
1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0
0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0
1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0
0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0 0 1 0 0 0 0
0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0
0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 1 0
0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 0
0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0
0 1 1 0 0 1 1 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0
```

Output

43

(0, 0) (0, 1) (1, 2) (2, 3) (2, 4) (2, 5) (2, 6) (3, 7) (4, 8) (5, 9) (6, 10) (7, 11) (7, 12) (8, 13) (9, 14) (10, 14) (11, 13) (12, 12) (13, 12) (14, 12) (15, 12) (16, 12) (17, 12) (18, 13) (19, 14) (19, 15) (19, 16) (20, 17) (21, 18) (22, 19) (23, 20) (24, 20) (25, 20) (26, 20) (27, 21) (28, 22) (28, 23) (28, 24) (28, 25) (28, 26) (29, 27) (29, 28) (29, 29)

```
S x o - o - - - - - - o o - - - - - - - - o o o o - - - - -
- o x o - - - - - - - o o - - - - - - - - o o o o - - - - -
- o - x x x x - - - - o o - - - - - - - - o o o o - - - - -
- o - o o o o x - - - o o - - - - - - - - - - o o o o - - -
- - - o o o o - x - - o o - - o o o o - - - - - - o o o o -
- - - o o o o - - x - o o - - o o o o - - - - - - o o o o o
- - - o o o o - - - x - - - - o o o o - - o o - - o o o o o
- - - o o o o - - - - x x - - o o o o - - o o - - - - - - -
- - - o o o o - - - - o o x - o o o o - - o o - - - - - - -
- - - o o o o - - - o o - - x o o o o - - o o - - - - - - -
- - - o o o o - - o o o - - x o o o o - - o o - - - - - - -
- - - o o o o - - o o - - x - o o o o - - o o - - - - - - -
- - - - - - - - o o - - x o o o o - - - - o o - o - o - - -
- - - - - - - - - - o x o o o o - - - - o o - o - o - - -
- - - o o - - - - - o o x o o o o - - - - o o - o - o - - -
- - - - - o o - - - o o x o o o o - - - - o o - o - o - - -
- - o o - - - - - - o o x o o o o - - - - o o - o - o - - -
- - o o - - - - - - o - x o o o o - - - - o o - o - o - - -
o o o o o o o o - - - - - x - - - - - - - - o - o - o - - -
o o o o o o o o - - - - - - x x x - - - - o - - o - o - - -
- - o o - - - - - - - - - o o o o x - - - - - - - - - - o -
o - o o - - - - - - - - - - - - - - - x - - o o - - - - o - -
- - - o - - - - - - - - - - - - - - - x - o o - - - o - - -
- - - - - o o o - - - - - o o o o - - - x o o - - o - - - -
- - - - - o o o - - - - - - o o o - - - x o o o o - - o o -
- - - - - o o o - - - - - o - - o - - - x o o - - - - - o -
- - - - - o o o - - - - - o o - o - - - x o o - - - o o - -
- - - - - o o o - - - - - o o - o - - - - x - - - - o o - -
```

```
- o o - - o o o - - - - - o - o o - - - - - x x x x x - - -
- o o - - o o o - - - - - o o o o - - - - - o o o o o x x G
```

4. **Test case 4:**

Input 4
34 10 1 31 30 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 1 0 1 1 0 1 0 0 0 0 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 0 1 1 0 1 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 0 0 1 1 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 1 0 0 0 0 1 1 0 1 1 0 1 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 0 1 1 0 0 0 1 0 1 1 0 1 1 0 1 1 0 1 0 1 1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1

Output

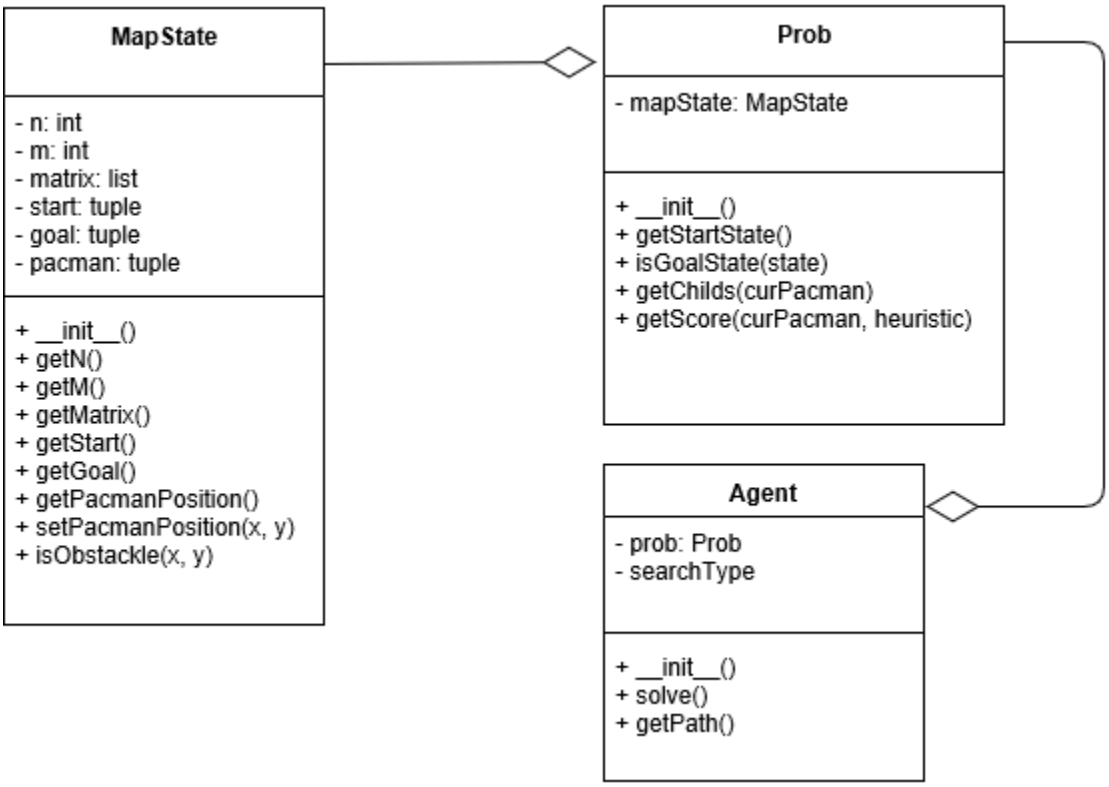
42

(10, 1) (11, 1) (12, 1) (13, 1) (14, 2) (15, 3) (16, 3) (17, 3) (18, 3) (19, 4) (20, 5) (21, 6) (22, 7) (23, 7) (24, 7) (25, 7) (26, 7) (27, 8) (28, 9) (29, 10) (30, 11) (29, 12) (28, 13) (27, 14) (26, 15) (27, 16) (28, 16) (29, 17) (30, 18) (31, 19) (31, 20) (31, 21) (31, 22) (30, 23) (29, 24) (28, 25) (27, 26) (28, 27) (29, 28) (30, 28) (31, 29) (31, 30)

```
o - - - - - - o o - - - - - - - - - - - - - - - - - - - - - - - -
o - - - - - - o o - - - - - - - - - - o - o - - - - - - - o o o o o o
o - o o - - - o o - - - - - - o - - - o - o - - - - - - - o o o o o o
o - o o - - - o o o - - - - - o - - - o - o - - - o - - - - o o o o o o
o o o o - - - o o o - - - - - o - - - o - o - - - o - - - - o o o o o o
o o o o - - - o o o - - - - - o - o - o - o o - o - - - - - o o o o o o
- o o o - o - o o o o - - - - o - o - o - - - o - o - - - - - o o o o o
- o o - o o - o o o o - o - o - o - - - o - o - - - - - o o o o o
- o o - - - o o o o o - o - o - o - - - o - o - - - - - o o o o o
- o - - o o o o o o o - o - o - o o o o - o - o o - - o - o o o o o
o S - - o o o o o o o - o - o - o o o o - o o o o - - o - o o o o o
- x o - o o o o o o o - o - o - o o o o - - o o o - - o - o o o o o
- x o - o o o o o o o - o - o - o o o o - - o o o - - o - o o o o o
- x o - o o o o o o o - o - o - o o o o - - o o o - - o - o o o o o
- - x - o o o o o o o o - o - o o o o - - o o o - - o - o o o o o
- - o x o o o - o - - o o - o - o o o o - - o o o - - o - o o o o o
- o - x o o o - o - - o o - o - - o o - o - - o o o - - o - - - - -
- o - x o o - - o - - o o - o - o o o o - - o o o - - o - o o o o o
- o - x o o - - - o - o o - o - o o o o - - o o o - - o - o o o o o
- o - - x - - - - o - o o - o - o o o o - - o o o o - o - o o o o o
- o - - - x - - - o - o o - o - o o o o - - - - - o - o - - o - o -
- o - o - - x - o o - o o - o - o o o o - - - - - o - o - - o - o o
- o - o - - - x o o - o o - o - o o o o - - - o - o - o - - - - o o
- - - o - o o x o o - o o - o - o o o o - - - o - - - o o - o o - o
- o - o - o o x o o - - - - - o o o o o - - o o - o - - o - - - - o
- o o - - o o x o o - - - - - o - - o o - - o o - o - - o - o o - o
- o o o - o o x o o - - - - - x o - o o - - o o - o - - o - o o - o
- o o o - o o - x - - - - - x o x - o o - - o - - - x - o - o o - o
- - o o - o o - - x - - - x - o x o o o - - o - - x o x - - o o o o
o - o - - - - - - - x - x - - o o x - - - - - - x - o - x o o o o -
o - o - - - - - - - - x - - - o o - x - - - - x - - o - x o - o o -
o - o - - - - - - - - - - - - o o - - x x x x - - - o - - x G o o o
o - o - - - - - - - - - - - - o o - - - - - - - - - o - - - - o o o
o - o - - - - - - - - - - - - o o - - - - - - - - - o - - - - - o o
```

V. Sơ đồ biểu diễn hệ thống phần mềm: (các hàm chính, công dụng, …):

- Có 3 class chính thể hiện bài toán tìm đường đi (như mô tả trong hình):



1. Class MapState:

- Được dùng để thể hiện một trạng thái hoàn toàn của bài toán trong một giai đoạn nhất định.

a) Các variable:

Tên biến	Kiểu dữ liệu	Công dụng
n	int	Độ dài chiều ngang (chiều từ trái sang phải) của bản đồ.
m	int	Độ dài chiều dọc (chiều từ trên xuống) của bản đồ.
matrix	list	Ma trận bản đồ đường đi.
start	tuple	Tọa độ điểm S.
goal	tuple	Tọa độ điểm G.
pacman	tuple	Tọa độ hiện tại của pacman, mặc định là trùng với tọa độ của điểm start.

b) Các method:

Tên phương thức	Công dụng

getN()	lấy giá trị của n.
getM()	lấy giá trị của m.
getMatrix()	lấy ma trận bảng đồ đường đi.
getStart()	lấy tọa độ điểm S.
getGoal()	lấy tọa độ điểm G.
getPacmanPosition()	lấy tọa độ hiện tại của pacman.
setPacmanPosition(x, y)	thiết lập tọa độ hiện tại của pacman với tọa độ (x, y) là tham số đầu vào.
isObStackle(x, y)	kiểm tra tại vị trí (x, y) trên bản đồ có phải là chướng ngại vật không.

2. Class Prob:

- Được dùng để thể hiện bài toán và các truy xuất thông tin liên quan đến việc giải quyết bài toán.

a) Các variable:

Tên biến	Kiểu dữ liệu	Công dụng
mapState	MapState	trạng thái hoàn toàn của bài toán lúc xuất phát (tức vị trí của pacman trùng với vị trí S)

b) Các method:

Tên phương thức	Công dụng
***getStartState()*:**	lấy trạng thái hoàn toàn ban đầu của bài toán.
***isGoalState(state)*:**	kiểm tra xem trạng thái hiện tại có phải là trạng thái đích không. Nếu vị trí hiện tại của pacman trùng với vị trí điểm G, trả về True và ngược lại.
getChilds(curPacman)	trả về danh sách các ô lân cận (dưới dạng tọa độ) mà pacman có thể đến được từ vị trí hiện tại của nó (*curPacman*).
getScore(curPacman, heuristic)	tính giá trị heuristic cho trạng thái hiện tại (*curPacman*) theo hàm tính heuristic được chọn (mặc định là hàm euclidDistance: tính khoảng cách Euclid).

3. Class Agent:

- Thể hiện tác vụ giải quyết bài toán.

a) Các variable:

Tên biến	Kiểu dữ liệu	Công dụng
prob	Prob	bài toán cần phải giải quyết
searchType	Con trỏ hàm	loại tìm kiếm để giải quyết bài toán (BFS, DFS, A*,…), được lưu trữ dưới dạng một hàm giải quyết.

b) Các method:

Tên phương thức	Công dụng
solve()	giải quyết bài toán theo phương pháp được chọn. Trả về danh sách các tọa độ của các điểm trên đường đi từ vị trí S đến vị trí G.
getPath()	Lấy các thông tin cần thiết để in ra file output. Trả về số lượng step cần phải đi, danh sách tọa độ các điểm trên đường đi tìm được và ma trận output theo yêu cầu của đề bài.

4. Ngoài ra, còn có thêm một số hàm độc lập như:

Tên hàm	Công dụng
euclidDistance(a, b)	tính khoảng các Euclid giữa 2 điểm a và b.
aStarSearch(prob	duyệt A* để giải quyết bài toán đầu vào *prob*. Trả về danh sách tọa độ các điểm trên đường đi tìm được.

VI. Cấu trúc dữ liệu:

- ***Stack*:** để hỗ trợ việc lấy danh sách các điểm trên đường đi tìm được.
- ***PriorityQueue*:** được dùng để lấy node có giá trị f = g + h nhỏ nhất trong mỗi lần lặp. Do nhóm lựa chọn cài đặt hàng đợi ưu tiên bằng heap nên khi thêm vào node có cùng giá trị ưu tiên với node trong hàng đợi thì theo cơ chế vun đốn (heapify) thì nó sẽ làm mất đi tính ổn định của mảng. Tức là node mới thêm vào sẽ nằm trước node có sẵn trong hàng đợi mặc dù vốn dĩ node mới đó phải được thêm vị trí vào phía sau của node có sẵn trong hàng đợi.
  Cải tiến nhỏ: Ngoài việc hàng đợi có độ ưu tiên (priority) và giá trị (value) thì có thêm một biến thứ tự (count). Vì để xét trường hợp khi push vào node có độ ưu tiên trùng với node có sẵn trong hàng đợi ưu tiên thì sẽ không làm thay đổi thứ tự của hàng đợi.

VII. Thuật toán chính sử dụng và các cải tiến (nếu có):

1. Tìm kiếm A*:

- Thuật toán chính nằm ở hàm aStarSearch.

Bước 1: Khai báo các biến hỗ trợ.

- *pq*: hàng đợi ưu tiên, mỗi node gồm vị trí hiện tại của pacman với giá trị ưu tiên là f = g + h.
- *occ*(kiểu dictionary): dùng để kiểm tra một trạng thái có được expand chưa (tức được pop ra khỏi pq).
- *dist*(kiểu dictionary): số lượng step để đi từ S đến điểm bất kỳ tại một thời điểm nhất định .
- *traverse*(kiểu dictionary): dùng để truy vết đường đi. Với Key là tọa độ điểm bất kỳ trên đường đi tìm được, Value là tọa độ điểm ngay trước điểm Key.

Bước 2: Bắt đầu duyệt từ điểm S.

Bước 3: Lặp các yêu cầu sau cho đến khi pq trống hoặc một trạng thái goal đã được expand

- Lấy node trên cùng của pq.
  Kiểm tra xem trạng thái của node này có phải là goal không? Nếu là goal, thoát lặp. Ngược lại, tiếp tục.
- Kiểm tra trạng thái của node này có được expand chưa? Nếu chưa, tiếp tục. Ngược lại, nhảy đến bước lặp tiếp theo.
- Đánh dấu trong occ là node này đã được expand.
- Khi xét các node con đến được từ node cha đang xét và node con này phải chưa được expand, nếu tìm được một đường đi tốt hơn đến một node con thì ta phải cập nhật lại số lượng step đi từ S đến node con đó trong *dist*. Đồng thời, ta cũng phải cập nhật lại *traverse* và *push* node con này (và giá trị ưu tiên của nó) vào *pq*.

Bước 4: Kết thúc vòng lặp và tìm được đường đi theo heuristic được chọn.

2. Các hàm heuristic:

a) Khoảng cách euclid: Đây là inadmissible heuristic.

	0	1	2	3
0	S			
1		x		
2				
3				G

- Xét trường hợp như trong bản đồ trên mà tại đó pacman đang ở ô (1, 1) và G ở ô (3, 3).
- Đường đi ngắn nhất để pacman đi từ (1, 1) đến G là: (1, 1) -> (2, 2) -> (3, 3). Ta thấy cần phải tốn 2 step để đi theo đường đi này.
- Tuy nhiên, khoảng cách Euclid của (1, 1) và G là

$$(3 \quad \sqrt{-1)^2 + (3 - 1)^2} = 2\sqrt{2}$$

- Do $2\sqrt{2} > 2$ nên khoảng cách euclid không thỏa tính chất của một admissible heuristic.
  ⇨ Không thể tìm được đường đi ngắn nhất với heuristic này.

b) *Cải tiến:*

- Lấy khoảng cách euclid tính được chia cho 2, ta sẽ có được một admissible heuristic.

C. Tham khảo:

References

Stuart J. Russell and Peter Norvig, 2010. *Artificial Intelligence: A Modern Approach.* 3rd ed. s.l.:Prentice Hall.