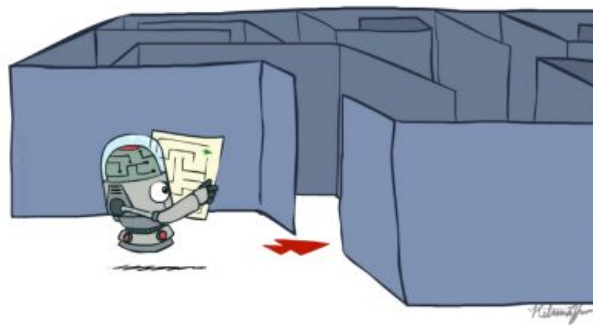


Trường đại học Khoa học Tự nhiên
Khoa công nghệ thông tin

BÁO CÁO ĐỒ ÁN TÌM KIẾM ĐƯỜNG ĐI NGẮN NHẤT

Search



Danh sách thành viên

- | | |
|--------------------------|---------|
| 1. Võ Quốc Thắng | 1712162 |
| 2. Lê Nguyễn Nhật Trường | 1712195 |
| 3. Lê Tuấn Đạt | 1712329 |

25/10/2019

BỘ MÔN NHẬP MÔN CƠ SỞ TRÍ TUỆ NHÂN TẠO

MỤC LỤC

| | |
|---|-----------|
| 1. Đề bài | 2 |
| 1.1. Nội dung | 2 |
| 1.2. Yêu cầu | 2 |
| 2. Thông tin | 2 |
| 2.1 Thông tin thành viên | 2 |
| 2.2. Phân công công việc | 2 |
| 3. Cài đặt | 3 |
| 3.1. Level 1 : Sử dụng thuật toán A* search để tìm đường đi ngắn nhất | 3 |
| 3.1.1 Thuật toán A* search | 3 |
| 3.1.2. Quá trình chạy | 4 |
| 3.2 : Level 2 : So sánh với ba thuật toán A * search , DFS, và BFS | 6 |
| 3.2.1 Thuật toán DFS | 6 |
| 3.2.2 Thuật toán BFS | 6 |
| 3.2.3 So sánh ba thuật toán | 6 |
| 3.3. Level 3: Thuật toán tìm đường đi ngắn nhất và đón các điểm | 9 |
| 3.3.1 Thuật toán | 9 |
| 3.3.2 Quá trình chạy | 9 |
| 3.4 Level 4 : Thuật toán A * search với các đa giác di chuyển | 11 |
| 3.4.1. Cài đặt | 11 |
| 3.4.2. Link demo | 11 |
| 4. Hướng dẫn sử dụng | 11 |
| 5. Tổng kết | 12 |
| 6. Đánh giá | 12 |
| 6.1 Đánh giá đồ án | 12 |
| 6.2. Đánh giá thành viên | 13 |
| REFERENCES | 14 |

1. Đề bài

1.1. Nội dung

Nghiên cứu, trình bày và cài đặt các thuật toán tìm kiếm đường đi

1.2. Yêu cầu

- ☐ Ngôn ngữ sử dụng Python.
- ☐ Được phép sử dụng thư viện đồ họa.

2. Thông tin

2.1 Thông tin thành viên

| Họ và tên | MSSV | Email |
|-----------------------|---------|--|
| Võ Quốc Thắng | 1712162 | voquochangit@gmail.com |
| Lê Nguyễn Nhật Trường | 1712195 | truongthk62014@gmail.com |
| Lê Tuấn Đạt | 1712329 | letuandat2110@gmail.com |

2.2. Phân công công việc

| STT | Công việc | Phụ trách |
|-----|---|-----------|
| 1 | Thiết kế xử lý đọc file input | Đạt |
| 2 | Level 1 (Trình bày và cài đặt thuật toán A* search) | Trường |
| 3 | Level 2 (Trình bày và cài đặt thuật toán BFS) | Thắng |

| | | |
|---|---|--------------------|
| 4 | Level 2 (Trình bày và cài đặt thuật toán DFS) | Đạt |
| 5 | Level 3 (Thuật toán đón các điểm) | Trường |
| 6 | Level 4 (Xử lý với đa giác di chuyển) | Đạt |
| 7 | Thiết kế xử lý output với đồ họa (pygame) | Thắng |
| 8 | Tạo test | Trường |
| 9 | Viết báo cáo | Trường, Đạt ,Thắng |

3. Cài đặt

3.1. Level 1 : Sử dụng thuật toán A* search để tìm đường đi ngắn nhất

3.1.1 Thuật toán A* search

- ❑ Tìm kiếm A* mở các nút có $f(n) = g(n) + h(n)$ nhỏ nhất. A* đầy đủ và tối ưu nếu hàm $h(n)$ là chấp nhận được. Độ phức tạp về mặt không gian của A* vẫn rất cao.
- ❑ Đối với bài toán này , độ ưu tiên để mở các nút là $f(n) = g(n) + h(n)$. Trong đó $g(n)$ là chiều dài đường đi từ start đến điểm N do đồ thị này không có trọng số, $h(n)$ là giá trị hàm heuristic từ điểm N đến goal (ở đây chúng tôi dùng khoảng manhattan).
- ❑ Mã giả:

```

Put node_start in the OPEN list with  $f(\text{node\_start}) = h(\text{node\_start})$  (initialization)

while the OPEN list is not empty {
    take from the open list the node node_current with the lowest
         $f(\text{node\_current}) = g(\text{node\_current}) + h(\text{node\_current})$ 

    if node_current is node_goal we have found the solution;
    break
    generate each state node_successor that come after node_current

```

```

for each node_successor of node_current {
    Set successor_current_cost = g(node_current) +
w(node_current, node_successor)

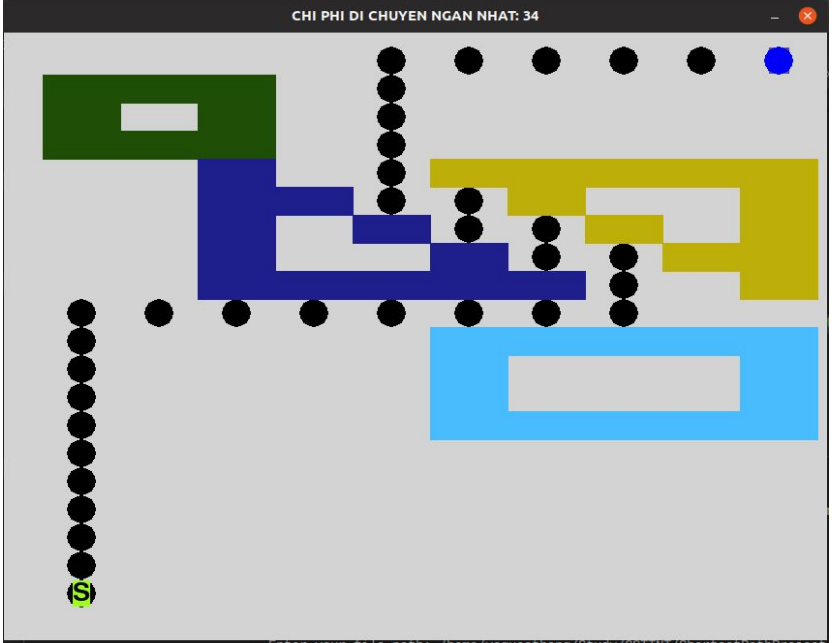
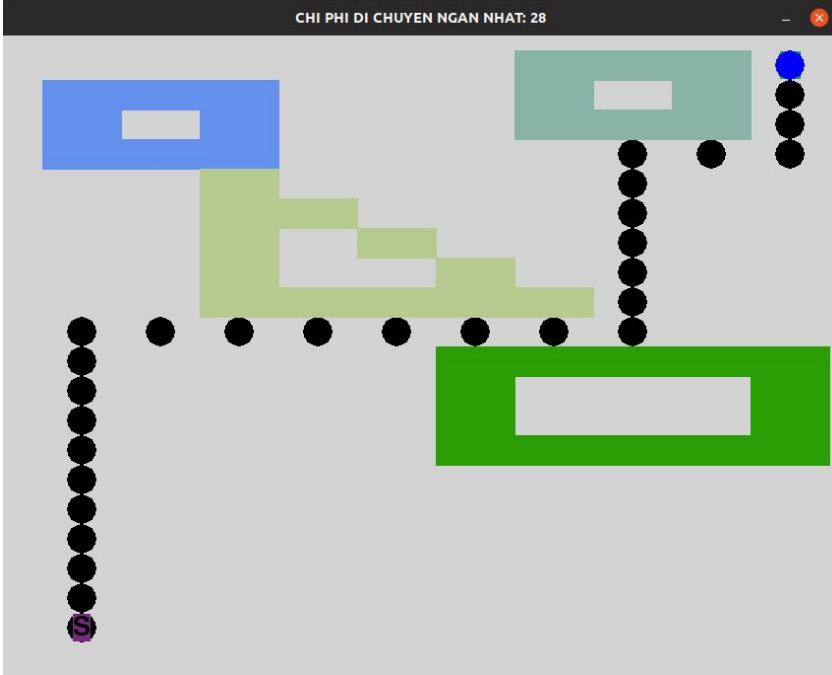
    if node_successor is in the OPEN list {
        if g(node_successor) ≤ successor_current_cost
        continue (to line 20)
    } else if node_successor is in the CLOSED list {
        if g(node_successor) ≤ successor_current_cost
        continue (to line 20)
        Move node_successor from the CLOSED list to
the OPEN list
    } else {
        Add node_successor to the OPEN list
        Set h(node_successor) to be the heuristic
distance to node_goal
    }
    Set g(node_successor) = successor_current_cost
    Set the parent of node_successor to node_current
}
Add node_current to the CLOSED list
}

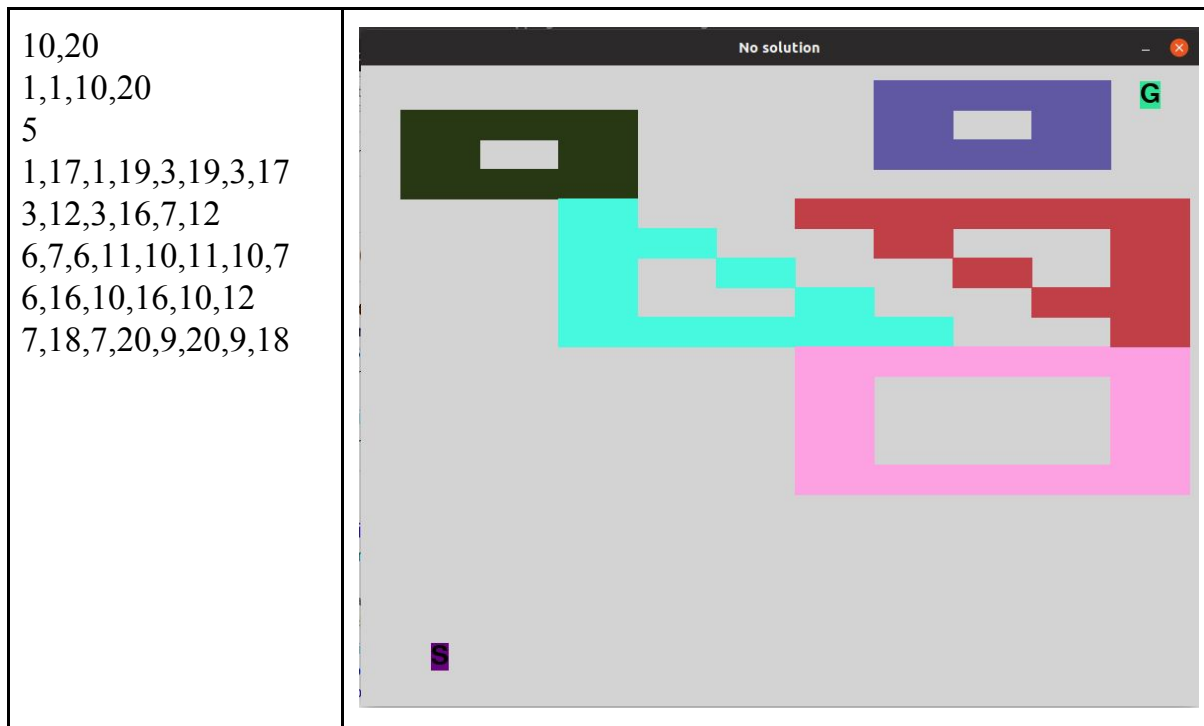
if(node_current != node_goal) exit with error (the OPEN list is
empty)

```

3.1.2. Quá trình chạy

- ❑ Hiệu quả của các thuật toán tìm kiếm heuristic phụ thuộc vào chất lượng của các hàm heuristic. Các heuristic tốt có thể được xây dựng bằng cách giản lược định nghĩa bài toán, hoặc bằng cách tính toán trước các chi phí cho các bài toán con trong cơ sở dữ liệu mẫu.
- ❑ Chạy thử với test:

| Input.txt | Output |
|---|---|
| <p>10,20 1,1,10,20 4 1,17,1,19,3,19,3,17 3,12,3,16,7,12 6,7,6,10,10,10,10,7 6,16,10,16,10,12</p> |  |
| <p>10,20 1,1,10,20 4 1,17,1,19,3,19,3,17 3,12,3,16,7,12 6,7,6,10,10,10,10,7 7,18,7,20,9,20,9,18</p> |  |



3.2 : Level 2 : So sánh với ba thuật toán A * search , DFS, và BFS

3.2.1 Thuật toán DFS

Tìm kiếm theo chiều sâu chọn nút sâu nhất chưa mở trên cây để mở rộng nó. Thuật toán không đầy đủ lần tối ưu và có độ phức tạp thời gian là $O(b^m)$ và độ phức tạp không gian là $O(b \cdot m)$, trong đó m là độ sâu tối đa của bất kỳ đường đi nào trong không gian trạng thái và b là số nút con tối đa trong không gian trạng thái.

3.2.2 Thuật toán BFS

Thuật toán BFS là thuật toán xét (duyệt) hoặc tìm kiếm trên cây và đồ thị, có chiến lược tìm kiếm mù (tìm kiếm không có định hướng, không chú ý đến thông tin, giá trị được duyệt).

Từ một đỉnh (nút) gốc ban đầu là đỉnh đang xét, xác định và lần lượt duyệt các đỉnh kề xung quanh đỉnh gốc vừa xét. Tiếp tục quá trình duyệt qua các đỉnh kề đỉnh vừa xét cho đến khi đạt được kết quả cần tìm hoặc duyệt qua tất cả các đỉnh.

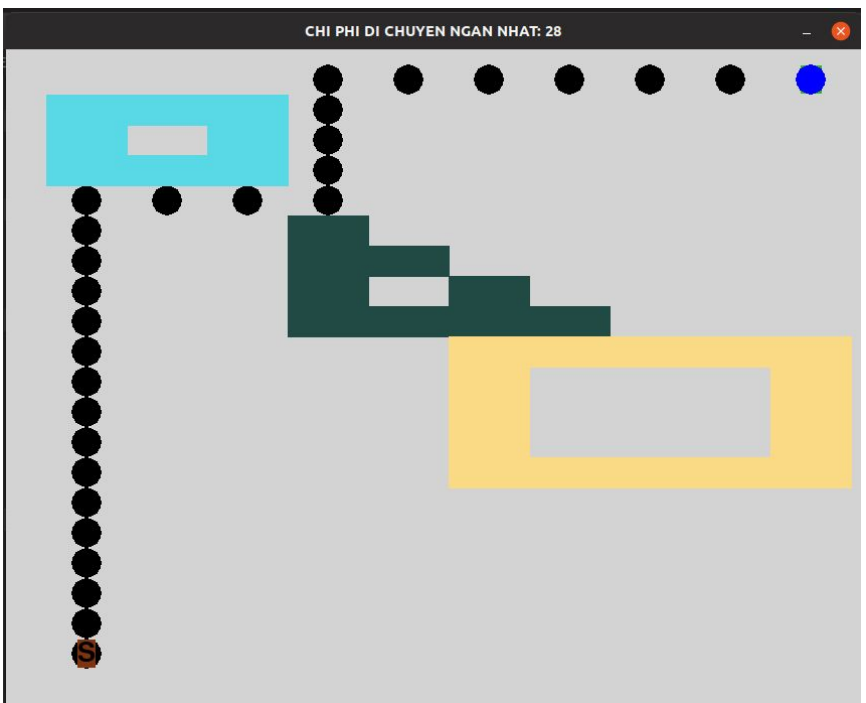
Độ phức tạp của thuật toán $O(V+E)$ (V là số đỉnh, E là số cạnh)

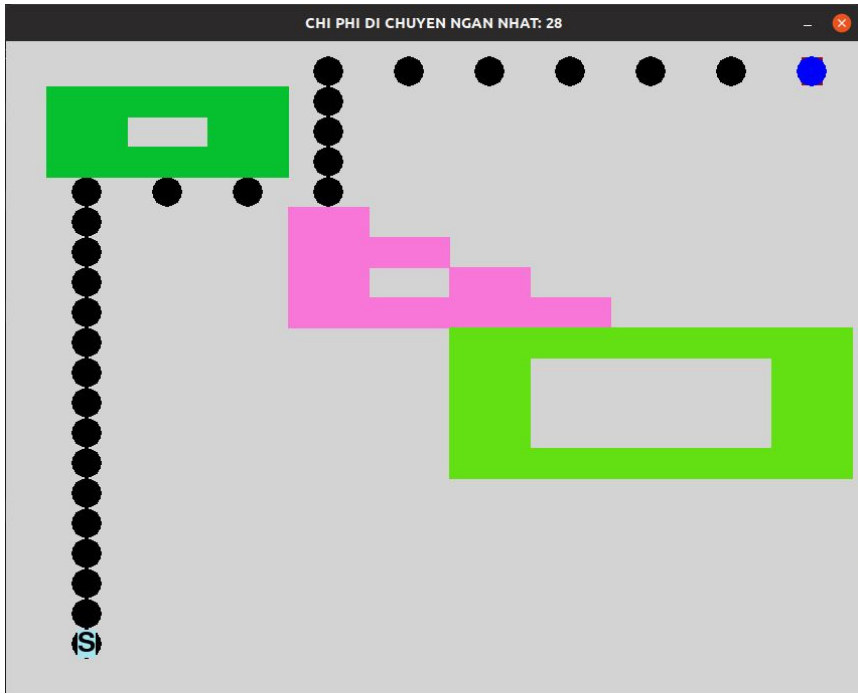
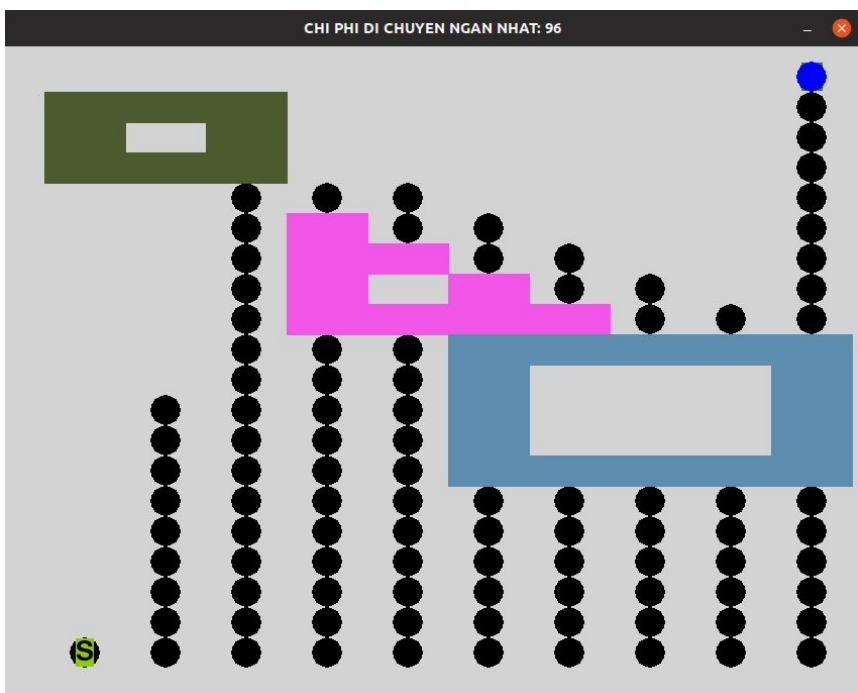
3.2.3 So sánh ba thuật toán

❑ Test :

Input.txt
10,20
1,1,10,20
3
1,17,1,19,3,19,3,17
4,12,4,15,7,12
6,7,6,11,10,11,10,7

❑ Kết quả:

| Thuật toán | Output |
|------------|---|
| A* search |  |

| | |
|-----|---|
| BFS |  |
| DFS |  |

❑ Nhận xét :

- Thuật toán A* Search và BFS cho phép tìm được đường đi ngắn nhất còn DFS chỉ tìm được đường đi nhưng chưa tối ưu.
- Thuật toán A* Search chạy với tốc độ tốt nhất (số lần chạy thử để tới được đích) dựa trên hàm heuristic.

3.3. Level 3: Thuật toán tìm đường đi ngắn nhất và đón các điểm

3.3.1 Thuật toán

Đầu tiên ta để điểm bắt đầu vào queue.

Ví dụ ta có các điểm đón = [A, B, C, D]

Ta có queue = [start] ta xét phần tử cuối cùng của queue rồi lần lượt tính heuristic các điểm đón. Chọn phần tử có giá trị heuristic nhỏ nhất với điểm cuối trong queue, thêm phần tử đó vào queue và bỏ phần tử đó ra khỏi các điểm đón

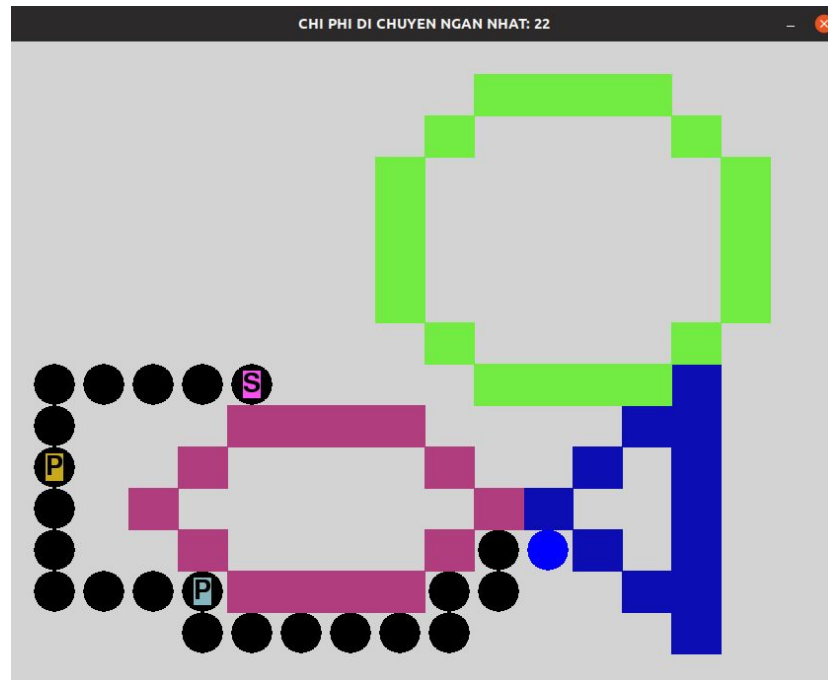
| Queue | Pick up points | Heuristic min |
|--------------------------|----------------|----------------------|
| [start] | [A, B, C, D] | (start, C) |
| [start, C] | [A, B, D] | (C, B) |
| [start, C, B] | [A, D] | (B, D) |
| [start, C, B, D] | [A] | (D, A) |
| [start, C, B, D, A] | [] | Thêm phần tử end vào |
| [start, C, B, D, A, end] | | |

Từ queue ta lấy từng cặp điểm liên tiếp: (start, C), (C, B), (B, D), (D, A), (A, end) và tái sử dụng lại A* Search ở level 1

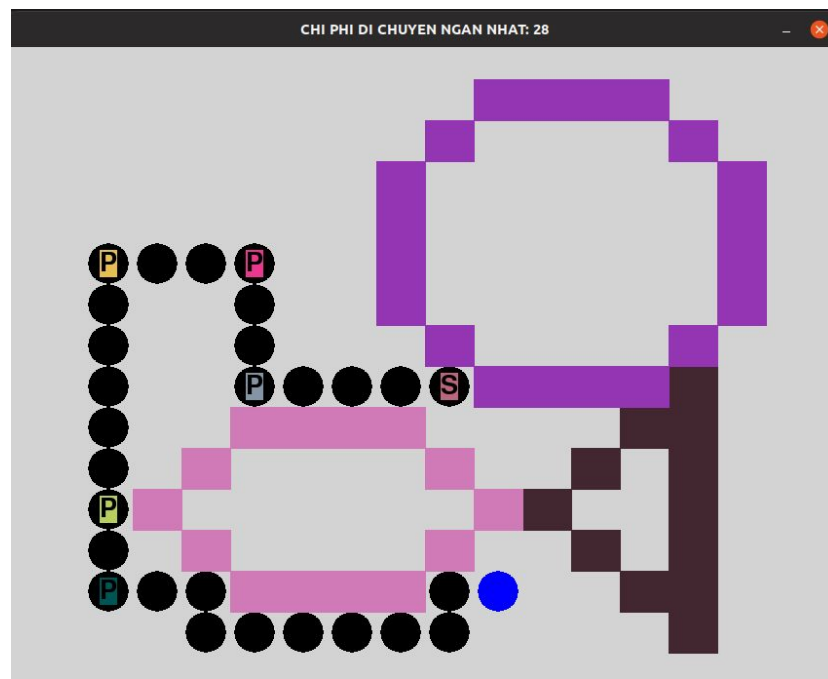
3.3.2 Quá trình chạy

| Input.txt | Output |
|-----------|--------|
|-----------|--------|

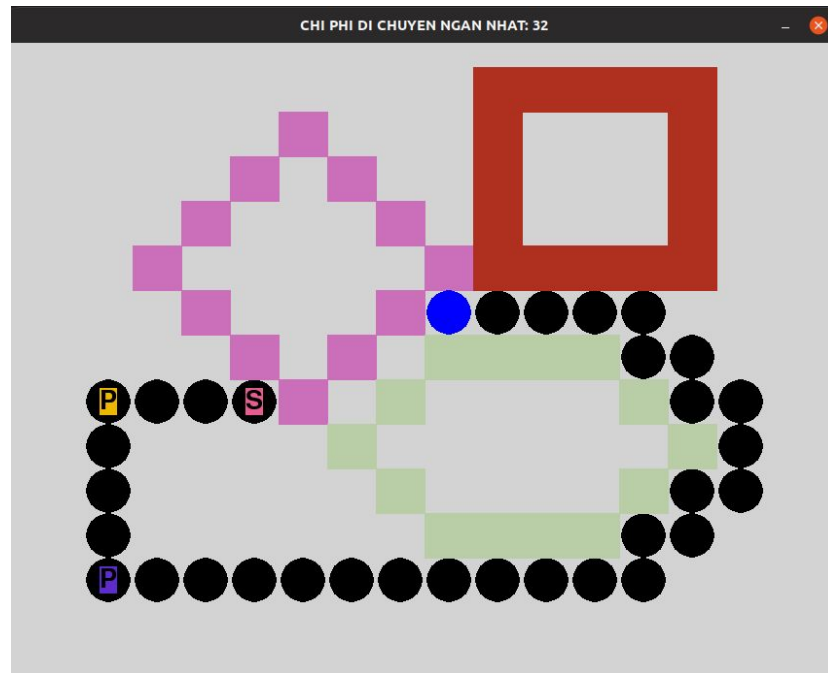
16,14
 5,7,11,3,1,5,4,2
 3
 11,4,14,7,14,1
 5,2,3,4,5,6,8,6,10,4,8,2
 10,7,8,9,8,12,10,14,13,14
 ,15,12,15,9,13,7



16,14
 9,7,10,2,5,7,5,10,2,10,2,2
 ,2,4
 3
 11,4,14,7,14,1
 5,2,3,4,5,6,8,6,10,4,8,2
 10,7,8,9,8,12,10,14,13,14
 ,15,12,15,9,13,7



16,13
 5,6,9,8,2,6,2,2
 3
 6,6,3,9,6,12,9,9
 10,9,10,13,14,13,14,9
 7,5,9,7,12,7,14,5,12,3,9,3



3.4 Level 4 : Thuật toán A * search với các đa giác di chuyển

3.4.1. Cài đặt

- ☐ Các đa giác di chuyển sao cho không trùng lên nhau và vật không chạm vào vị trí của các đa giác.
- ☐ Việc đa giác di chuyển làm ảnh hưởng đến bước đi của vật trong thuật toán A * search
- ☐ Lưu lại các trạng thái di chuyển của polygons và quá trình tìm đường đi của thuật toán A* search.
- ☐ Cuối cùng thể hiện đường đi đã chọn trên bản đồ

3.4.2. Link demo

<https://youtu.be/48cSOiGZuwo>

4. Hướng dẫn sử dụng

Để chạy chương trình, sử dụng terminal, nhập lệnh: > **python3 main.py**

Màn hình hiển thị menu:

1. level 1

2. level 2
3. level 3
4. level 4
5. Press 0 to exit

(Yêu cầu nhập vào level muốn chạy)

Level:

(Yêu cầu nhập vào đường dẫn chứa test.txt)

Enter your file path:

Kết quả hiển thị một cửa sổ với chi phí đường đi ở tiêu đề và đồ họa cho thấy các bước di chuyển. Lưu ý, đối với trường hợp các đa giác di chuyển màn hình sẽ hiển thị quá trình tìm đường đi của vật và cuối cùng sẽ hiển thị đường đi ngắn nhất của vật.

5. Tổng kết

Qua đồ án này nhóm đã có cơ hội để tìm hiểu về các thuật toán tìm đường đi, cài đặt trên môi trường python cũng như các thư viện đồ họa. Ngoài ra, đồ án còn giúp nhóm biết cách phân chia công việc hợp lý và giải quyết các bài toán phát sinh, kết nối bài giảng trên lớp với thực hành.

6. Đánh giá

6.1 Đánh giá đồ án

| Level | Nội dung | Mức độ hoàn thành |
|---------|---|-------------------|
| Level 1 | Sử dụng thuật toán A * search để tìm đường đi | 100% |
| Level 2 | So sánh quá trình chạy ba thuật toán A * search, DFS, BFS | 100% |
| Level 3 | Thuật toán tìm đường đi có đón điểm | 100% |

| | | |
|------------|--|------|
| Level 4 | Chạy thuật toán A* search trong trường hợp các đa giác di chuyển | 100% |
| Tổng đồ án | Hoàn thành bốn level | 100% |

6.2. Đánh giá thành viên

| Thành viên | Nội dung | Khối lượng công việc | Mức độ hoàn thành |
|-----------------------|--|----------------------|-------------------|
| Lê Tuấn Đạt | - Xử lý file input - DFS - Đa giác di chuyển | 30% | 100% |
| Lê Nguyễn Nhựt Trường | - A* search - Xử lý đón điểm - Tạo test | 30% | 100% |
| Võ Quốc Thắng | - BFS - Đồ họa - Đa giác di chuyển - Báo cáo tổng kết | 40% | 100% |

REFERENCES

- ❑ <https://www.geeksforgeeks.org/a-search-algorithm/>
- ❑ <https://medium.com/tebs-lab/breadth-first-search-and-depth-first-search-4310f3bf8416>
- ❑ <https://www.tutorialspoint.com/python/index.htm>
- ❑ <https://www.pygame.org/docs/>
- ❑ <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>