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

○♦♦♦ 🕮 ♦♦♦○

PROJECT 1 – SEARCH

FINDING PATH BY SEARCH ALGORITHM

Môn học: Cơ sở trí tuệ nhân tạo

GIÁO VIÊN HƯỚNG DẪN

TS. Nguyễn Hải Minh
TS. Nguyễn Ngọc Thảo
Trợ giảng. Nguyễn Thái Vũ

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

○♦♦♦ 🕮 ♦♦♦○

PROJECT 1 – SEARCH

FINDING PATH BY SEARCH ALGORITHM

Môn học: Cơ sở trí tuệ nhân tạo

SINH VIÊN THỰC HIỆN

20127662 - Nguyễn Đình Văn
20127166 – Nguyễn Huy Hoàn
20127061 – Lưu Minh Phát

MỤC LỤC

Mục lục

THÔNG TIN THÀNH VIÊN

Mã số sinh viên	Họ và tên	Nhiệm vụ
20127662	Nguyễn Đình Văn	- Code tạo maze và vật cản Polygon, hàm main. - Code 3 thuật toán UCS, A*, IDS - Viết báo cáo
20127166	Nguyễn Huy Hoàn	- Code thuật toán BFS
20127061	Lưu Minh Phát	- Code thuật toán GBFS

CÁC CHỨC NĂNG ĐÃ HOÀN THÀNH

STT	Tên chức năng	Mức độ hoàn thành	Ghi chú
1	Breadth first search	100 %	
2	Uniform cost search	100 %	
3	Iterative deepening search	100 %	
4	Greedy best first search	100 %	
5	Graph search A*	100 %	

SƠ LƯỢC VỀ TỔ CHỨC CODE TRONG ĐỒ ÁN

- Ngôn ngữ: Python
- Các thư viện sử dụng: numpy, turtle
- Các Class:
    - Pen: Dùng để vẽ hình vuông trên màn hình
    - Draw
        - self.level: lưu ma trận được truyền vào
        - self.pen: Vẽ hình vuông
        - self.maze_height: chiều cao ma trận
        - self.maze_width: chiều rộng ma trận
        - axis: Vẽ đánh số trục Ox, Oy
        - drawSquare: Vẽ hình vuông
        - drawCharacter: Vẽ kí tự
        - setup_maze: Tạo ra ma trận trên màn hình
        - polygons: Tạo các vật cản
        - aPolygon: Tạo một vật cản
        - edgeOfPolygon: Tạo một cạnh của vât cản
    - Graph
        - self.start: tọa độ điểm bắt đầu
        - selft.goal: tọa độ điểm kết thúc
        - self.costExpandNode: số node đã expand
        - self.costPath: số node đi từ start đến goal
        - self.depth: Với IDS thì đếm chiều sâu khi tìm được goal
        - calcG: Tính chi phí đường đi
        - calcH: Tính Heuristic
        - posOfQueue: tính thứ tự ưu tiên của hàng đợi
        - drawStartGoal: vẽ đường đi từ start đến goal
        - printCost: In chi phí ra màn hình
        - BFS: breadth first search
        - UCS: uniform cost search
        - GBFS: Greedy best first search
        - Astar: Graph search A*
        - DLS: Depth limited search
        - IDS: Iterative deepening search
- Các hàm chính:
    - Hàm Visualize: Biểu diễn ma trận, các đa giác vật cản, node expand, đường đi, chi phí nút mở rộng và chi phí đường dẫn cuối cùng.
    - Hàm main: Đọc dữ liệu từ file *"input.txt"* và truyền dữ liệu cho hàm Visualize thực hiện.

CÁC THUẬT TOÁN SEARCH

1. Breadth first search – BFS

- Ý tưởng: Sử dụng hàng đợi truy cập vào các node lân cận (Trên, dưới, trái, phải) đưa vào hàng đợi, duyệt theo thứ tự FIFO cho đến thi tìm được node Goal.
- Input:

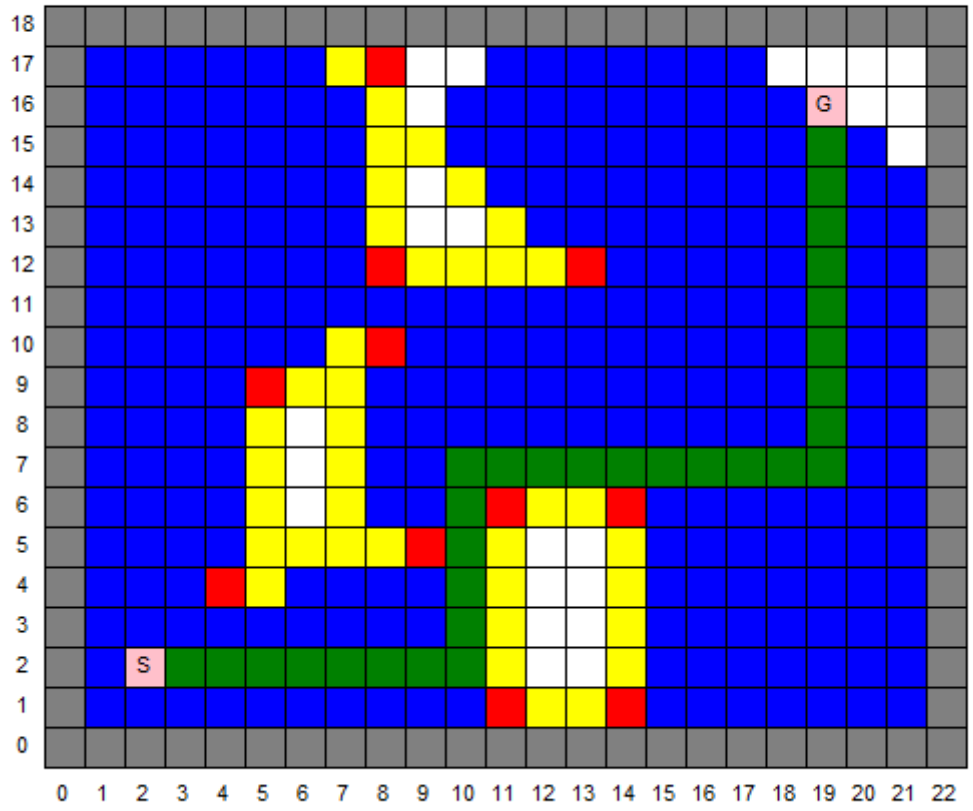
input.txt

```
22 18
2 2 19 16
3
4 4 5 9 8 10 9 5
8 12 8 17 13 12
11 1 11 6 14 6 14 1
```

- Output:

Cost of expanded node: 284

Cost of final path: 32



- Kết luận: Tìm ra được đường đi từ Start đến Goal.
- Ưu điểm: Duyệt tổng quát, luôn tìm ra được đường đi từ start đến Goal, giải quyết tốt bài toán trong thời gian và không gian tối thiểu.
- Nhược điểm: Đối với bài toán có không gian rộng thì sẽ tốn thời gian và không gian tìm kiếm.

2. Uniform cost search – UCS

- Ý tưởng: Sử dụng hàng đợi truy cập vào các node lân cận (Trên, dưới, trái, phải) đưa vào hàng đợi, duyệt theo thứ tự FIFO và chi phí đường đi cho đến thi tìm được node Goal. Chi phí đường đi được tính theo công thức: Chi phí hiện tại = chi phí node cha đến con + chi phí của node Cha. Từ đó sắp xếp lại thứ tự ưu tiên trong hàng đợi.
- Input:

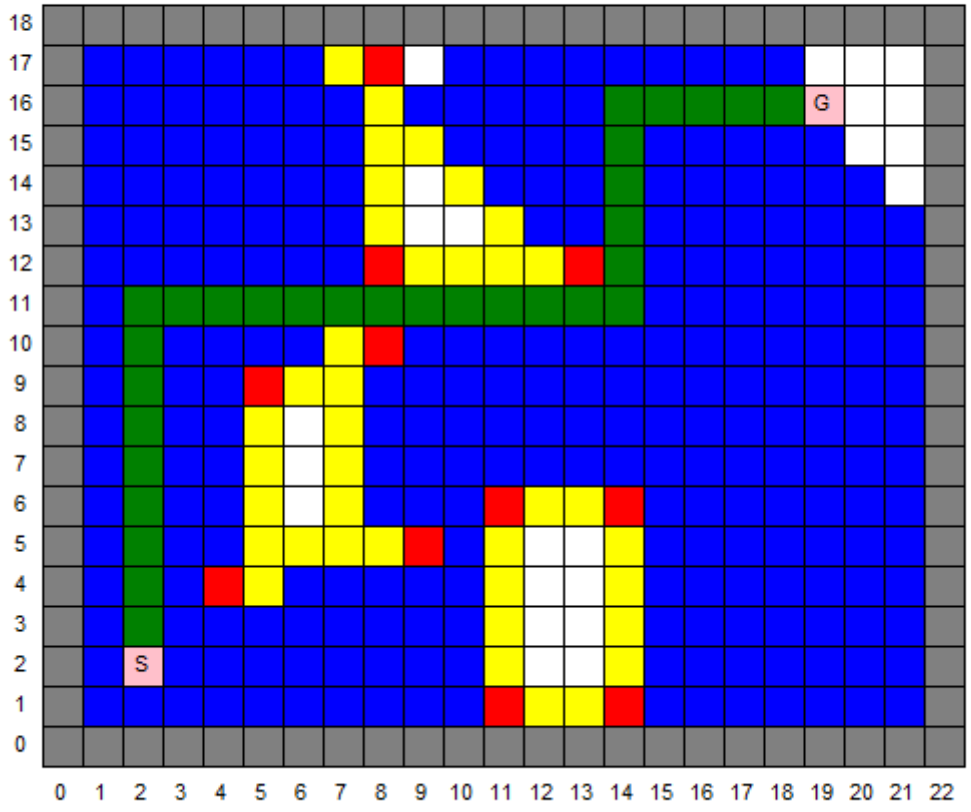
input.txt

```
22 18
2 2 19 16
3
4 4 5 9 8 10 9 5
8 12 8 17 13 12
11 1 11 6 14 6 14 1
```

- Output:

Cost of expanded node: 285

Cost of final path: 32



- Kết luận: Tìm ra được đường đi ngắn nhất từ Start đến Goal
- Ưu điểm: Luôn tìm ra con đường có chi phí thấp nhất, ở mọi không gian.
- Nhược điểm: Nó không quan tâm đến số bước liên quan đến việc tìm kiếm mà chỉ quan tâm đến chi phí đường đi. Do đó thuật toán này có thể mắc kẹt ở không gian vô hạn.

3. Iterative deepening search – IDS

- Ý tưởng: Sử dụng vòng lặp để tăng độ sâu mỗi lần duyệt, và sử dụng thuật toán DLS để search. Mỗi lần vòng lặp tăng một đơn vị sẽ truyền sẽ tiến hành duyệt sâu với độ sâu bằng đơn vị truyền vào cho đến khi tìm thấp node Goal.
- Input:

input.txt

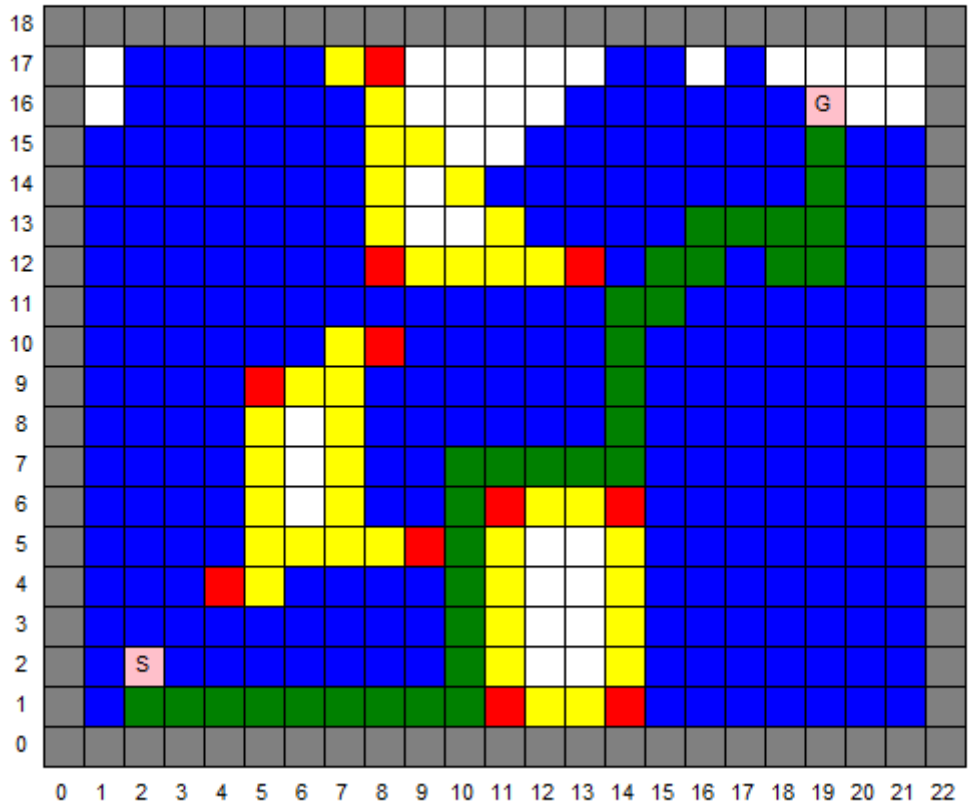
```
22 18
2 2 19 16
3
4 4 5 9 8 10 9 5
8 12 8 17 13 12
11 1 11 6 14 6 14 1
```

- Output:

Cost of expanded node: 120

Cost of final path: 36

Depth: 35



- Kết luận: Tìm thấy đường đi từ Start đến Goal
- Ưu điểm: Nó kết hợp BFS và DFS về khả năng tìm kiếm nhanh và hiệu quả bộ nhớ
- Nhược điểm: Nó lặp lại các node ở độ sau trước đó. Không tìm được đường đi ngắn nhất. Với độ sâu càng lớn thì thời gian duyệt càng lâu.

4. Greedy best first search - GBFS

- Ý tưởng: Sử dụng hàng đợi truy cập vào các node lân cận (Trên, dưới, trái, phải) đưa vào hàng đợi, duyệt theo thứ tự FIFO và heuristic cho đến thi tìm được node Goal. Heuristic được tính theo công thức: Heuristic node hiện tại = chi phí từ node đó đến Goal (Lấy |y – y'| + |x – x'|). Từ đó sắp xếp lại thứ tự ưu tiên trong hàng đợi.

- Input:

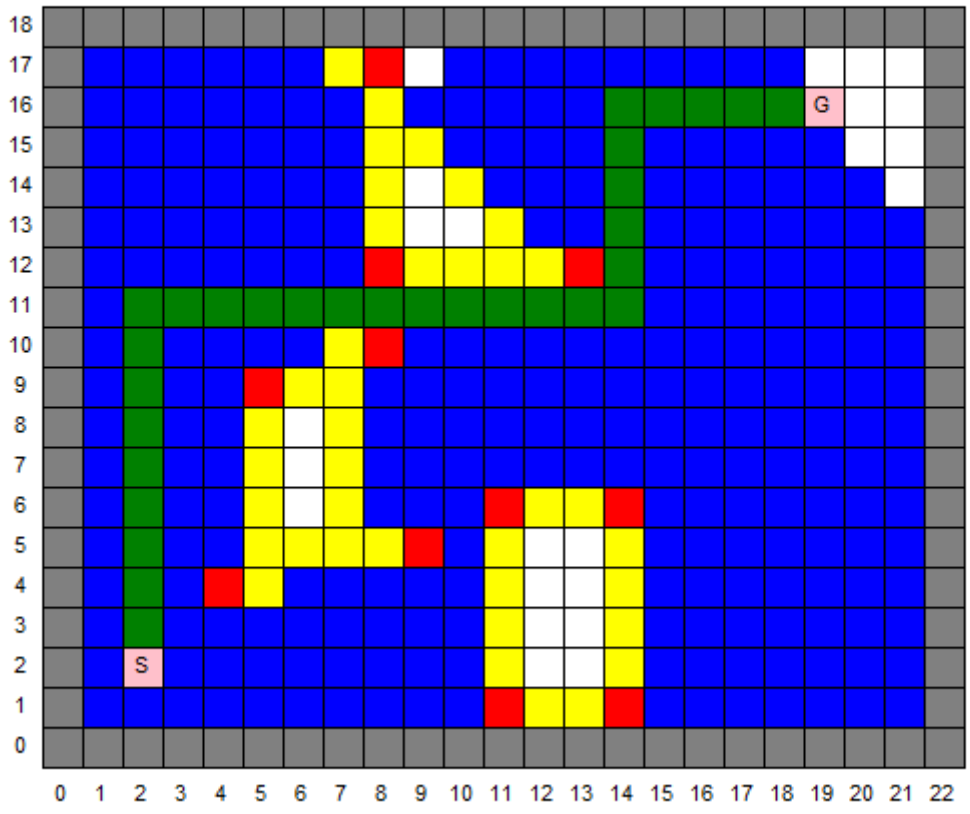
input.txt

```
22 18
2 2 19 16
3
4 4 5 9 8 10 9 5
8 12 8 17 13 12
11 1 11 6 14 6 14 1
```

- Output:

Cost of expanded node: 285

Cost of final path: 32



- Kết luận: Tìm được đường từ Start đến Goal
- Ưu điểm: Tìm kiếm nhanh
- Nhược điểm: Độ hiệu quả của thuật toán hoàn toàn dựa vào Heuristic.

5. Graph search A* - Astar

- Ý tưởng: Sử dụng hàng đợi truy cập vào các node lân cận (Trên, dưới, trái, phải) đưa vào hàng đợi, duyệt theo thứ tự FIFO, heuristic và chi phí đường đi cho đến thi tìm được node Goal. Heuristic được tính theo công thức: Heuristic h(x) node hiện tại = chi phí từ node đó đến Goal (Lấy |y – y'| + |x – x'|). Chi phí hiện tại g(x) = chi phí node cha đến con + chi phí của node Cha. F(x) = G(x) + H(x) từ đó sắp xếp lại thứ tự ưu tiên trong hàng đợi.

- Input:

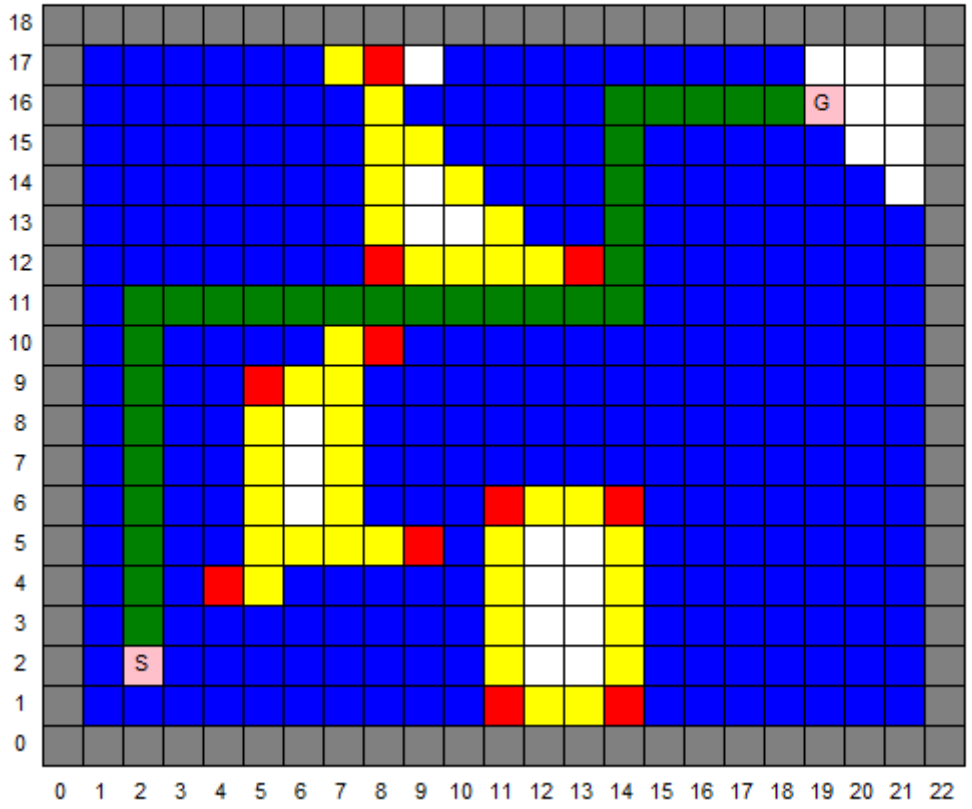
input.txt

```
22 18
2 2 19 16
3
4 4 5 9 8 10 9 5
8 12 8 17 13 12
11 1 11 6 14 6 14 1
```

- Output:

Cost of expanded node: 284

Cost of final path: 32



- Kết luận: Tìm ra được đường đi ngắn nhất từ Start đến Goal
- Ưu điểm: Nhanh, hiệu quả nhất trong các thuật toán.
- Nhược điểm: Tốc độ tìm kiếm của A* phụ thuộc vào độ chính xác của Heuristic.

TÀI LIỆU THAM KHẢO

[1] https://stackoverflow.com/questions/64602600/how-to-plot-out-grid-lines-in-python-turtle-module

[2] https://www.geeksforgeeks.org/iterative-deepening-searchids-iterative-deepening-depth-first-searchiddfs/

[3] https://www.geeksforgeeks.org/building-an-undirected-graph-and-finding-shortest-path-using-dictionaries-in-python/