BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM
KHOA CÔNG NGHỆ THÔNG TIN

MÔN HỌC: CƠ SỞ TRÍ TUỆ NHÂN TẠO
PROJECT 02: LOGIC

Giảng viên lý thuyết: **Bùi Duy Đăng**
Giảng viên thực hành: **Trần Quốc Huy**

Thành phố Hồ Chí Minh, 2023

MỤC LỤC

I. Giới thiệu:

STT	MSSV	Họ tên
1	21120439	Bùi Minh Duy
2	21120447	Nguyễn Nhật Hào
3	21120453	Tô Phương Hiếu
4	21120457	Lê Minh Hoàng

II. Phân bổ công việc:

Phần công việc	Công việc	Người thực hiện
Phần thuật toán	Tìm hiểu thuật toán	Bùi Minh Duy Nguyễn Nhật Hào Tô Phương Hiếu Lê Minh Hoàng
Phần giao diện	Cài đặt thuật toán	Nguyễn Nhật Hào
	Menu game phần introduce	Bùi Minh Duy
	Menu game about us, Menu start (chọn map)	Tô Phương Hiếu
	Màn hình chạy thuật toán	Nguyễn Nhật Hào
	Màn hình thông tin logic thuật toán	Lê Minh Hoàng
Thiết kế map	Map 1, map 2	Tô Phương Hiếu
	Map 3, map 4, map 5	Bùi Minh Duy
	Random Map	Nguyễn Nhật Hào
Báo cáo	Tổng hợp, trình bày	Lê Minh Hoàng

III. Môi trường:

- Ngôn ngữ: Python (version >= 3.0).

- Đồ họa: thư viện Pygame.

- Logic: thư viện python-sat và pysat.

IV. Cấu trúc thư mục và hướng dẫn thực thi:

1. Cấu trúc thư mục:

./Group_6

|_/Document

|_/Report.pdf

|_/Input

|_/map1.txt

|_/map2.txt

|_/map3.txt

|_/map4.txt

|_/map5.txt

|_/randMap.txt

|_/Output

|_/result1.txt

|_/result2.txt

|_/result3.txt

|_/result4.txt

|_/result5.txt

|_/resultRandMap.txt

|_/Source

|_/Assets

|_/Fonts

|_/Images

|_/Entity

|_/Board.py

|_/Menu

|_/Run

|_/Solution.py

main.py

Game.py

constants.py

utils.py

requirements.txt

Thư mục **Input** chứa thông tin các map của trò chơi.

Thư mục **Output** chứa thông tin **chi tiết logic** cho các map tương ứng ở folder **Input**.

Thư mục **Run** chứa file **Solution.py** là file giải quyết logic cho bài toán (ngoài ra còn có các file phụ trợ khác trong thư mục này).

Thư mục **Entity** chứa file **Board.py** là file quản lý giao diện đồ họa cho bài toán (ngoài ra còn có các file phụ trợ khác trong thư mục này).

File **requirements.txt** là file chứa tên packages cần cài đặt để thực thi chương trình.

File **main.py** là file chạy chính của project.

2. Thực thi chương trình:

Bước 1: Bật console **cùng cấp** với file **main.py** (trong thư mục **Source**).

Bước 2: Cài đặt package cần thiết trong file **requirements.txt** với lệnh sau:

pip install –r requirements.txt

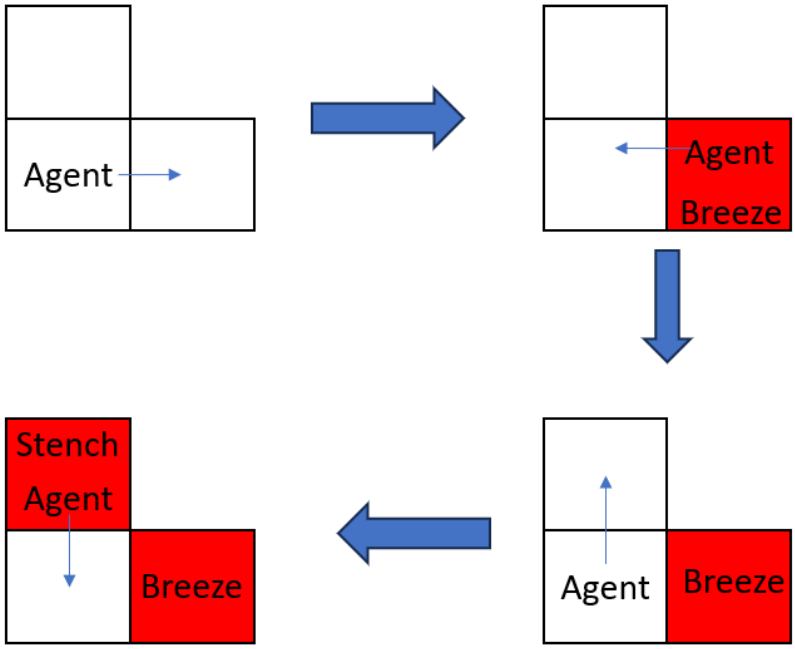
Bước 3 Để chạy chương trình sử dụng lệnh **py main.py** hoặc **python main.py**

V. Đánh giá:

STT	Yêu cầu	Thực hiện	Hoàn thành
1	Giải quyết vấn đề thành công	Sử dụng thuật toán Propositional Logic để tạo Knowledge Base cho agent.	100%
2	Biểu diễn đồ họa mỗi bước.	Sử dụng Pygame	100%
3	Tạo ít nhất 5 map có cấu trúc khác nhau về vị trí và số lượng Pit, Gold và Wumpus	Có hỗ trợ thêm tạo map ngẫu nhiên	100%
4	Báo cáo thuật toán		100%

VI. Thuật toán:

- Để duyệt qua tất cả các ô trong bản đồ, ta sẽ sử dụng thuật toán backtracking. Ví dụ, khi Agent đang đứng ở ô (1,1) sau đó di chuyển sang ô (1,2), Agent nhận thấy có Breeze, sau đó Agent sẽ quay lại ô (1,1) và tiếp tục di chuyển đến ô (2,1), Agent nhận thấy có Stench. Thuật toán hoạt động một cách đệ quy như hình dưới:



- Khi Agent di chuyển tới một ô

+ Nếu tại đó có Wumpus hoặc Pit, agent chết.

+ Nếu tại đó có vàng, Agent sẽ lấy vàng.

+ Nếu tại đó có Stench hay Breeze, Agent sẽ thêm một vài quy luận vào knowledge base (theo **Propositional Logic**):

- Ô đó có Stench hay Breeze hay không?
- Wumpus và Pit không thể cùng một ô: **(P ^ -W) v (-P ^ W)**
- Nếu vị trí hiện tại có Breeze thì bốn ô chung cạnh với ô đó có thể có ít nhất một Pit và ngược lại nếu bốn ô chung cạnh với ô hiện tại có ô có Pit thì ô này sẽ có Breeze: **B <=> Pa v Pb v Pc v Pd**.
- Nếu vị trí hiện tại không có Breeze thì bốn ô chung cạnh sẽ không có Pit**: -Pa ^ -Pb ^ -Pc ^ -Pd.**
- Nếu vị trí hiện tại có Stench thì bốn ô chung cạnh với ô đó có thể có ít nhất một Wumpus và ngược lại nếu bốn ô chung cạnh với ô hiện tại có ô có Wumpus thì ô này sẽ có Stench: **S <=> Wa v Wb v Wc v Wd.**

- Nếu vị trí hiện tại không có Stench thì bốn ô chung cạnh sẽ không có Wumpus: **-Wa ^ -Wb ^ -Wc ^ -Wd**

- Đi qua mỗi vị trí trên bản đồ, Agent sẽ lưu lại logic có được dựa trên KNOWLEDGE BASE ở trên.

- Đến vị trí mới nếu cảm nhận được Breeze hay Stench thì sẽ thực hiện dự đoán bốn ô chung cạnh dựa theo logic hiện có. Đối với ô có Breeze nếu dự đoán được ô chung cạnh có Pit thì mở ô đó. Đối với ô có Stench nếu dự đoán được ô chung cạnh có Wumpus thì dùng mũi tên bắn nó. Sau khi thực hiện dự đoán nếu vẫn còn cảm nhận được mùi hôi (Stench) thì Agent sẽ dùng mũi tên bắn lần lượt bốn ô chung cạnh đến khi không còn cảm nhận được mùi hôi.

- Sau mỗi lần Wumpus bị giết bởi Agent, Agent sẽ cập nhật lại những trạng thái của stench cũng như là Knowledge Base.

- Để có thể suy ra ô tiếp theo có Wumpus hay Pit không, nhóm quyết định dùng package **Glucose3** từ **pysat.solvers** để cài đặt hàm thực thi giải quyết vấn đề này:

  + Ý tưởng của việc sử dụng một solver SAT như Glucose3 cho quá trình suy luận được dựa trên việc chúng ta muốn suy luận "alpha", chúng ta cần chứng minh rằng cả KB và phủ định của "alpha" ("not alpha") không hợp lý, có nghĩa là không có mô hình nào thỏa mãn cả hai điều kiện này.

  + Khi ta gọi "g.solve()" trên Glucose3, nó cố gắng tìm kiếm một mô hình thỏa mãn các mệnh đề đã thêm vào. Nếu nó thành công trong việc tìm kiếm một mô hình, điều đó có nghĩa là tồn tại một phân công giá trị đúng cho các biến sao cho cả KB và not alpha đều được thoả mãn. Do đó, ta không thể suy luận "alpha" từ cơ sở tri thức.

  + Ngược lại, nếu "g.solve()" trả về "False" (không thể thoả mãn), nghĩa là không có mô hình nào thỏa mãn cả KB và phủ định của "not alpha". Trong trường hợp này, ta có thể suy luận "alpha" từ KB vì không có tình huống nào mà "alpha" sai dựa trên tri thức hiện tại.

Nhận xét:

+ Khi cảm thấy Stench nếu không nhận biết được vị trí chính xác của Wumpus, Agent sẽ thực hiện bắn tất cả các ô chưa biết xung quanh đến khi hết cảm nhận được Stench việc này làm cho điểm số bị trừ đi đáng kể (trong trường hợp xấu nhất là 300 điểm), thay vì bỏ qua ô này mà đi đến các ô an toàn khác để có thêm logic, song việc này nhằm đảm bảo Agent có thể mở nhiều ô chưa biết nhất vì có thể loại bỏ Wumpus và Stench một cách nhanh chóng và làm giảm logic bất lợi - nhóm đã quyết định ưu tiên việc này hơn.

+ Nếu có nhiều Pit bao quanh vị trí của Agent, việc này sẽ ảnh hưởng đến logic của Agent do sẽ có nhiều Breeze bao quanh Agent (tạo thành vòng khép kín), Agent sẽ không có cơ sở để đi tiếp nên Agent sẽ thực hiện việc **`climb out of the cave`** mặc dù bản đồ chưa được khám phá hết. Đối với Stench do khi cảm thấy Stench thì luôn thực hiện việc bắn mũi tên nên ngăn chặn được tình trạng trên.

HẾT

Tài liệu tham khảo

- Pygame docs (https://www.pygame.org/docs/).

- PySAT documentation (https://pysathq.github.io/docs/html/index.html).

- Project khóa trước (https://github.com/kieuconghau/ai-wumpus-world).

- Wumpus World Simulator (https://thiagodnf.github.io/wumpus-world-simulator/).