THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Báo cáo đồ án 2

ĐỀ TÀI

WUMPUS WORLD

Phan Tấn Đạt 18127078

Trần Phước Lộc 18127130

Môn: Cơ sở trí tuệ nhân tạo

Thành phố Hồ Chí Minh – 2020

I. Thông tin nhóm và phân công công việc

Họ và tên	MSSV	Công việc
Phan Tấn Đạt	18127078	Làm UI, đọc file, xử lí việc thay đổi dữ liệu trên bảng (nhặt vàng, loại bỏ stench khi bắn chết wumpus), góp ý và tham gia phần FOL
Trần Phước Lộc	18127130	Map generator, Think, KB initialize, KB update each move, KB asking.

II. Môi trường biên dịch thuật toán:

- IDE: Pycharm.
- Ngôn ngữ sử dụng: Python.
- Tham khảo: Thư mục aima chứa các hàm và thư viện tham khảo từ các nguồn sau:
  - https://github.com/aimacode/aima-python
  - https://www.annytab.com/first-order-logic-in-python/
  - http://www.sista.arizona.edu/~clayton/courses/ai/projects/wumpus/
  - https://www.annytab.com/wp-content/uploads/2020/02/aima-code.zip

III. Ý tưởng và Thực hiện

- Về KB trong project: Sử dụng class FolKB từ nguồn tham khảo. Các lớp và phương thức thường sử dụng:
  - Class Fol KB chứa các phương thức:
    - def tell(self, sentence): Thêm vào FolKB một sentence.
    - def ask_generator(self, query): Trả về kết quả backward chaining.
    - def def retract(self, sentence): Xóa bỏ sentence trong KB.
  - Hàm: def expr(x): Đưa x về dạng đúng cho logic.FolKB sử dụng.
- Ý tưởng KB:

- Ban đầu KB sẽ được khởi tạo với các clauses của logic game:

```
sen = ["(ADJ(x, y) & ADJ(x, z) & ADJ(x, t) & ADJ(x, v) & B(y) & B(z) & B(t) & B(v)) ==> PIT(x)",
       "(ADJ(x, y) & ADJ(x, z) & ADJ(x, t) & ADJ(x, v) & S(y) & S(z) & S(t) & S(v)) ==> WUM(x)",
       "(EDGE(x) & ADJ(x, y) & ADJ(x, z) & ADJ(x, t) & S(y) & S(z) & S(t)) ==> WUM(x)",
       "(EDGE(x) & ADJ(x, y) & ADJ(x, z) & ADJ(x, t) & B(y) & B(z) & B(t)) ==> PIT(x)",
       "(CORNER(x) & ADJ(x, y) & ADJ(x, z) & S(y) & S(z)) ==> WUM(x)",
       "(CORNER(x) & ADJ(x, y) & ADJ(x, z) & B(y) & B(z)) ==> PIT(x)",
       "(ADJ(x, y) & NULL(x)) ==> SAFE(y)",
       "B(x) ==> SAFE(x)",
       "S(x) ==> SAFE(x)",
       "NULL(x) ==> SAFE(x)"]
```

- Giải thích các hàm và quan hệ:
    - ADJ(x, y): x kề y.
    - B(x): room x có Breeze.
    - S(x): room x có Stench.
    - PIT(x): room x có Pit.
    - WUM(x): room x có Wumpus.
    - EDGE(x): x là phòng ở rìa(cạnh) của map.
    - CORNER(x): x là phòng ở góc của map.
    - SAFE(x): room x an toàn để đặt chân vào.
    - NULL(x): room x không có dấu hiệu nguy hiểm.
    - EXP(x): room x đã được AI khám phá(đi qua).

- Giải thích các câu:
    - "(ADJ(x, y) & ADJ(x, z) & ADJ(x, t) & ADJ(x, v) & B(y) & B(z) & B(t) & B(v)) ==> PIT(x)": 4 room kề với x có Breeze thì x là room chứa PIT.
    - "(ADJ(x, y) & ADJ(x, z) & ADJ(x, t) & ADJ(x, v) & S(y) & S(z) & S(t) & S(v)) ==> WUM(x)": 4 room kề với x có Stench thì x là room chứa WUMPUS.
    - "(EDGE(x) & ADJ(x, y) & ADJ(x, z) & ADJ(x, t) & S(y) & S(z) & S(t)) ==> WUM(x)": nếu room x ở cạnh của map và các phòng nằm kề chung quanh 3 hướng còn lại có Stench thì room x có Wumpus.
    - "(EDGE(x) & ADJ(x, y) & ADJ(x, z) & ADJ(x, t) & B(y) & B(z) & B(t)) ==> PIT(x)": nếu room x ở cạnh của map và các phòng nằm kề chung quanh 3 hướng còn lại có Breeze thì room x có Pit.
    - "(CORNER(x) & ADJ(x, y) & ADJ(x, z) & S(y) & S(z)) ==> WUM(x)": nếu room x ở góc của map và các phòng nằm. kề chung quanh 2 hướng còn lại có Stench thì room x có Wumpus.
    - "(CORNER(x) & ADJ(x, y) & ADJ(x, z) & B(y) & B(z)) ==> PIT(x)": nếu room x ở góc của map và các phòng nằm kề chung quanh 2 hướng còn lại có Breeze thì room x có Pit.

- "(ADJ(x, y) & NULL(x)) ==> SAFE(y)": room kề 1 room NULL(không thấy dấu hiệu Breeze hay Stench) thì an toàn.
- "B(x) ==> SAFE(x)": room chứa Breeze thì an toàn.
- "S(x) ==> SAFE(x)": room chứa Stench thì an toàn.
- "NULL(x) ==> SAFE(x)": room NULL(không thấy dấu hiệu Breeze hay Stench trong room này) thì an toàn.

- Giải thích quá trình chạy thuật toán trên UI:
  - Sau khi compile file *UI.py*, file input với format theo yêu cầu đề bài đặt trong folder *../Input/* sẽ được load vào thành 1 object của class *Map* được định nghĩ trong file *File_loader.py.*
  - Vị trí bắt đầu của Agent trong Map sẽ được random bằng function random_spawning_location() trong class *Map* để lấy 1 vị trí bất kì trên Map đã được load vào không phải Pit hay ổ Wumpus.
  - UI được thực hiện bằng module Turtle trong các class *Room* và *Player*. Ngoài ra còn có class *ScreenMessage* để in ra các dữ liệu (Tọa độ, số bước đi, số điểm, các thứ tìm thấy tại tọa độ đó…).
  - Khi Player thực hiện việc di chuyển đến các phòng mới, các mệnh đề sẽ được "Tell" cho KB để phục vụ việc suy đoán các hành động tiếp theo (Bắn tên, hướng di chuyển, bỏ cuộc).
  - Chương trình sẽ lập tức ngừng nếu Agent đi vào phòng có Pit hay Wumpus.

- Mô tả cách Player Agent hoạt động:
  - Trước khi làm gì đó shoot, về or move.
  - AI sẽ check xem từ trong KB còn có thể suy ra ô nào SAFE để đi không.
  - Nếu không còn ô nào SAFE, AI sẽ tìm ô có MÙI và truy diệt wumpus để tạo đường.
  - Nếu có ô SAFE thì dùng thuật toán BFS tìm đường và di chuyển đến đó để tránh lặp lại các bước đi ở các ô đã khám phá.
  - Nếu không MÙI và không ô nào SAFE thì VỀ NHÀ.
  - Mỗi khi di chuyển KB sẽ được update bằng cách tell các fact như SAFE ,WUM, PIT, S, B để phục vụ việc suy luận các hành động kế tiếp.
  - Mỗi lần think AI đưa ra một chuỗi hành động.

IV. Mức độ hoàn thành và nhận xét

Do thời gian và kinh nghiệm có hạn, chúng em xin tự đánh giá mức độ hoàn thành là 90% vì còn chưa tối ưu bởi thời gian để đưa ra chuỗi hành động tiếp theo khá lâu và đôi khi số điểm khi kết thúc trò chơi là âm.