BÀI TẬP DAEDLINE TUẦN 2 – TÌM HIỂU GAME PACMAN

Họ tên: Nguyễn Sanh Tuấn.

MSSV: 1760227.

Lớp: 17CK2 (Ca 3 chiều thứ 7).

Môn: CTTTMNT&UD.

1. **Cài đặt và thử nghiệm thuật toán DFS.**
   Tiến hành cài đặt hoàn chỉnh hàm deapthFirstSearch trong search.py
   - Sử dụng lại Stack trong file util.py
   - Đầu tiên, tiến hành gọi hàm start = problem.getStartState() để lấy trạng thái bắt đầu, problem.isGoalState(c) với c = problem.getStartState() để kiểm tra trạng thái state có phải là trạng thái đích không.
   - Đầu vào: trạng thái bắt đầu, hàm successor, hàm kiểm tra trạng thái đích.
   - Dầu ra: kế hoạch tìm được(chuỗi các hành động để đi từ trạng thái bắt đầu đến trạng thái đích).
   - Quá trình thực hện của cài đặt thuật toán DFS:
   - Khởi tạo:
     - Frigne: trạng thái bắt đầu của thuật toán tìm kiếm.
     - Closed set: rỗng.
   - Trong khi frigne chưa rỗng:
     - Lấy một kế hoạch ra khỏi frigne theo bộ.
     - Nếu kế hoạch đi đến tới đích(dùng problem.isGoalState(c)): return action + [direction].
   - Nếu trạng thái cuối của kế hoạch chưa có trong closed set:
     - Đưa trạng thái vào closed set.
     - Mở rông bằng hàm successor=problem.getSuccessor(state) và đưa kế hoạch vào fringe.
   - Nếu ra khỏi vào lặp: thuật toán không có lời giải.

   Code của thuật toán DFS.

```
def depthFirstSearch(problem):
    """
    Search the deepest nodes in the search tree first.

    Your search algorithm needs to return a list of actions that reaches the
    goal. Make sure to implement a graph search algorithm.

    To get started, you might want to try some of these simple commands to
    understand the search problem that is being passed in:

    print("Start:", problem.getStartState())
    print("Is the start a goal?", problem.isGoalState(problem.getStartState())
)
    print("Start's successors:", problem.getSuccessors(problem.getStartState()
))
    """
    "*** YOUR CODE HERE ***"
```

```
    util.raiseNotDefined()
    start = problem.getStartState()
    goal = problem.getStartState()
    exploredState = []
    exploredState.append(start)
    states = util.Stack()
    stateTuple = (start, [])
    states.push(stateTuple)
    while not states.isEmpty() and not problem.isGoalState(goal):
        state, actions = states.pop()
        exploredState.append(state)
        successor = problem.getSuccessors(state)
        for i in successor:
            coordinates = i[0]
            if not coordinates in exploredState:
                goal = i[0]
                direction = i[1]
                states.push((coordinates, actions + [direction]))
    return actions + [direction]
```

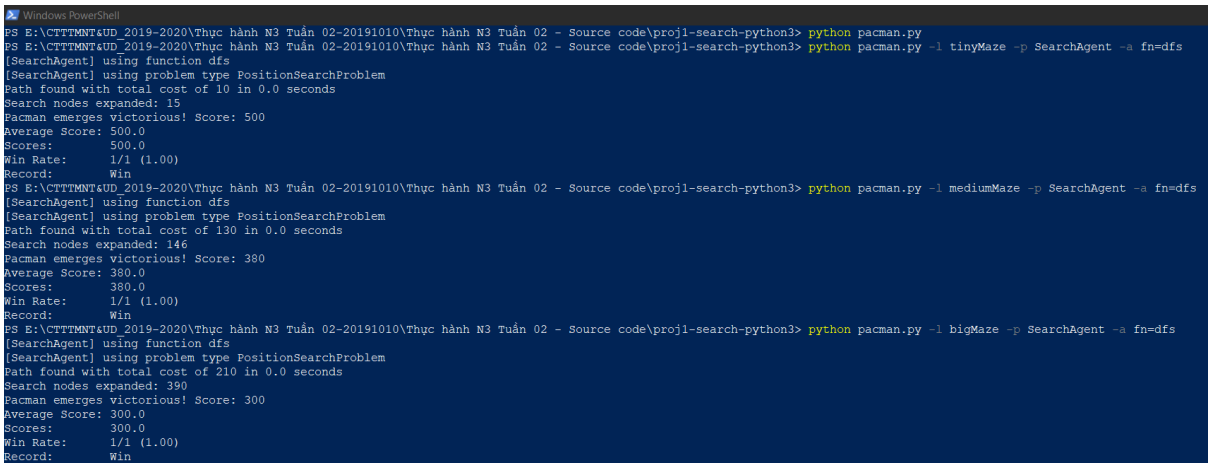
problem.getSuccessors(state) để mở trạng thái state (phương thức này trả về một dah sách, trong đó mỗi phần tử là một bộ (succerssor, action, coordinates) với successor là trạng thái mà có thể đi đến được từ trạng thái state hiện tại bằng cách thực hiện hành động action, và chi phí của việc vận chuyển trạng thái này là stepCost).

Hình ảnh kiểm tra chạy thử nghiệm thuật toán DFS trên game pacman:

- Mê cung tinyMaze: ***python pacman.py -l tinyMaze -p SearchAgent -a fn=dfs***

- Mê cung mediumMaze: ***python pacman.py -l mediumMaze -p SearchAgent -a fn=dfs***

- Mê cung bigMaze: ***python pacman.py -l bigMaze -p SearchAgent -a fn=dfs***



2. **Khung chương trình tìm kiếm.**

- *Diễn giải khung chương trình tìm kiếm:*
    - Pacman là Gamestate.
    - Pacman chỉ định trạng thái trò chơi đầy đủ, bao gồm thức ăn, viên nang, cấu hình tác nhân và thay đổi điểm số.
    - Dữ liệu game được lưu trong classGameState, dùng để tạo gói dữ liệu mới bằng cách sao chép thông tin từ người tiền nhiệm của nó.

- Pacman hoạt động theo cơ chế Stack & Queue.

- Lưu ý rằng trong Pacman cổ điển, Pacman luôn là tác nhân 0.
- Tạo một cấu hình mới đạt được bằng cách dịch hiện tại cấu hình bằng vectơ hành động. Đây là một cuộc gọi cấp thấp và không cố gắng tôn trọng tính hợp pháp của phong trào.
- Hành động là các vectơ chuyển động.
- Quy ước bước đi của Pacman:
  - Mảng 2 chiều của các đối tượng được hỗ trợ bởi một danh sách các danh sách. Dữ liệu được truy cập thông qua lưới [x] [y] trong đó (x, y) là các vị trí trên bản đồ Pacman với x ngang, y dọc và gốc (0,0) ở góc dưới bên trái.
  - Cấu hình giữ tọa độ (x, y) của một ký tự, cùng với nó hướng đi.
  - Quy ước cho các vị trí, giống như biểu đồ, là (0,0) là góc dưới bên trái, x tăng theo chiều ngang và y tăng theo chiều dọc. Do đó, hướng bắc là hướng tăng y, hoặc (0,1).

- *Cách thức hoạt động của chương trình:* đầu tiên, SearcgAgent sẽ sử dụng các thuật toán được cài đặt để lên kế hoạch trong đầu; sau khi đã tìm ra kế hoạch (chuỗi các hành động để đi từ trạng thái bắt đầu đến trạng thái đích), SearchAgent sẽ thực thi kế hoạch này. Khi chạy, ta sẽ thấy các ô ở mê cung có màu đỏ với mức đô đậm khác nhau; mức độ đậm này cho biết thứ tự mở các ô khi chạy thuật toán tìm kiếm: các ô có màu đỏ càng đậm thì được mở càng sớm.
- *Mục đích của từng file:*
  - search.py: nơi cài đặ các thuật toán tìm kiếm. Trong này có 1 lớp phác thảo cấu trúc của một vấn đề tìm kiếm, bất kì phương thức nào (theo thuật ngữ hướng đối tượng: một lớp trừu tượng).
  - searchAgent.py: là nơi chứa tất cả các tác nhân có thể được chọn để kiểm soát Pacman. Gồm môt thuật toán cho một vấn đề tìm kiếm được cung cấp, sau đó trả về các hành động để theo con đường đó.
  - game.py: nơi xây dựng các điều kiện bước đi của pacman. Gồm các class về tọa độ bước đi, thời gian, điều kiện thắng, ...
  - pacman.py: giữ logic cho trò chơi pacman cổ điển cùng với chính mã để chạy một trò chơi. Tập tin này được chia làm ba phần:
    Giao diện của bạn với thế giới pacman: Pacman là một môi trường phức tạp
    Những bí mật ẩn giấu của pacman: Phần này chứa tất cả các mã logic mà pacman
    môi trường sử dụng để quyết định ai có thể di chuyển ở đâu, ai chết khi nào những thứ va chạm,...
    Khung để bắt đầu trò chơi: Phần cuối cùng chứa mã để đọc lệnh.
  - untils.py: chứa các dữ liệu hữu ích để triển khai SearchAgents.

----------THE END----------