Tìm kiếm đối kháng

1.Thuật toán cắt tỉa ampla và Beta(Alpha-Beta Pruning)

Xin chào thầy và các bạn, hôm nay mình sẽ viết báo cáo về thuật toán cải tiến về mặt giảm bớt nhánh của tìm kiếm hôm trước- tìm kiếm Minimax.Đó chính là ***thuật toán Alpha-Beta Pruning***

Trong thuật toán này,mình sẽ viết code vào **hàm getAction()**, trong **class AlphaBetaAgent**. Và có thể viết thêm các hàm phụ để hỗ trợ cho hàm **getAction()** nếu cần thiết.

Cắt tỉa *Alpha-beta* sẽ giúp loại bỏ những không gian trạng thái không cần thiết và hỗ trợ tối ưu hóa thuật toán tìm kiếm *Minimax*.

-Hình ảnh minh hoạ code:

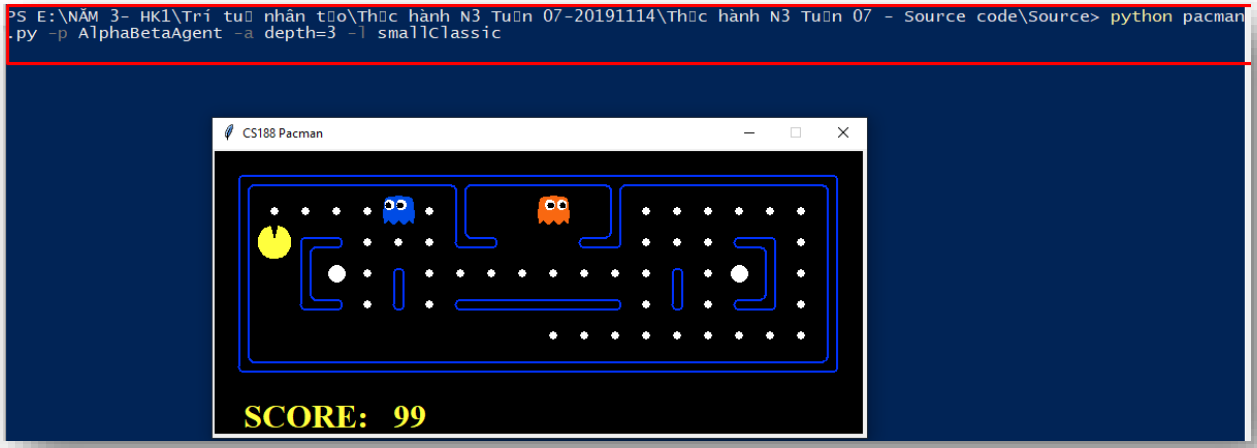
```
class AlphaBetaAgent(MultiAgentSearchAgent):
    """
      Your minimax agent with alpha-beta pruning (question 3)
    """

    def getAction(self, gameState):
        """
          Returns the minimax action using self.depth and self.evaluationFunction
        """
        "*** YOUR CODE HERE ***"
        def maximizer(agent, depth, game_state, a, b):  # maximizer function
            v = float("-inf")
            for newState in game_state.getLegalActions(agent):
                v = max(v, alphabetaprune(
                    1, depth, game_state.generateSuccessor(agent, newState), a, b))
                if v > b:
                    return v
                a = max(a, v)
            return v
```

```
def minimizer(agent, depth, game_state, a, b):  # minimizer function
    v = float("inf")

    # calculate the next agent and increase depth accordingly.
    next_agent = agent + 1
    if game_state.getNumAgents() == next_agent:
        next_agent = 0
    if next_agent == 0:
        depth += 1

    for newState in game_state.getLegalActions(agent):
        v = min(v, alphabetaprune(next_agent, depth,
                                  game_state.generateSuccessor(agent, newState), a, b))
        if v < a:
            return v
        b = min(b, v)
    return v
```

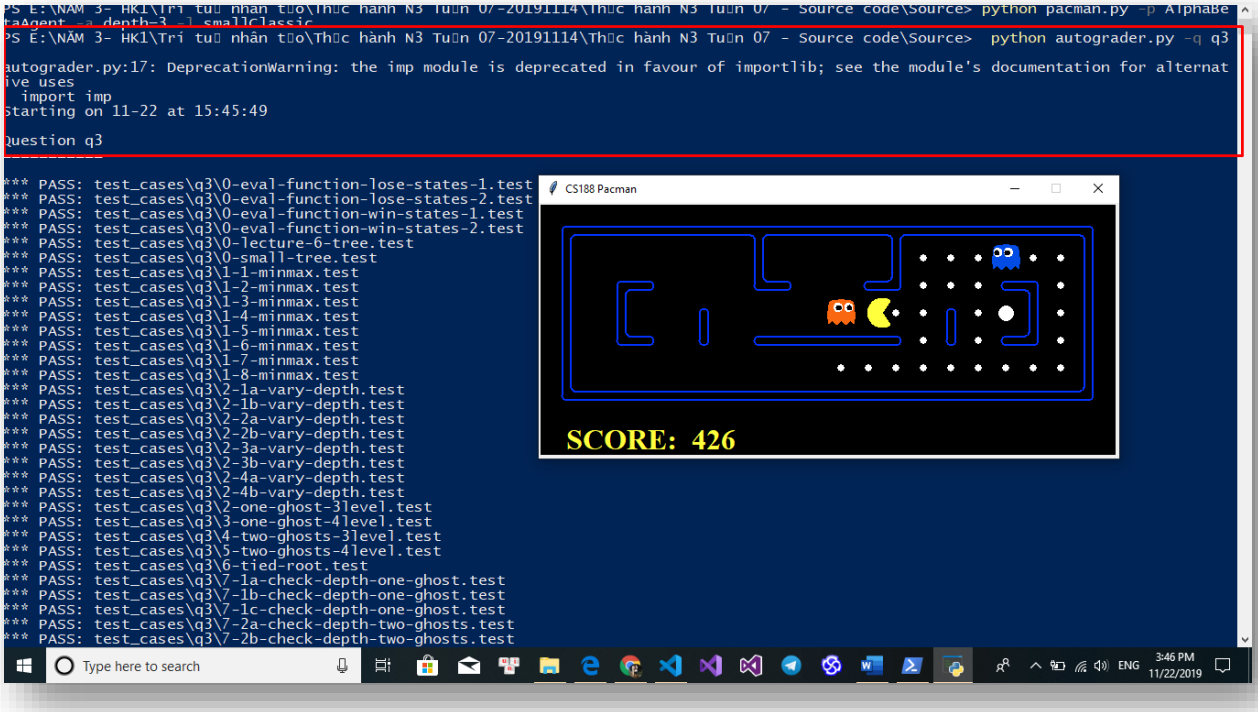
```
def alphabetaprune(agent, depth, game_state, a, b):
    # return the utility in case the defined depth is reached or the game is won/lost.
    if game_state.isLose() or game_state.isWin() or depth == self.depth:
        return self.evaluationFunction(game_state)

    if agent == 0:  # maximize for pacman
        return maximizer(agent, depth, game_state, a, b)
    else:  # minimize for ghosts
        return minimizer(agent, depth, game_state, a, b)

"""Performing maximizer function to the root node i.e. pacman using alpha-beta pruning."""
utility = float("-inf")
action = Directions.WEST
alpha = float("-inf")
beta = float("inf")
for agentState in gameState.getLegalActions(0):
    ghostValue = alphabetaprune(
        1, 0, gameState.generateSuccessor(0, agentState), alpha, beta)
    if ghostValue > utility:
        utility = ghostValue
        action = agentState
    if utility > beta:
        return utility
    alpha = max(alpha, utility)

return action
```

-Demo chạy bằng cách chạy l

+ *python pacman.py -p AlphaBetaAgent -a depth=3 -l smallClassic*

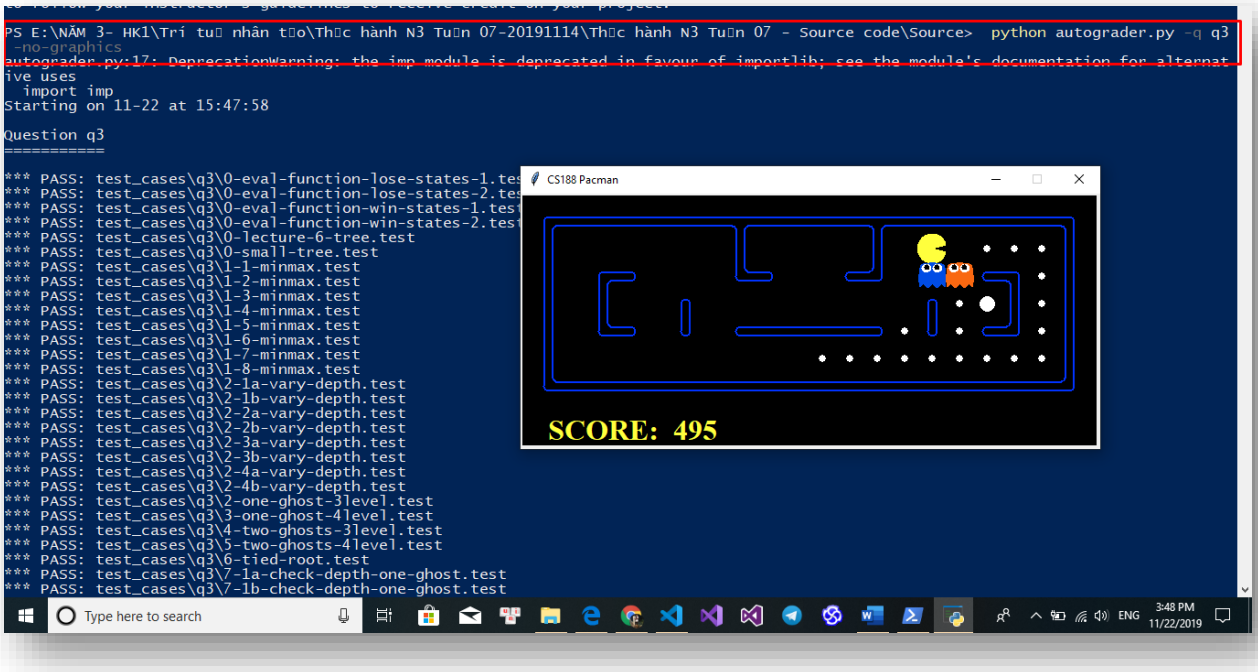


+ python autograder.py -q q3



```
*** Running AlphaBetaAgent on smallClassic 1 time(s).
Pacman died! Score: 84
Average Score: 84.0
Scores:        84.0
Win Rate:      0/1 (0.00)
Record:        Loss
*** Finished running AlphaBetaAgent on smallClassic after 31 seconds.
*** Won 0 out of 1 games. Average score: 84.000000 ***
*** PASS: test_cases\q3\8-pacman-game.test

### Question q3: 5/5 ###


Finished at 15:46:21

Provisional grades
==================
Question q3: 5/5
------------------
Total: 5/5
```

+ python autograder.py -q q3 –no-graphics



Lời kết: Thực hiện đúng alpha-beta pruning, Pacman sẽ có thể bị ma tiêu diệt trong một số bộ test. Đây không phải lá vấn đề do chương trình của bạn, mà là do thuật toán alpha-beta chưa được tối ưu.