

# Tìm kiếm đối kháng

## 1. Thuật toán cắt tỉa alpha 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 họa 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, alphabeta(agent, 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, alphabeta prune(next_agent, depth,
                                   game_state.generateSuccessor(agent, newState), a, b))
        if v < a:
            return v
        b = min(b, v)
    return v

```

```

def alphabeta prune(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)

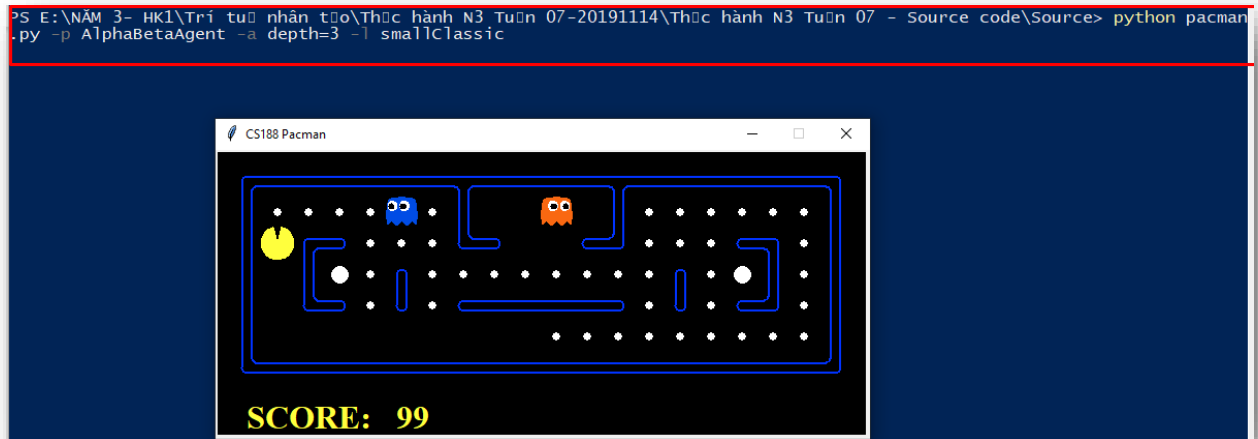
"""Performing maximizer function to the root node i.e. pacman using alpha-beta pruning."""
utility = float("-inf")
action = Directions.WEST
alpha = float("-inf")
beta = float("inf")
for agentState in gameState.getLegalActions(0):
    ghostValue = alphabeta prune(
        1, 0, gameState.generateSuccessor(0,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

return action

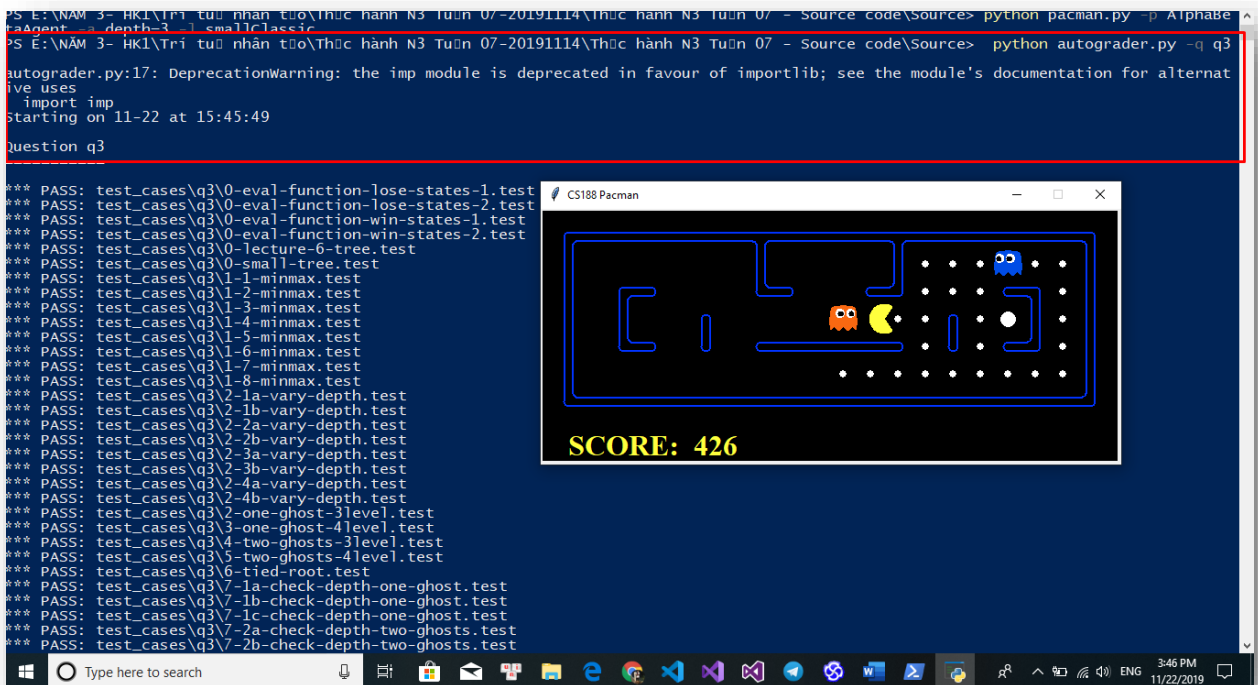
```

-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

```

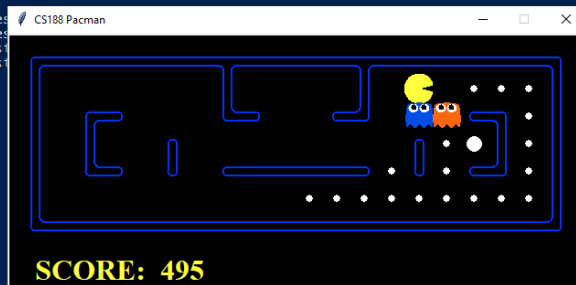
+ `python autograder.py -q q3 --no-graphics`

```

PS E:\NĂM 3- HK1\Trí tuệ nhân tạo\Thực hành N3 Tuần 07-20191114\Thực hành N3 Tuần 07 - Source code\Source> python autograder.py -q q3
--no-graphics
autograder.py:17: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
import imp
Starting on 11-22 at 15:47:58

Question q3
=====
*** PASS: test_cases\q3\0-eval-function-lose-states-1.test
*** PASS: test_cases\q3\0-eval-function-lose-states-2.test
*** PASS: test_cases\q3\0-eval-function-win-states-1.test
*** PASS: test_cases\q3\0-eval-function-win-states-2.test
*** PASS: test_cases\q3\0-lecture-6-tree.test
*** PASS: test_cases\q3\0-small-tree.test
*** PASS: test_cases\q3\1-1-minmax.test
*** PASS: test_cases\q3\1-2-minmax.test
*** PASS: test_cases\q3\1-3-minmax.test
*** PASS: test_cases\q3\1-4-minmax.test
*** PASS: test_cases\q3\1-5-minmax.test
*** PASS: test_cases\q3\1-6-minmax.test
*** PASS: test_cases\q3\1-7-minmax.test
*** PASS: test_cases\q3\1-8-minmax.test
*** PASS: test_cases\q3\2-1a-vary-depth.test
*** PASS: test_cases\q3\2-1b-vary-depth.test
*** PASS: test_cases\q3\2-2a-vary-depth.test
*** PASS: test_cases\q3\2-2b-vary-depth.test
*** PASS: test_cases\q3\2-3a-vary-depth.test
*** PASS: test_cases\q3\2-3b-vary-depth.test
*** PASS: test_cases\q3\2-4a-vary-depth.test
*** PASS: test_cases\q3\2-4b-vary-depth.test
*** PASS: test_cases\q3\2-one-ghost-3level.test
*** PASS: test_cases\q3\3-one-ghost-4level.test
*** PASS: test_cases\q3\4-two-ghosts-3level.test
*** PASS: test_cases\q3\5-two-ghosts-4level.test
*** PASS: test_cases\q3\6-tied-root.test
*** PASS: test_cases\q3\7-1a-check-depth-one-ghost.test
*** PASS: test_cases\q3\7-1b-check-depth-one-ghost.test

```



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