Report

Project 2

Nguyễn Văn Hậu –20127493

Table of contents

Mục lục

1 Introduce

1.1 Student information:

ID	Full name
20127493	**Nguyễn Văn Hậu**

Completion level: 100%

1.2 Information report:

- In this project, students research and implement the adversarial searching algorithm
- In addition, students implement an application (tic-tac-toe problem) and apply the adversarial technique to solve that tic-tac-toe.

2 Description of the algorithm

Minimax algorithm :

- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally
- Mini-max mostly used for the game playing with AI such as chess,tic tac toe
- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree

Properties of Mini-Max

- Complete: algorithm is complete. If finite search tree or definitely find a solution (if exist)
- Optimal: yes
- Time complexity : O(b^m) b:the legal move, m: maximum of depth
- Space complexity: O(bm) b:the legal move, m: maximum of depth

Pseudo code

```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))
```

```
function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← -∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v
```

```
function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

- **principles:**
  - In this algorithm two players play the game, one is called MAX and other is called MIN.
  - Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
  - The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
  - The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Pruning Alpha-Beta:

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm
- Pruning Alpha-Beta is a teachnique by which without checking each node of search tree we can compute the correct mini-max decision, base on 2 parameter alpha ,beta
- Alpha The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞.
- Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The

initial value of beta is +∞.

- The algorithm will removes all the nodes which are not really affecting the final decision but making algorithm slow

- **Pesudo code**

```
function ALPHA-BETA-SEARCH(state) returns an action
    v ← MAX-VALUE(state,-∞,+∞)
    return the action in ACTIONS(state) with value v
```

```
function MAX-VALUE(state,α,β) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← -∞
    for each a in ACTIONS(state) do
        v ← MAX(v, MIN-VALUE(RESULT(s,a),α,β))
        if v ≥ β then return v
        α ← MAX(α, v)
    return v
```

```
function MIN-VALUE(state,α,β) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← +∞
    for each a in ACTIONS(state) do
        v ← MIN(v, MAX-VALUE(RESULT(s,a) ,α,β))
        if v ≤ α then return v
        β ← MIN(β, v)
    return v
```

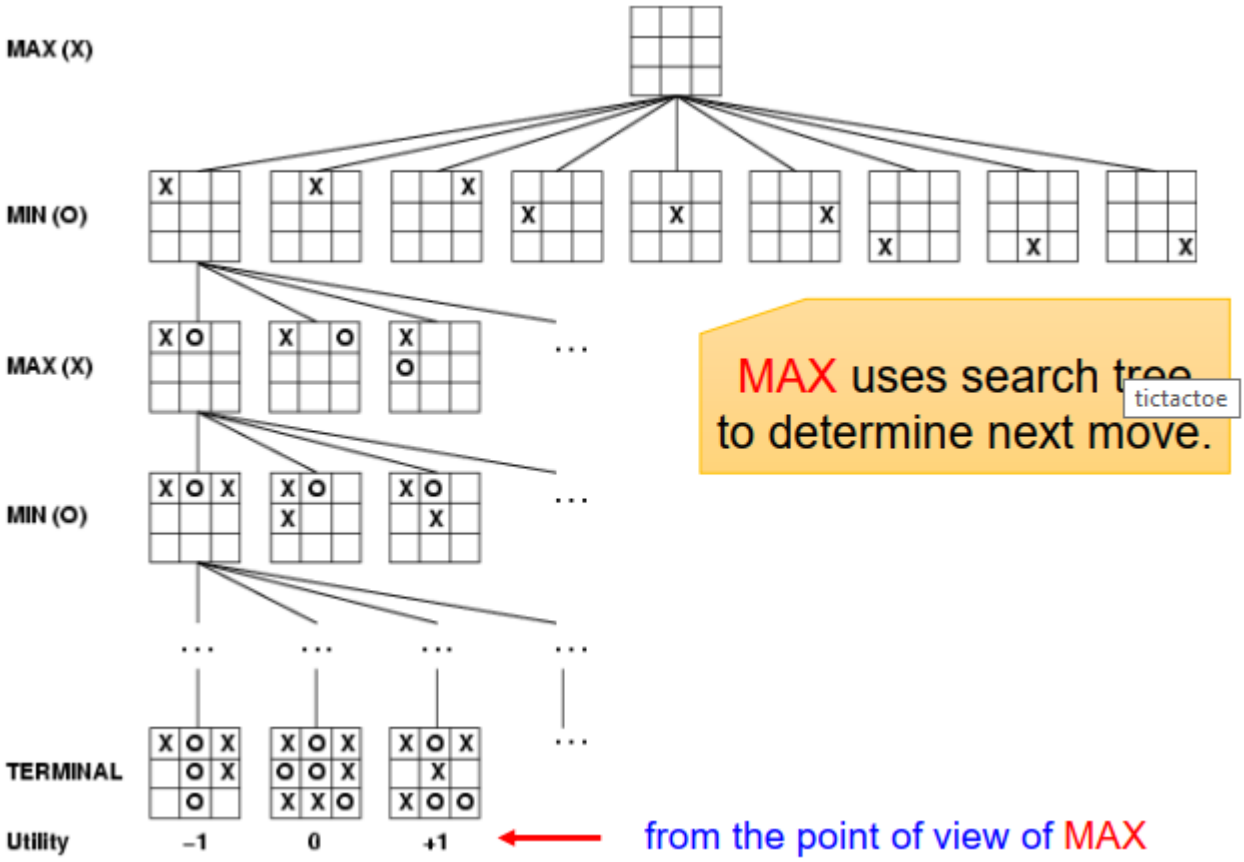
- **Case**
  - Good move ordering: time complexity $O(b^{(m/2)})$ → x2 search depth
  - Worst move ordering: time complexity $O(b^m)$

3 Apply algorithm to the tic tac toe

In this project, I will use mini-max algorithm

- Initial state: It specifies how the game is set up at the start.
- Player(s): It specifies which player has moved in the state space.
- Action(s): It returns the set of legal moves in state space
- Result(s,a): It is the transition model, which specifies the result of moves in the state space.
- Terminal-Test(s): true if the game is over, else it is false at any case.
- Utility(s,p): for tic-tac-toe, utility values are +1, -1, and 0.

Game tree:

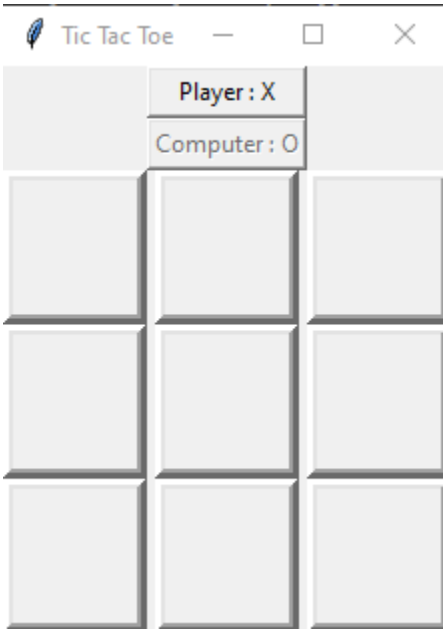


4 User manual

1. Run program
2. The gui of game will appear

- Play(player): player play first
- Play(robot): robot play first
- Exit : is to exit the game

3. Interface



5 Reference

1. Interface Game: https://www.geeksforgeeks.org/tic-tac-toe-game-with-gui-using-tkinter-in-python/
2. Minimax algorithm: https://github.com/javacodingcommunity/TicTacToeAI-with-Minimax
3. Report: https://www.javatpoint.com/ai-adversarial-search
4. Video demo: https://drive.google.com/file/d/1AqXOgDMYX_qAQRNfp4Np0IbKXFm6Xjib/view?usp=sharing