Alexandria University
Faculty of Engineering
CSED 2025

Report

AI lab 2

Submitted to

Eng. Mohamed Elhabebe

Faculty of engineering Alex. University

Submitted by

Islam Yasser Mahmoud 20010312

David Michel Nagib 20010545

Marwan Yasser Sabry 20011870

Mkario Michel Azer 20011982

Faculty of engineering Alex. University

Table of Contents

Lab overview

Connect 4 is a two-player game in which the players first choose a color and then take turns. dropping their colored discs from the top into a grid. The pieces fall straight down, occupying. the next available space within the column. The objective of the game is to connect-four of one's own discs of the same color next to each other vertically, horizontally, or diagonally. The two players keep playing until the board is full. The winner is the player having greater. number of connected fours.

Data Structures:

Node:

- The `Node` class represents nodes in the minimax tree.
- It contains three attributes: `value` (representing the value of the node), `identifier` (used for identification), and `children` (a list of child nodes).
- Nodes are used to build the minimax tree, storing values for each state and their relationships in the game tree.
- The `children` attribute is a list that holds the child nodes, representing the possible future states from the current state.

Algorithms:

Minimax:

- Max-value :

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, min-value(successor))
    return v
```

- Min-value:

```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, max-value(successor))
    return v
```

Minimax w/ pruning:

- Max-value:

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

- Min-value:

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

Details and Assumptions:

- The algorithm explores the game tree recursively, evaluating the possible outcomes at each level.
- The `max_value` and `min_value` methods in each class represent the maximizing and minimizing steps, respectively.

1. State Representation:

   The state is a character array of length 42 representing each cell in the board. The state class has the methods `get_heuristic()`, this is used to evaluate its heuristic in case we arrive at the maximum depth of the tree without reaching leaves ,and `get_utility()` is used to find the utility of the state at the leaf. These two functions are used to determine the determine the desirabilityof the state.

2. Node Representation:

   The `Node` class is used to represent the nodes in the minimax tree. Each node has a value, identifier, and children, forming a tree structure.

3. Depth Limit:

   The `maxTreeDepth` parameter determines the maximum depth the algorithm searches in the game tree. Beyond this depth, the algorithm uses heuristic evaluation.

4. Alpha-Beta Pruning:

   The algorithm assumes that alpha-beta pruning is correctly implemented to efficiently prune branches of the game tree that do not affect the final decision.

5. Successor Generation:

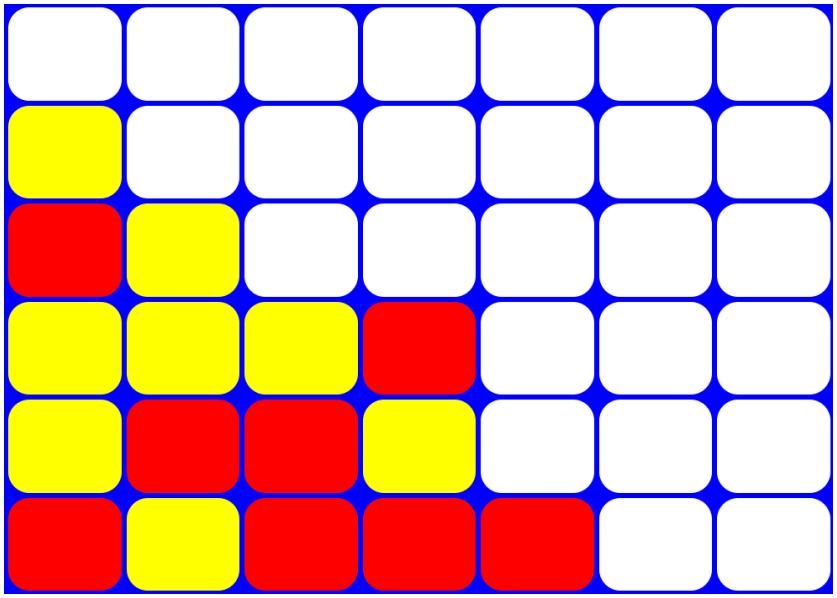
   The algorithm assumes that the `get_neighbors()` method of the `State` class correctly generates successor states for a given state. It returns a list of states, each of which correspondsto a circle in a different column.

Note: The specific details of the `State` class and its methods are not provided, so some assumptions are made based on common practices in implementing game-playing algorithms

Sample runs:

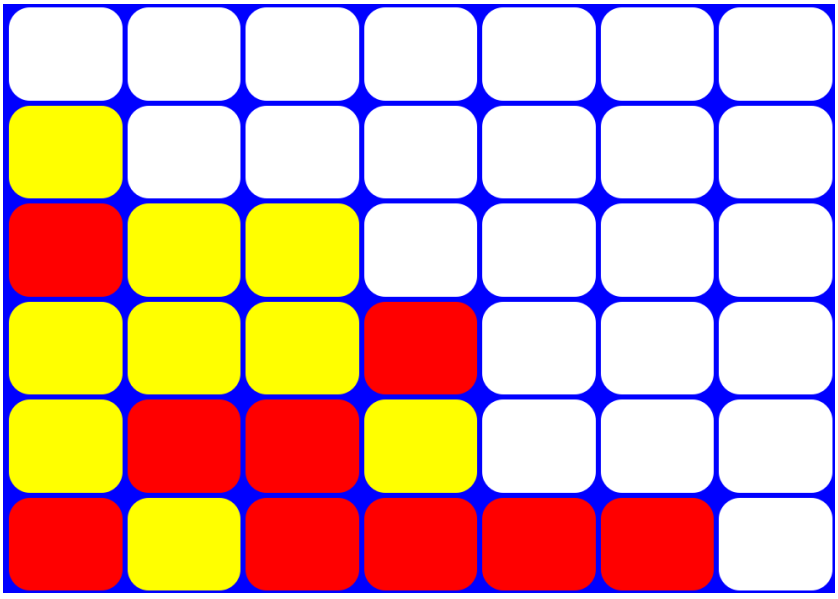
1.

Initial State:



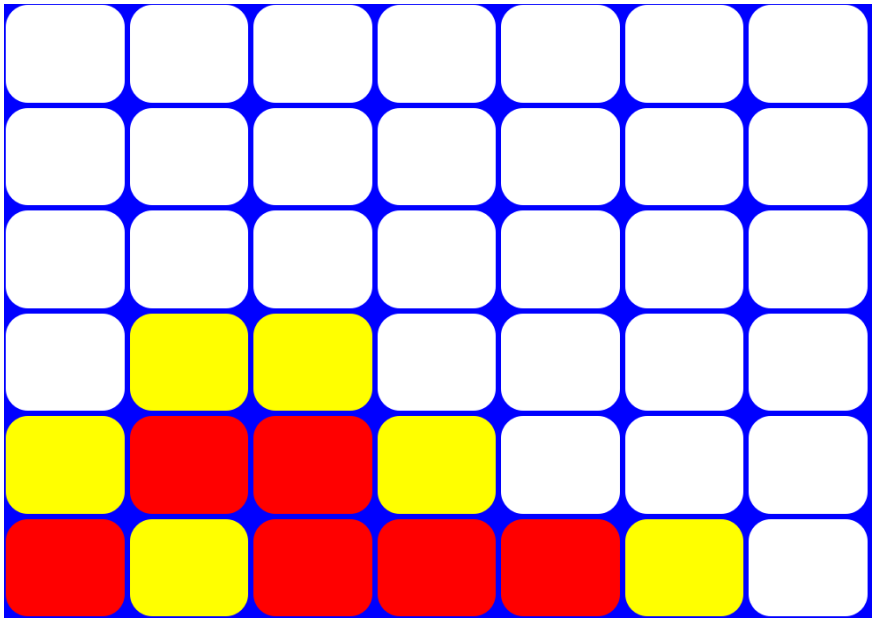
With depth 5 tree Insert into columns 6:

	Without pruning	With pruning
Time	6.2554 sec	0.3297 sec
Nodes expanded	16325	1874



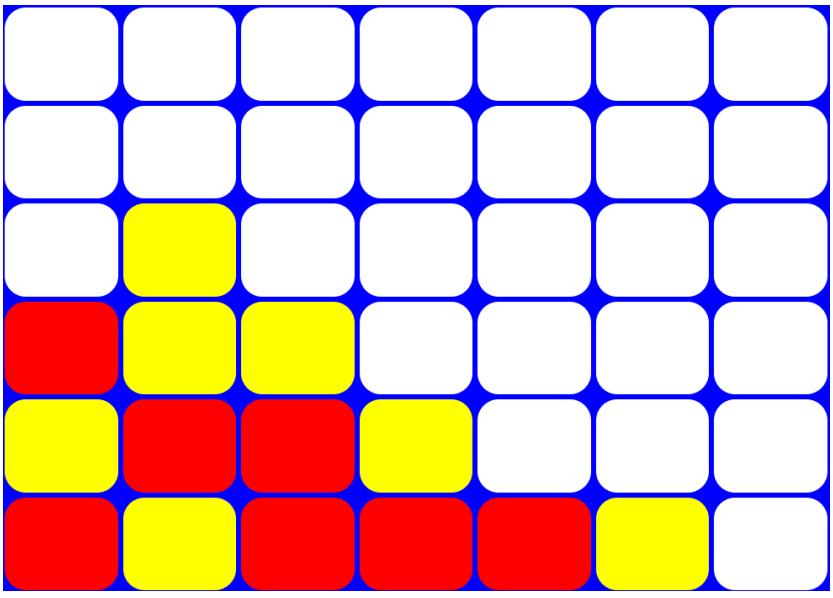
2.

Initial State:

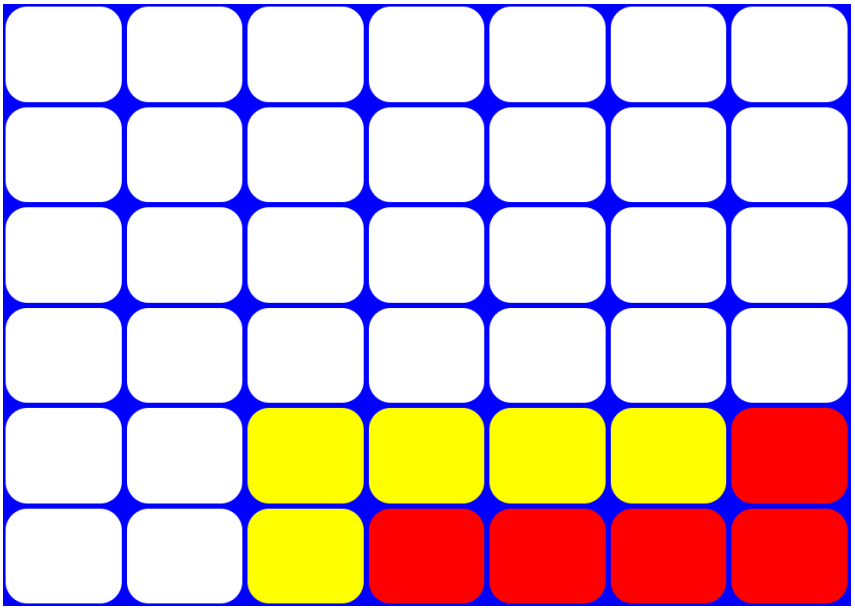


With depth 5 tree Insert into columns 1:

	Without pruning	With pruning
Time	8.6350 sec	0.7992 sec
Nodes expanded	19511	4025



3.



With depth 4 tree Insert into columns 7:

	Without pruning	With pruning
Time	1.2523 sec	0.2371 sec
Nodes expanded	2800	990