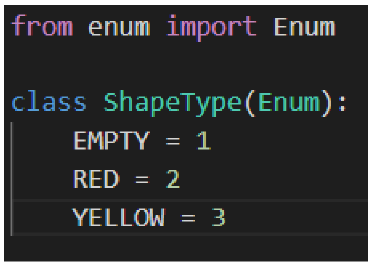
Islam Yousry Abdelwahid 14 Andrew Adel Sanad 17 Seif Eldeen Ehab Mostafa 32

REPORT

Algorithms

- Shape_type:

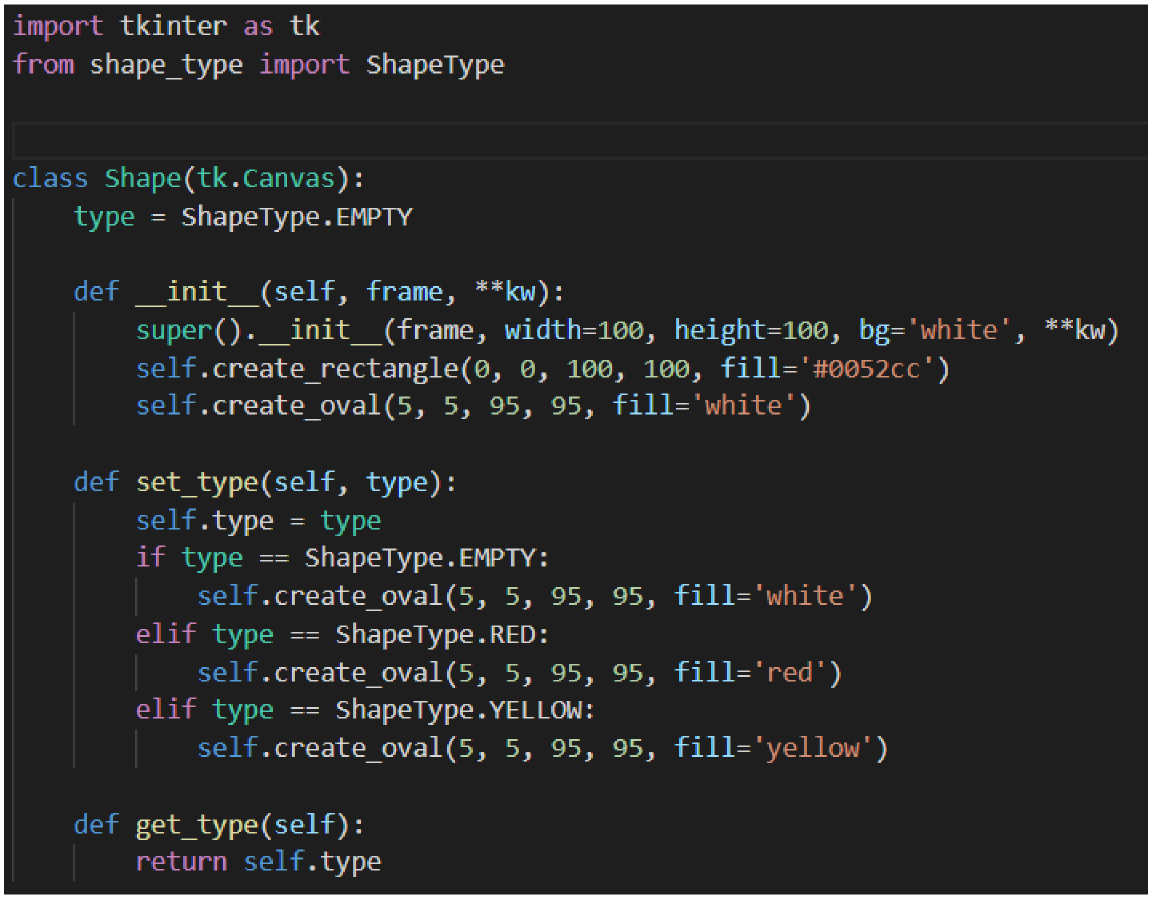
Classifies the shapes into Empty, Red or Yellow shapes.



```python
from enum import Enum

class ShapeType(Enum):
    EMPTY = 1
    RED = 2
    YELLOW = 3
```

- Shape:

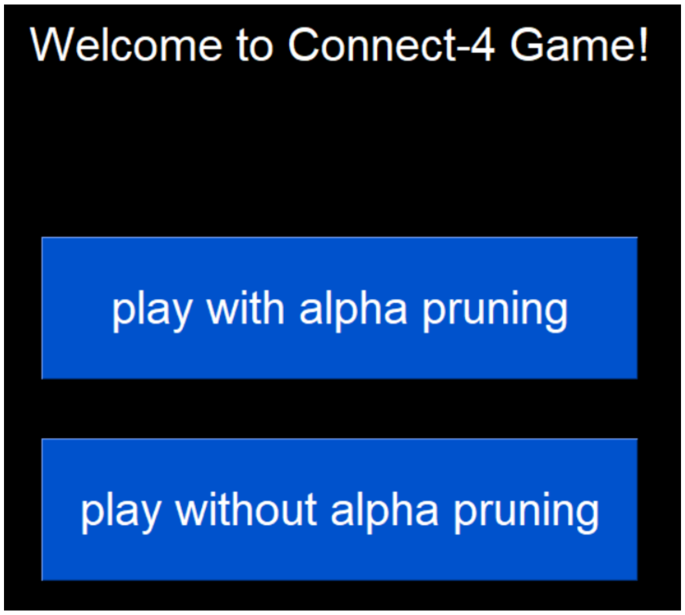
The shapes drawn in the GUI for the Connect 4 game from rectangles and circles with their specific colors and the possibility of the circle to be either Empty, Red or Yellow.

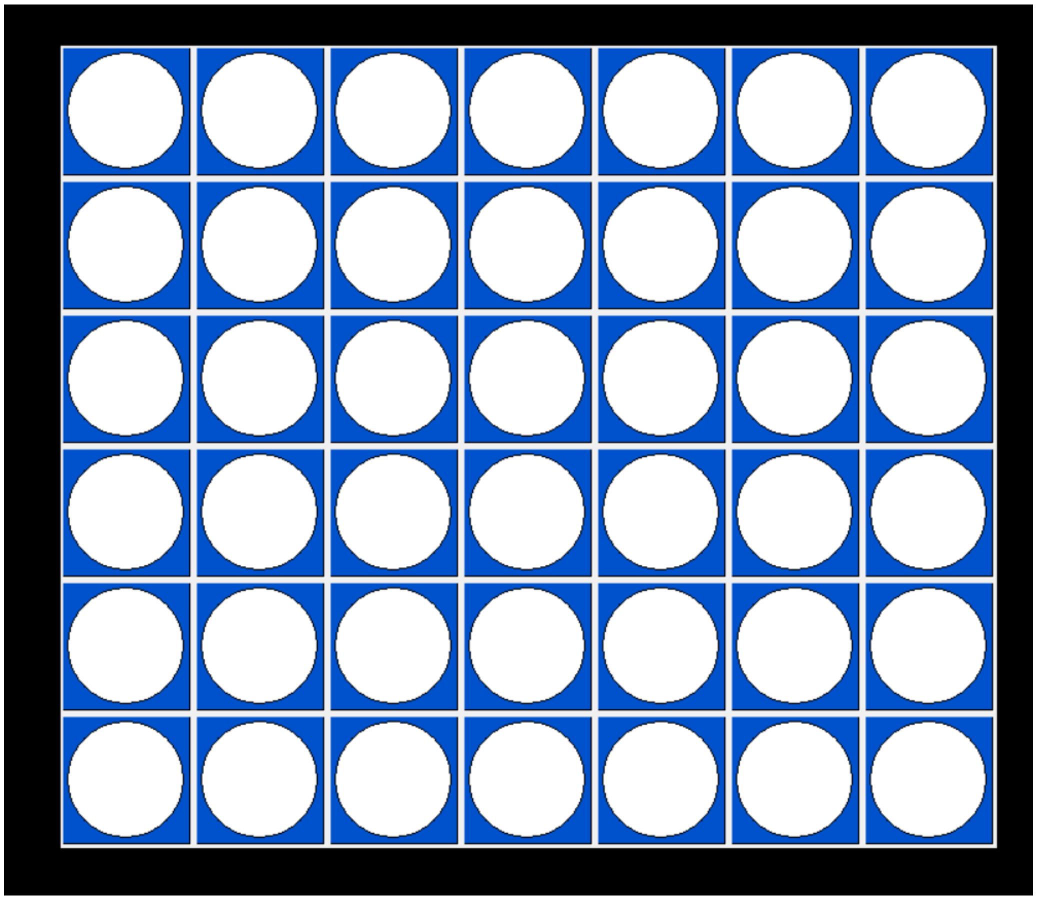


```python
import tkinter as tk
from shape_type import ShapeType


class Shape(tk.Canvas):
    type = ShapeType.EMPTY

    def __init__(self, frame, **kw):
        super().__init__(frame, width=100, height=100, bg='white', **kw)
        self.create_rectangle(0, 0, 100, 100, fill='#0052cc')
        self.create_oval(5, 5, 95, 95, fill='white')

    def set_type(self, type):
        self.type = type
        if type == ShapeType.EMPTY:
            self.create_oval(5, 5, 95, 95, fill='white')
        elif type == ShapeType.RED:
            self.create_oval(5, 5, 95, 95, fill='red')
        elif type == ShapeType.YELLOW:
            self.create_oval(5, 5, 95, 95, fill='yellow')

    def get_type(self):
        return self.type
```

- GUI

Suitable functions for the GUI from main menu to the GUI of the Connect 4 game with dynamic color changes by mouse click using the function insert_disc.





- Calculate_heuristics:

Used to calculate the heuristics function used in the minimax tree algorithm for choosing the best move for the computer.

Vertically and horizontally, it checks the last 3 changes happened in all 4 directions and determine the score as it adds 1 point if the new disc connects a 4 in a row or adds 0.1 point if there is a possibility that the next disc would score a 1 point for 3 connected in a row.

Diagonally, it traverses through the board using the zigzag method in both directions from left to right and right to left and calculates the score the same way as before.

- Build_minimax_tree

Recursive function for dealing with K which is the maximum depth for the search tree as it explores every path possible from having input in different columns except if a column is already full then calculating the heuristic score of every child then doing the same for the next children level.

- Minimaxing

Starting from score equal zero if we are trying to find the maximum or score equal infinite if we are trying to find the minimum it traverses through all the children recursively until it finds the score for every node.

- Finding the path

During the build of the minmax tree it stores the index of the chosen column for the current path alongside the score so we can trace back the patch after finding the best path where the next index is the index of a child of the root that has the same score of the root after applying the minmax algorithm.

- Min value

Initialize v as infinite and for each successor return its value which will be max in the next level or will return its score if it is a leaf level where v is the minimum of previous v and value of successors and beta is the min between beta and v.

- Max value

Initialize v as negative infinite and for each successor return its value which will be min in the next level or will return its score if it is a leaf level where v is the maximum of previous v and value of successors and alpha is the max between alpha and v.

- Alpha beta pruning

A node calculates its min value by traversing over its children recursively until it reaches the leaves level as it alternates each level between calculating min or max as if a value of other child is worse than current alpha, so we stop considering the other children left without expanding it.

Data Structures

- Tkinder is used for the frame and the GUI.
- The board is a 2D array where each row contains shapes that can have different colors.
- prev_rows_type/prev_cols_type is a 2D array of 7 colors 1 for each column input used for the previous 3 color changes related to the current cell as it has 3 elements last one is the color of last change before the current color and so on.
- prev_rows_count/prev_cols_count is a 2D array of 7 numbers 1 for each column input used for the count of previous 3 color changes related to the current cell for each change in a row as it has 3 elements last one is the color of last change before the current color and so on.

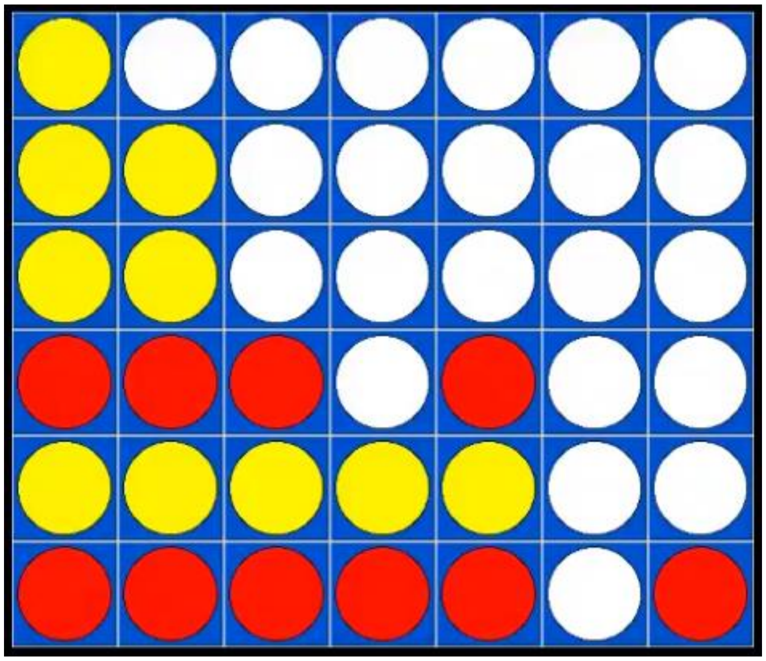
- The tree used is a structure of nodes using the AnyNode data structure as each node points to its parent and a list of its children alongside 2 attributes of the node's score and the chosen path's index.

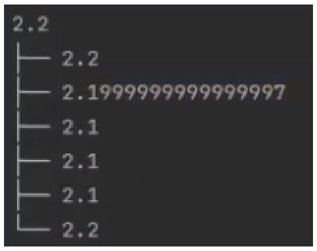
Assumptions

- Human always starts first with color red and computer always has color yellow.
- Score for 4 in a row is 1 point while score of 3 in a row with empty color is 0.1 point so 10 3 in a row is equivalent to 1 4 in a row.

Sample Runs

1.

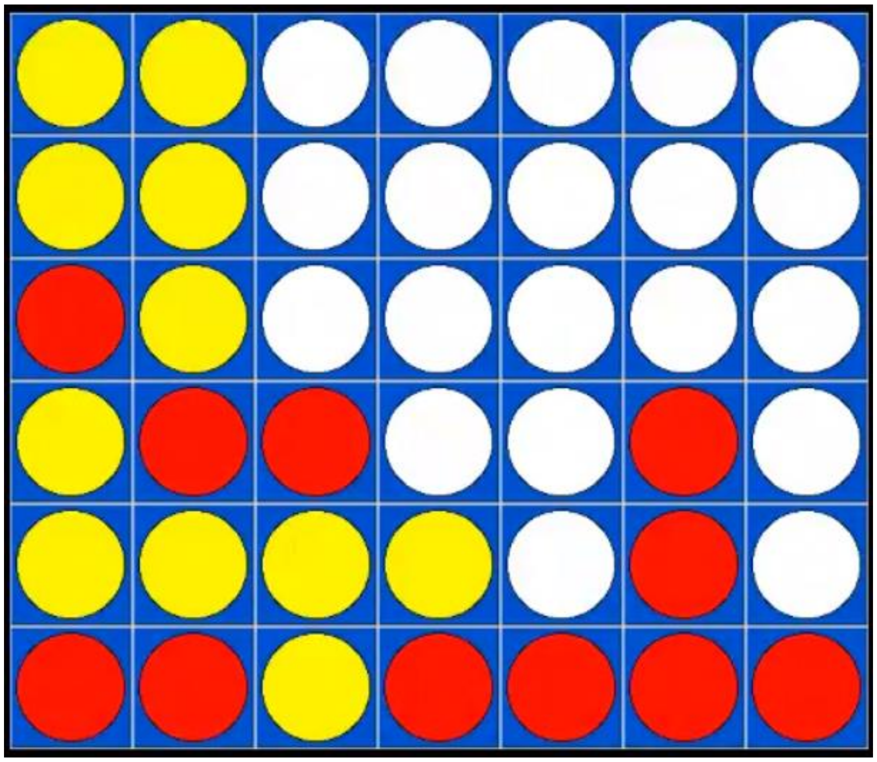


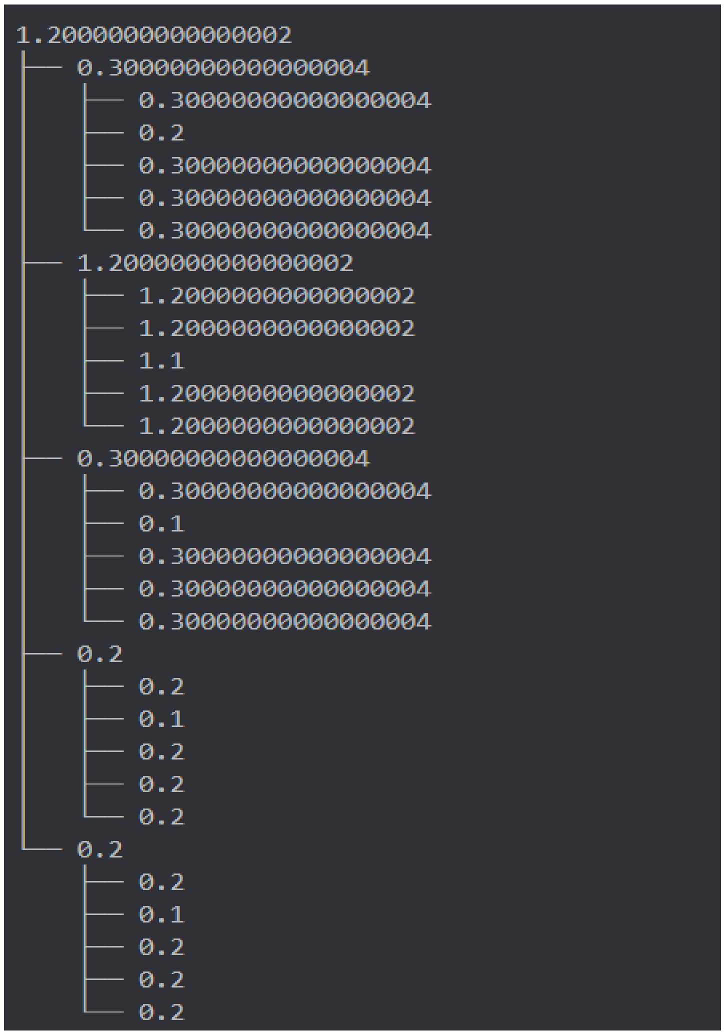


```
2.2
├── 2.2
├── 2.1999999999999997
├── 2.1
├── 2.1
├── 2.1
└── 2.2
```

Islam Yousry Abdelwahid 14 Andrew Adel Sanad 17 Seif Eldeen Ehab Mostafa 32

2.

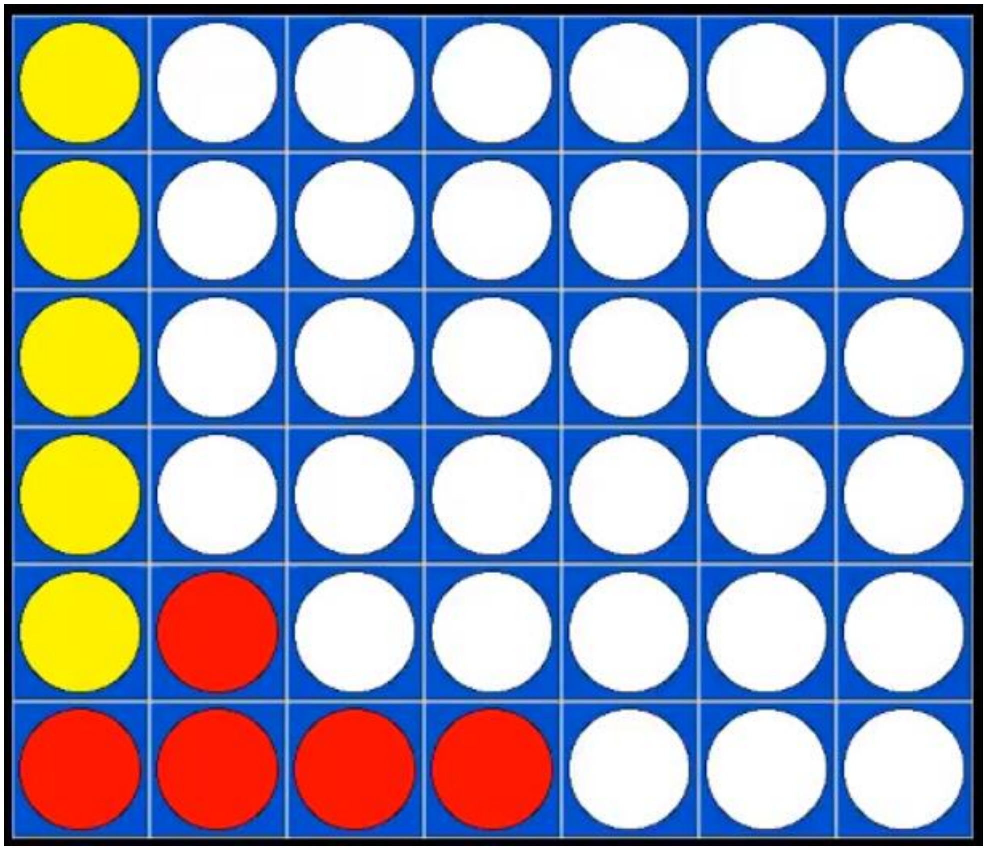


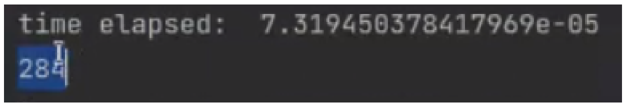


```
1.2000000000000002
├── 0.30000000000000004
│   ├── 0.30000000000000004
│   ├── 0.2
│   ├── 0.30000000000000004
│   ├── 0.30000000000000004
│   └── 0.30000000000000004
├── 1.2000000000000002
│   ├── 1.2000000000000002
│   ├── 1.2000000000000002
│   ├── 1.1
│   ├── 1.2000000000000002
│   └── 1.2000000000000002
├── 0.30000000000000004
│   ├── 0.30000000000000004
│   ├── 0.1
│   ├── 0.30000000000000004
│   ├── 0.30000000000000004
│   └── 0.30000000000000004
├── 0.2
│   ├── 0.2
│   ├── 0.1
│   ├── 0.2
│   ├── 0.2
│   └── 0.2
└── 0.2
    ├── 0.2
    ├── 0.1
    ├── 0.2
    ├── 0.2
    └── 0.2
```

Comparison

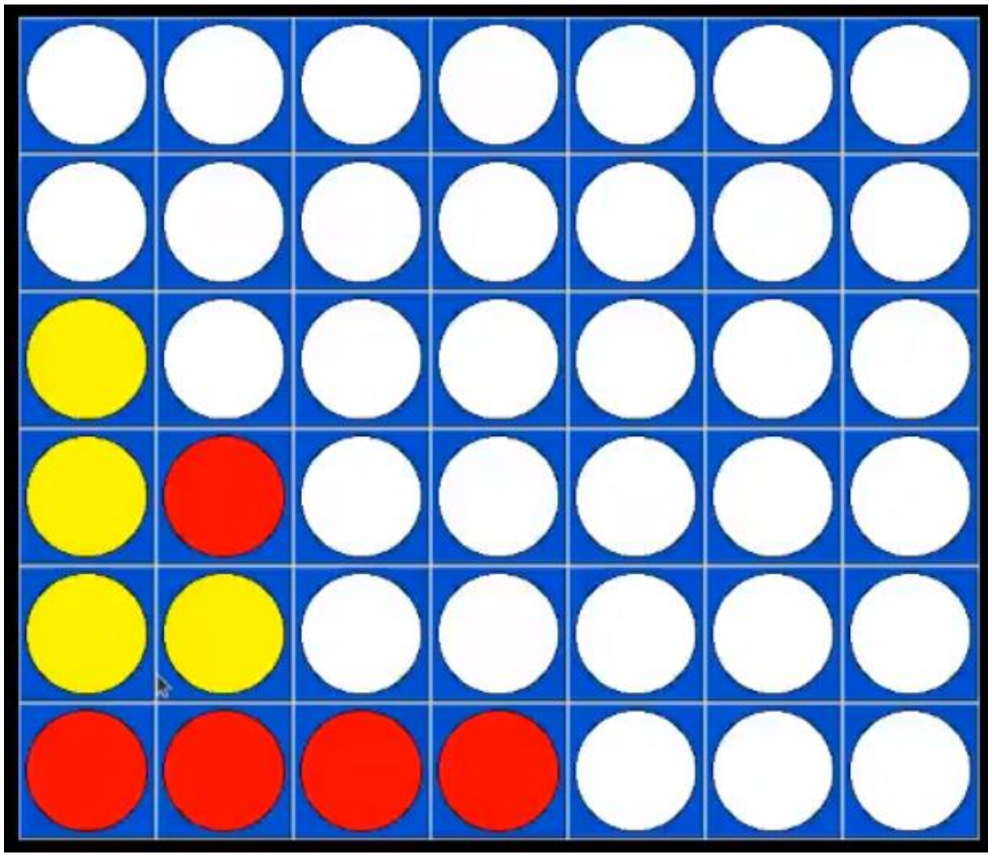
1. K = 2
    - Without prunning

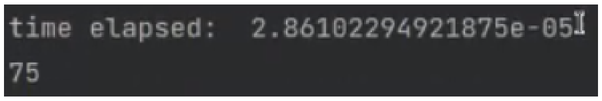




```
time elapsed:  7.319450378417969e-05
284
```

- With pruning

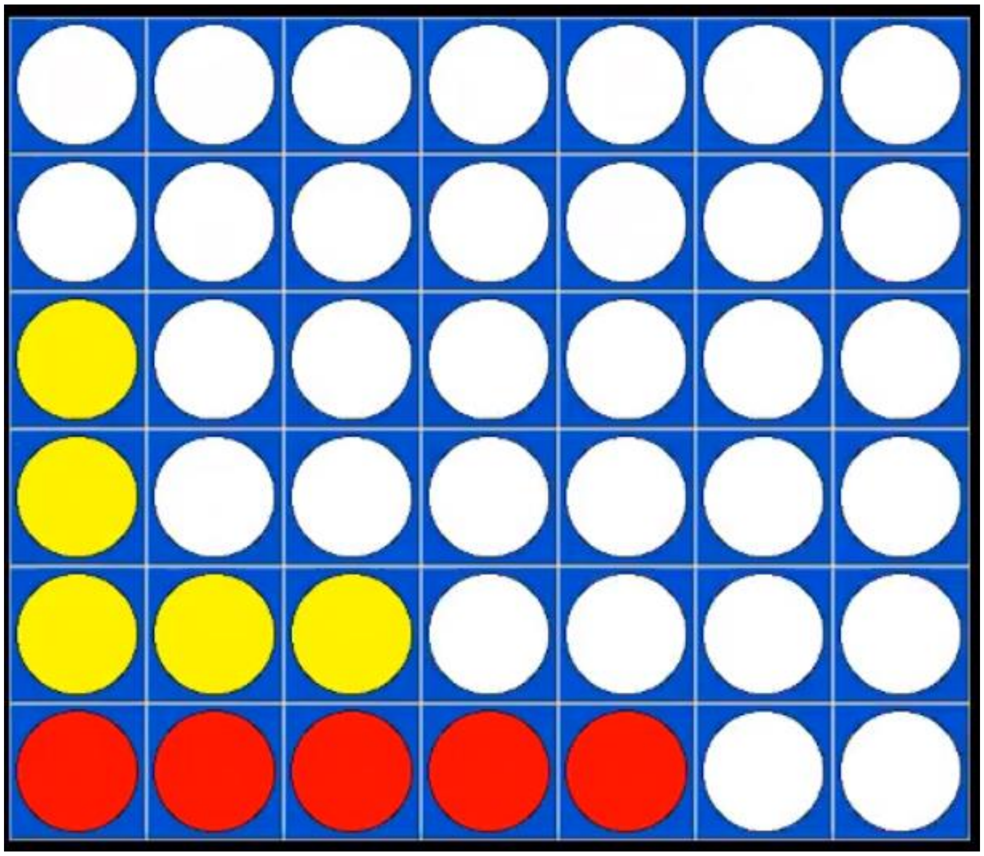


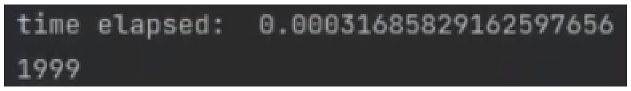


```
time elapsed:  2.86102294921875e-05
75
```

2. K = 3

- Without pruning





```
time elapsed:  0.0003168582916259765б
1999
```