МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ІНСТИТУТ ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ФУНДАМЕНТАЛЬНИХ НАУК

Кафедра прикладної математики

**ЗВІТ**

про виконання лабораторної роботи №6 із дисципліни:

“**Математичні основи штучного інтелекту”**

Виконав:

студент групи ПМ-31

Матолін Марко

Прийняв:

доцент кафедри ПМ

Пабирівський В.В.

Львів ‒ 2023

**Тема:** «Алгоритм гри на основі дерева рішень»

**Завдання**

Розробити програмну реалізацію гри «хрестики-нулики» на основі дерева рішень.

Етапи виконання завдання

1. Вивчити із використанням запропонованих літературних джерел зміст методу та застосування дерева прийняття рішень.
2. Розробити структуру даних для зберігання дерева рішень.
3. Розробити дерево рішень для гри «хрестики-нулики».
4. Реалізувати допоміжну функцію для вибору комп’ютером рішення з дерева рішень.
5. Реалізувати допоміжні функції для візуалізація поля гри «хрестики-

нулики» та взаємодії з користувачем через маніпулятор «миша» (може бути використано довільний інструмент візуальної розробки, наприклад, MS Visual Studio та Win Forms).

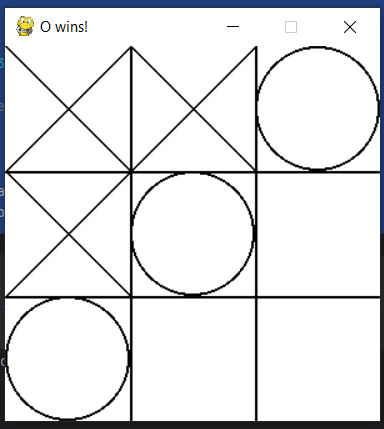
1. Безпосередньо реалізувати гру «хрестики-нулики» для гри людини з комп’ютером (забезпечити безпрограшну гру комп’ютера).

Варіант - 3

Код програми:

import pygame  
import sys  
  
# Розміри вікна гри  
WIDTH = 300  
HEIGHT = 300  
  
# Колір клітинок  
WHITE = (255, 255, 255)  
BLACK = (0, 0, 0)  
  
# Розміри клітинки  
CELL\_SIZE = WIDTH // 3  
  
# Ініціалізація Pygame  
pygame.init()  
  
# Створення вікна гри  
screen = pygame.display.set\_mode((WIDTH, HEIGHT))  
pygame.display.set\_caption("Tic-Tac-Toe")  
  
# Ініціалізація гри "хрестики-нулики"  
game\_board = [[None, None, None], [None, None, None], [None, None, None]]  
current\_player = "X"  
game\_over = False  
  
# Оновлення екрану  
def update\_screen():  
 screen.fill(WHITE)  
  
 # Малюємо лінії  
 pygame.draw.line(screen, BLACK, (CELL\_SIZE, 0), (CELL\_SIZE, HEIGHT), 2)  
 pygame.draw.line(screen, BLACK, (2 \* CELL\_SIZE, 0), (2 \* CELL\_SIZE, HEIGHT), 2)  
 pygame.draw.line(screen, BLACK, (0, CELL\_SIZE), (WIDTH, CELL\_SIZE), 2)  
 pygame.draw.line(screen, BLACK, (0, 2 \* CELL\_SIZE), (WIDTH, 2 \* CELL\_SIZE), 2)  
  
 # Малюємо хрестики та нулики  
 for row in range(3):  
 for col in range(3):  
 if game\_board[row][col] == "X":  
 pygame.draw.line(screen, BLACK, (col \* CELL\_SIZE, row \* CELL\_SIZE),  
 ((col + 1) \* CELL\_SIZE, (row + 1) \* CELL\_SIZE), 2)  
 pygame.draw.line(screen, BLACK, (col \* CELL\_SIZE, (row + 1) \* CELL\_SIZE),  
 ((col + 1) \* CELL\_SIZE, row \* CELL\_SIZE), 2)  
 elif game\_board[row][col] == "O":  
 pygame.draw.circle(screen, BLACK, ((col + 0.5) \* CELL\_SIZE, (row + 0.5) \* CELL\_SIZE),  
 CELL\_SIZE // 2, 2)  
  
 pygame.display.flip()  
  
# Перевірка переможця  
def check\_winner(board):  
 # Перевірка по горизонталі  
 for row in range(3):  
 if board[row][0] == board[row][1] == board[row][2] is not None:  
 return board[row][0]  
  
 # Перевірка по вертикалі  
 for col in range(3):  
 if board[0][col] == board[1][col] == board[2][col] is not None:  
 return board[0][col]  
  
 # Перевірка по діагоналі  
 if board[0][0] == board[1][1] == board[2][2] is not None:  
 return board[0][0]  
 if board[0][2] == board[1][1] == board[2][0] is not None:  
 return board[0][2]  
  
 # Перевірка нічиєї  
 if all(board[row][col] is not None for row in range(3) for col in range(3)):  
 return "DRAW"  
  
 # Гра продовжується  
 return None  
  
# Оновлення гри після ходу гравця  
def update\_game(row, col):  
 if game\_board[row][col] is None:  
 game\_board[row][col] = current\_player  
  
 winner = check\_winner(game\_board)  
 if winner is not None:  
 end\_game(winner)  
 else:  
 change\_turn()  
  
# Зміна ходу  
def change\_turn():  
 global current\_player  
 current\_player = "O" if current\_player == "X" else "X"  
  
# Закінчення гри  
def end\_game(winner):  
 global game\_over  
 game\_over = True  
  
 if winner == "DRAW":  
 message = "It's a draw!"  
 else:  
 message = f"{winner} wins!"  
  
 pygame.display.set\_caption(message)  
  
# Функція для вибору комп'ютером рішення з дерева рішень  
def choose\_computer\_move():  
 best\_score = -float('inf')  
 best\_move = None  
  
 for row in range(3):  
 for col in range(3):  
 if game\_board[row][col] is None:  
 game\_board[row][col] = "O"  
 score = minimax(game\_board, 0, False)  
 game\_board[row][col] = None  
  
 if score > best\_score:  
 best\_score = score  
 best\_move = (row, col)  
  
 return best\_move  
  
# Рекурсивний алгоритм Мінімакс  
def minimax(board, depth, is\_maximizing):  
 result = check\_winner(board)  
 if result is not None:  
 if result == "X":  
 return -1  
 elif result == "O":  
 return 1  
 else:  
 return 0  
  
 if is\_maximizing:  
 best\_score = -float('inf')  
 for row in range(3):  
 for col in range(3):  
 if board[row][col] is None:  
 board[row][col] = "O"  
 score = minimax(board, depth + 1, False)  
 board[row][col] = None  
 best\_score = max(score, best\_score)  
 return best\_score  
 else:  
 best\_score = float('inf')  
 for row in range(3):  
 for col in range(3):  
 if board[row][col] is None:  
 board[row][col] = "X"  
 score = minimax(board, depth + 1, True)  
 board[row][col] = None  
 best\_score = min(score, best\_score)  
 return best\_score  
  
# Головний цикл гри  
def run\_game():  
 while True:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 elif event.type == pygame.MOUSEBUTTONDOWN and not game\_over:  
 mouse\_pos = pygame.mouse.get\_pos()  
 col = mouse\_pos[0] // CELL\_SIZE  
 row = mouse\_pos[1] // CELL\_SIZE  
 update\_game(row, col)  
 if not game\_over:  
 computer\_move = choose\_computer\_move()  
 if computer\_move:  
 update\_game(computer\_move[0], computer\_move[1])  
  
 update\_screen()  
  
run\_game()

**Результат виконання:**



**Висновок:**

На лабораторній роботі я ознайомилась з реалізаціє алгоритму гри на основі дерева рішень, та реалізувала його на практиці на прикладі гри «Хрестики-нулики»