Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 6 з дисципліни
«Проектування алгоритмів»

„Пошук в умовах протидії, ігри з повною інформацією, ігри з елементом випадковості, ігри з неповною інформацією”

Виконав(ла) *ІП-22 Іщенко Кіра Віталіївна* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Ахаладзе І.Е.* _____
(прізвище, ім'я, по батькові)

Київ 2024

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи - вивчити основні підходи до формалізації алгоритмів знаходження рішень задач в умовах протидії. Ознайомитися з підходами до програмування алгоритмів штучного інтелекту в іграх з повною інформацією, іграх з елементами випадковості та в іграх з неповною інформацією.

2 ЗАВДАННЯ

Для ігор з повної інформацією, згідно варіанту (таблиця 2.1) реалізувати візуальний ігровий додаток для гри користувача з комп'ютерним опонентом. Для реалізації стратегії гри комп'ютерного опонента використовувати алгоритм альфа-бета-відсікань. Реалізувати три рівні складності (легкий, середній, складний).

Для ігор з елементами випадковості, згідно варіанту (таблиця 2.1) реалізувати візуальний ігровий додаток, з користувацьким інтерфейсом, не консольним, для гри користувача з комп'ютерним опонентом. Для реалізації стратегії гри комп'ютерного опонента використовувати алгоритм мінімакс.

Для карткових ігор, згідно варіанту (таблиця 2.1), реалізувати візуальний ігровий додаток, з користувацьким інтерфейсом, не консольним, для гри користувача з комп'ютерним опонентом. Потрібно реалізувати стратегію комп'ютерного опонента, і звести гру до гри з повною інформацією (див. Лекцію), далі реалізувати стратегію гри комп'ютерного опонента за допомогою алгоритму мінімаксу або альфа-бета-відсікань.

Реалізувати анімацію процесу жеребкування (+1 бал) або реалізувати анімацію ігрових процесів (роздачі карт, анімацію ходів тощо) (+1 бал).

Реалізувати варто тільки одне з бонусних завдань.

Зробити узагальнений висновок лабораторної роботи.

Таблиця 2.1 – Варіанти

№	**Варіант**	**Тип гри**
1	Яцзи https://game-wiki.guru/published/igryi/yaczzyi.html	З елементами випадковості
2	Лудо http://www.iggamecenter.com/info/ru/ludo.html	З елементами випадковості
3	Генерал http://www.rules.net.ru/kost.php?id=7	З елементами випадковості

4	Нейтріко http://www.iggamecenter.com/info/ru/neutreeko.html	З повною інформацією
5	Тринадцять http://www.rules.net.ru/kost.php?id=16	З елементами випадковості
6	Індійські кості http://www.rules.net.ru/kost.php?id=9	З елементами випадковості
7	Dots and Boxes https://ru.wikipedia.org/wiki/Палочки_(игра)	З повною інформацією
8	Двадцять одне http://gamerules.ru/igry-v-kosti-part8#dvadtsat-odno	З елементами випадковості
9	Тіко http://www.iggamecenter.com/info/ru/teeko.html	З повною інформацією
10	Клоббер http://www.iggamecenter.com/info/ru/clobber.html	З повною інформацією
11	101 https://www.durbetsel.ru/2_101.htm	Карткові ігри
12	Hackenbush http://www.papg.com/show?1TMP	З повною інформацією
13	Табу https://www.durbetsel.ru/2_taboo.htm	Карткові ігри
14	Заєць і Вовки (за Зайця) http://www.iggamecenter.com/info/ru/foxh.html	З повною інформацією
15	Свої козирі https://www.durbetsel.ru/2_svoi-koziri.htm	Карткові ігри
16	Війна з ботами https://www.durbetsel.ru/2_voina_s_botami.htm	Карткові ігри
17	Domineering 8x8 http://www.papg.com/show?1TX6	З повною інформацією
18	Останній гравець https://www.durbetsel.ru/2_posledny_igrok.htm	Карткові ігри
19	Заєць и Вовки (за Вовків) http://www.iggamecenter.com/info/ru/foxh.html	З повною інформацією

20	Богач https://www.durbetsel.ru/2_bogach.htm	Карткові ігри
21	Редуду https://www.durbetsel.ru/2_redudu.htm	Карткові ігри
22	Эльферн https://www.durbetsel.ru/2_elfern.htm	Карткові ігри
23	Ремінь https://www.durbetsel.ru/2_remen.htm	Карткові ігри
24	Реверсі https://ru.wikipedia.org/wiki/Реверси	З повною інформацією
25	Вари http://www.iggamecenter.com/info/ru/oware.html	З повною інформацією
26	Яцзи https://game-wiki.guru/published/igryi/yaczzyi.html	З елементами випадковості
27	Лудо http://www.iggamecenter.com/info/ru/ludo.html	З елементами випадковості
28	Генерал http://www.rules.net.ru/kost.php?id=7	З елементами випадковості
29	Сим https://ru.wikipedia.org/wiki/Сим_(игра)	З повною інформацією
30	Col http://www.papg.com/show?2XLY	З повною інформацією
31	Snort http://www.papg.com/show?2XM1	З повною інформацією
32	Chomp http://www.papg.com/show?3AEA	З повною інформацією
33	Gale http://www.papg.com/show?1TPI	З повною інформацією
34	3D Noughts and Crosses 4 x 4 x 4 http://www.papg.com/show?1TND	З повною інформацією
35	Snakes http://www.papg.com/show?3AE4	З повною інформацією

3 ВИКОНАННЯ

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

```
#include <utility>
#include <climits>
#include <algorithm>
#include <random>
class miniMax
{
        int board[25];
        int currentPlayer = 1;
        int figureInUse = 0;
        int pos[4] = { -1, -1, -1, -1 };

        int miniMaxWork(int depth, bool isMaximizingPlayer);
public:
        std::pair<int, int> move();
        miniMax();
        bool checkWin();
        bool checkOver();
        int computerMove(int);
};
miniMax::miniMax()
{
   for (int i = 0; i < 25; i++) {
      board[i] = 0;
   }
};

bool miniMax::checkWin()
{
   for (int col = 0; col < 5; col++) {
      for (int row = 0; row < 2; row++) {
         int count = 0;
         for (int i = 0; i < 4; i++) {
```

```
            if (board[col + row * 5 + i] == currentPlayer) {
                count++;
            }
        }
        if (count == 4) {
            return true;
        }
    }
}

for (int row = 0; row < 5; row++) {
    for (int col = 0; col < 2; col++) {
        int count = 0;
        for (int i = 0; i < 4; i++) {
            if (board[col + row * 5 + i * 5] == currentPlayer) {
                count++;
            }
        }
        if (count == 4) {
            return true;
        }
    }
}

for (int diag = 0; diag < 2; diag++) {
    for (int start = 0; start < 21; start += 5) {
        int count = 0;
        for (int i = 0; i < 4; i++) {
            if (board[start + diag * 4 + i * 6] == currentPlayer) {
                count++;
            }
        }
        if (count == 4) {
            return true;
        }
    }
}
for (int row = 0; row < 4; row++) {
```

```
        for (int col = 0; col < 4; col++) {
            int count = 0;
            for (int i = 0; i < 2; i++) {
                for (int j = 0; j < 2; j++) {
                    if (board[col + row * 5 + i + j * 5] == currentPlayer) {
                        count++;
                    }
                }
            }
            if (count == 4) {
                return true;
            }
        }
    }
    return false;
}
void miniMax::updateBoard(int old, int newPos)
{
    int i = 0;
    if (old-1 > -1) {
        board[old-1] = 0;
    }
    if (newPos - 1 > -1) {
        board[newPos - 1] = currentPlayer;
    }
}
std::pair<int, int> miniMax::move() {
    if (figureInUse < 4) {
        figureInUse++;
        return { computerMove(figureInUse), figureInUse};
    }
    else {
        int selectedFigure = std::rand() % 4;
        board[pos[selectedFigure-1]] = 0;
        return {computerMove(selectedFigure), selectedFigure };
    }

}
```

```
int miniMax::computerMove(int position)
{
   int bestMove = -1;
   int bestScore = INT_MIN;;
   int depth = 3;
   for (int i = 0; i < 25; i++) {
      if (board[i] == 0) {
         board[i] = 2;
         int moveScore = miniMaxWork(depth - 1, true);
         board[i] = 0;

         if (moveScore > bestScore) {
            bestScore = moveScore;
            bestMove = i;
         }
      }
   }
   pos[position-1] = bestMove;
   board[bestMove] = 2;
   return bestMove;
}

int miniMax::miniMaxWork (int depth, bool isMaximizingPlayer) {
   int score = checkWin();
   if (score != 0 || depth == 0) {
      return score;
   }
   if (isMaximizingPlayer) {
      int bestScore = INT_MIN;

      for (int i = 0; i < 25; i++) {
         if (board[i] == 0) {
            board[i] = 2;
            int childScore = miniMaxWork(depth - 1, false);
            bestScore = std::max(bestScore, childScore);

            board[i] = 0;
         }
```

```
        }
        return bestScore;
    }
    else {
        int bestScore = INT_MAX;
        for (int i = 0; i < 25; i++) {
            if (board[i] == 0) {
                board[i] = 2;
                int childScore = miniMaxWork(depth - 1, true);
                bestScore = std::min(bestScore, childScore);
                board[i] = 0;
            }
        }
        return bestScore;
    }
}
bool miniMax::checkOver()
{
    int cur = currentPlayer;
    if (checkWin()) {
        return true;
    }
    else {
        if (cur == 1) {
            currentPlayer = 2;
        }
        else {
            currentPlayer = 1;
        }
        if (checkWin()) {
            return true;
        }
        else {
            currentPlayer = cur;
            return false;
        }
    }
}
```

3.1.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

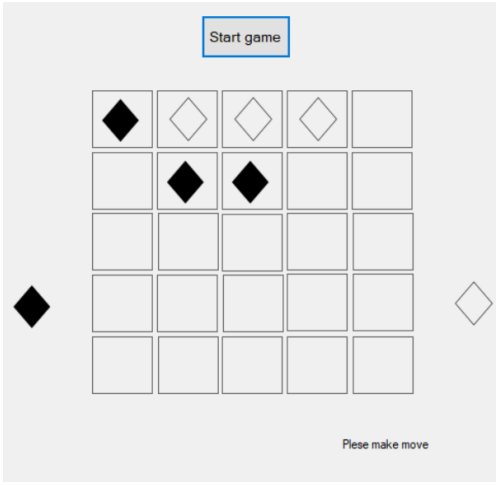

Рисунок 3.1 – Робота програми в процесі, користувач ходить чорними, комп'ютер білими

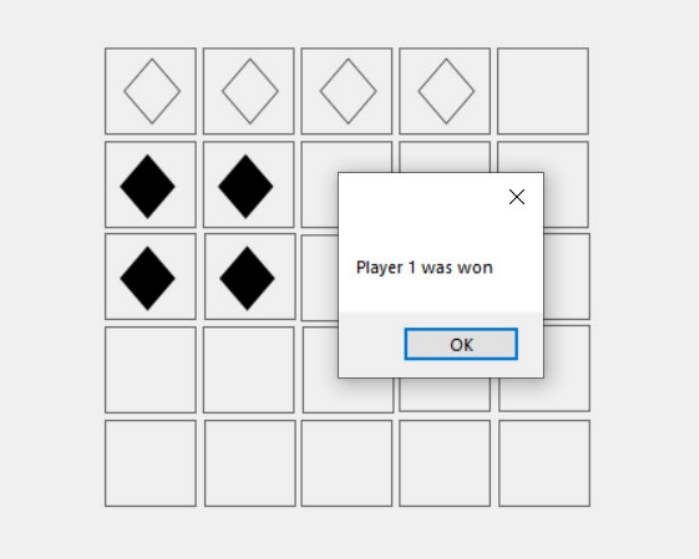

Рисунок 3.2 – Користувач переміг комп'ютер

ВИСНОВОК

В рамках даної лабораторної роботи було розроблено алгоритм альфа-бета-відсікань для гри Тіко. Розроблена програма дозволяє грати з комп’ютером у цю гру. Результати роботи програми залежать від глибини рекурсії алгоритму, важливо знайти баланс між швидкістю роботи та оптимальністю ходів алгоритму.

КРИТЕРІЇ ОЦІНЮВАННЯ

При здачі лабораторної роботи до 31.12.2023 включно максимальний бал дорівнює – 5. Після 31.12.2023 максимальний бал дорівнює – 4,5.

Критерії оцінювання у відсотках від максимального балу:

- програмна реалізація – 75%;
- робота з гіт – 20%;
- висновок – 5%.

+1 додатковий бал можна отримати за реалізацію анімації ігрових процесів (жеребкування, роздачі карт, анімацію ходів тощо).

+1 додатковий бал можна отримати за виконання та захист роботи до 24.12.2023.