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: Григоренко Родіон Ярославович

Перевірив *Сопов О. О.*

(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

Варіант 5

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.

• 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	Индійські 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

ВИКОНАННЯ

Псевдокод алгоритмів

- Алгоритм MiniMax

public int AI(int diceNum, int iterations, int sum, int score)

ПОЧАТОК

```
int a = Minimax(diceNum, iterations + 1, sum + diceNum, sum % 13 == 0 ? sum / 13 : score);
    int b = Minimax(diceNum, iterations + 1, sum - diceNum, sum % 13 == 0 ? sum / 13 : score);
    int c = Minimax(diceNum, iterations + 1, sum + 2 * diceNum, sum % 13 == 0 ? sum / 13 : score);
    int d = Minimax(diceNum, iterations + 1, sum + 3 * diceNum, sum % 13 == 0 ? sum / 13 : score);
    int e = Minimax(diceNum, iterations + 1, sum + 4 * diceNum, sum % 13 == 0 ? sum / 13 : score);
    int f = Minimax(diceNum, iterations + 1, sum + diceNum / 4, sum % 13 == 0 ? sum / 13 : score);
    int g = Minimax(diceNum, iterations + 1, sum + diceNum / 3, sum % 13 == 0 ? sum / 13 : score);
    int h = Minimax(diceNum, iterations + 1, sum + diceNum / 2, sum % 13 == 0 ? sum / 13 : score);
int max = Max(a, b, c, d, e, f, g, h);
```

ЯКЩО (a == max)

return 1;

ІНАКШЕ ЯКЩО (b == max)

return 2;

ІНАКШЕ ЯКЩО (c == max)

return 3;

ІНАКШЕ ЯКЩО (d == max)

return 4;

ІНАКШЕ ЯКЩО (e == max)

return 5;

ІНАКШЕ ЯКЩО (f == max)

return 6;

ІНАКШЕ ЯКЩО (g == max)

return 7;

ІНАКШЕ ЯКЩО (h == max)

return 8;

return 0;

КІНЕЦЬ

public int Minimax(int diceNum, int iterations, int sum, int score)

ПОЧАТОК

ЯКЩО (iterations >= 5)

return sum % 13 == 0 ? sum / 13 : score;

int max = -1;

ПОКИ (int i = 3; i < 14; i++)

int a = Minimax(i, iterations + 1, sum + diceNum, sum % 13 == 0 ? sum / 13 : score);

```
        int b = Minimax(i, iterations + 1, sum - diceNum, sum % 13 == 0 ? sum / 13 : score);
        int c = Minimax(i, iterations + 1, sum + 2*diceNum, sum % 13 == 0 ? sum / 13 : score);
        int d = Minimax(i, iterations + 1, sum + 3*diceNum, sum % 13 == 0 ? sum / 13 : score);
        int e = Minimax(i, iterations + 1, sum + 4*diceNum, sum % 13 == 0 ? sum / 13 : score);
        int f = Minimax(i, iterations + 1, sum + diceNum / 4, sum % 13 == 0 ? sum / 13 : score);
        int g = Minimax(i, iterations + 1, sum + diceNum/3, sum % 13 == 0 ? sum / 13 : score);
        int h = Minimax(i, iterations + 1, sum + diceNum/2, sum % 13 == 0 ? sum / 13 : score);
        int m = Max(a, b, c, d, e, f, g, h);
        ЯКЩО (m > max)
            max = m;

    return max;

КІНЕЦЬ
```

Вихідний код

Thirteen.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
using TMPro;

public class Thirteen : MonoBehaviour
{

    [SerializeField] public Sprite[] sprites;
    [SerializeField] public Transform dice1;
    [SerializeField] public Transform dice2;
    [SerializeField] TextMeshProUGUI playerSumText;
    [SerializeField] TextMeshProUGUI AISumText;
    [SerializeField] TextMeshProUGUI playerScoreText;
    [SerializeField] TextMeshProUGUI AIScoreText;
    public int playerSum;
    public int AISum;
    public int playerScore;
    public int AIScore;
    public int dice1num;
    public int dice2num;
    public int playerIterations;
    public int AIIterations;
    public bool canMakeMovePlayer;
    public bool canMakeMoveAI;
    // Start is called before the first frame update
    void Start()
    {
        AIIterations = 0;
```

```
        playerIterations = 0;
        playerScore = 0;
        AIScore = 0;
        playerSum = 0;
        AISum = 0;
        canMakeMovePlayer = true;
        canMakeMoveAI = false;
        UpdateScore(true);
        UpdateScore(false);
        UpdateSum(true);
        UpdateSum(false);
    }

    // Update is called once per frame
    void Update()
    {

    }



    public int AI(int diceNum, int iterations, int sum, int score)
    {
        int a = Minimax(diceNum, iterations + 1, sum + diceNum, sum
% 13 == 0 ? sum / 13 : score);
        int b = Minimax(diceNum, iterations + 1, sum - diceNum, sum
% 13 == 0 ? sum / 13 : score);
```

```
int c = Minimax(diceNum, iterations + 1, sum + 2 * diceNum, sum % 13 == 0 ? sum / 13 : score);

int d = Minimax(diceNum, iterations + 1, sum + 3 * diceNum, sum % 13 == 0 ? sum / 13 : score);

int e = Minimax(diceNum, iterations + 1, sum + 4 * diceNum, sum % 13 == 0 ? sum / 13 : score);

int f = Minimax(diceNum, iterations + 1, sum + diceNum / 4, sum % 13 == 0 ? sum / 13 : score);

int g = Minimax(diceNum, iterations + 1, sum + diceNum / 3, sum % 13 == 0 ? sum / 13 : score);

int h = Minimax(diceNum, iterations + 1, sum + diceNum / 2, sum % 13 == 0 ? sum / 13 : score);


int max = Max(a, b, c, d, e, f, g, h);

if(a == max)

{

   return 1;

}

else if(b == max)

{

   return 2;

}

else if (c == max)

{

   return 3;

}

else if (d == max)

{
```

```
        return 4;
    }
    else if (e == max)
    {
        return 5;
    }
    else if (f == max)
    {
        return 6;
    }
    else if (g == max)
    {
        return 7;
    }
    else if (h == max)
    {
        return 8;
    }
    return 0;
}

public int Minimax(int diceNum, int iterations, int sum, int score)
{
    if(iterations >= 5)
    {
        return sum % 13 == 0 ? sum / 13 : score;
```

```
    }

    int max = -1;

    for (int i = 3; i < 14; i++)
    {
        int a = Minimax(i, iterations + 1, sum + diceNum, sum % 13
== 0 ? sum / 13 : score);
        int b = Minimax(i, iterations + 1, sum - diceNum, sum % 13
== 0 ? sum / 13 : score);
        int c = Minimax(i, iterations + 1, sum + 2*diceNum, sum %
13 == 0 ? sum / 13 : score);
        int d = Minimax(i, iterations + 1, sum + 3*diceNum, sum %
13 == 0 ? sum / 13 : score);
        int e = Minimax(i, iterations + 1, sum + 4*diceNum, sum %
13 == 0 ? sum / 13 : score);
        int f = Minimax(i, iterations + 1, sum + diceNum / 4, sum %
13 == 0 ? sum / 13 : score);
        int g = Minimax(i, iterations + 1, sum + diceNum/3, sum % 13
== 0 ? sum / 13 : score);
        int h = Minimax(i, iterations + 1, sum + diceNum/2, sum % 13
== 0 ? sum / 13 : score);
        int m = Max(a, b, c, d, e, f, g, h);
        if(m > max)
        {
            max = m;
        }
    }
```

```
    return max;

}

public int Max(int a, int b, int c, int d, int e, int f, int g, int h)
{
    List<int> list = new List<int>();
    list.Add(a);
    list.Add(b);
    list.Add(c);
    list.Add(d);
    list.Add(e);
    list.Add(f);
    list.Add(g);
    list.Add(h);

    int max = -1;

    for (int i = 0; i < list.Count; i++)
    {
        if(list[i] > max)
        {
            max = list[i];
        }
    }
```

```
        return max;
    }

    public void RollTheDice(bool isPlayer)
    {
        if (isPlayer && canMakeMovePlayer || !isPlayer &&
canMakeMoveAI)
        {
            System.Random rnd = new System.Random();
            dice1num = rnd.Next(1, 7);
            dice2num = rnd.Next(2, 8);

            dice1.GetComponent<SpriteRenderer>().sprite =
sprites[dice1num - 1];
            dice2.GetComponent<SpriteRenderer>().sprite =
sprites[dice2num - 1];



            if (isPlayer)
            {
                playerIterations++;
                canMakeMovePlayer = false;
            }
            else
            {
                AIIterations++;
```

```
            canMakeMoveAI = false;
        }
    }

}

public void AITurn()
{
    while(AIIterations < 5)
    {
        Debug.Log(AIIterations);
        RollTheDice(false);
        int command = AI(dice1num + dice2num, AIIterations,
AISum, AIScore);
        if(command == 1)
        {
            AddSum(false);
        }
        else if(command == 2)
        {
            SubtractSum(false);
        }
        else if (command == 3)
        {
            MultiplyByTwo(false);
        }
```

```
        else if (command == 4)
        {
            MultiplyByThree(false);
        }
        else if (command == 5)
        {
            MultiplyByFour(false);
        }
        else if (command == 6)
        {
            TakeQuarter(false);
        }
        else if (command == 7)
        {
            TakeThird(false);
        }
        else if (command == 8)
        {
            TakeHalf(false);
        }
    }
}

public void UpdateSum(bool isPlayer)
{
    if (isPlayer)
```

```
        {
            playerSumText.text = "Player sum is " +
playerSum.ToString();
        }
        else if(!isPlayer)
        {
            AISumText.text = "AI sum is " + AISum.ToString();
        }
    }

    public void UpdateScore(bool isPlayer)
    {
        if (isPlayer)
        {
            playerScoreText.text = "Player score is " +
playerScore.ToString();
        }
        else if(!isPlayer && !canMakeMoveAI)
        {
            AIScoreText.text = "AI score is " + AIScore.ToString();
        }
    }

    public void AddSum(bool isPlayer)
    {
        if (isPlayer && !canMakeMovePlayer && playerIterations < 5)
        {
```

```
        playerSum += dice1num + dice2num;
        canMakeMovePlayer = true;
    }
    else if(!isPlayer && !canMakeMoveAI && AIIterations < 5)
    {
        AISum += dice1num + dice2num;
        canMakeMoveAI = true;
    }

    ManageScore(isPlayer);

}

public void SubtractSum(bool isPlayer)
{
    if (isPlayer && !canMakeMovePlayer && playerIterations < 5)
    {
        playerSum -= dice1num + dice2num;
        canMakeMovePlayer = true;
    }
    else if(!isPlayer && !canMakeMoveAI && AIIterations < 5)
    {
        AISum -= dice1num + dice2num;
        canMakeMoveAI = true;
    }
    ManageScore(isPlayer);
```

```
}

public void MultiplyByTwo(bool isPlayer)
{
    if (isPlayer && !canMakeMovePlayer && playerIterations < 5)
    {
        playerSum += (dice1num + dice2num) * 2;
        canMakeMovePlayer = true;
    }
    else if(!isPlayer && !canMakeMoveAI && AIIterations < 5)
    {
        AISum += (dice1num + dice2num) * 2;
        canMakeMoveAI = true;
    }
    ManageScore(isPlayer);
}

public void MultiplyByThree(bool isPlayer)
{
    if (isPlayer && !canMakeMovePlayer && playerIterations < 5)
    {
        playerSum += (dice1num + dice2num) * 3;
        canMakeMovePlayer = true;
    }
    else if(!isPlayer && !canMakeMoveAI && AIIterations < 5)
    {
```

```
            AISum += (dice1num + dice2num) * 3;
            canMakeMoveAI = true;
        }
        ManageScore(isPlayer);
    }

    public void MultiplyByFour(bool isPlayer)
    {
        if (isPlayer && !canMakeMovePlayer && playerIterations < 5)
        {
            playerSum += (dice1num + dice2num) * 4;
            canMakeMovePlayer = true;
        }
        else if(!isPlayer && !canMakeMoveAI && AIIterations < 5)
        {
            AISum += (dice1num + dice2num) * 4;
            canMakeMoveAI = true;
        }
        ManageScore(isPlayer);
    }

    public void TakeQuarter(bool isPlayer)
    {
        if (isPlayer && !canMakeMovePlayer && playerIterations < 5)
        {
            playerSum += (dice1num + dice2num) / 4;
```

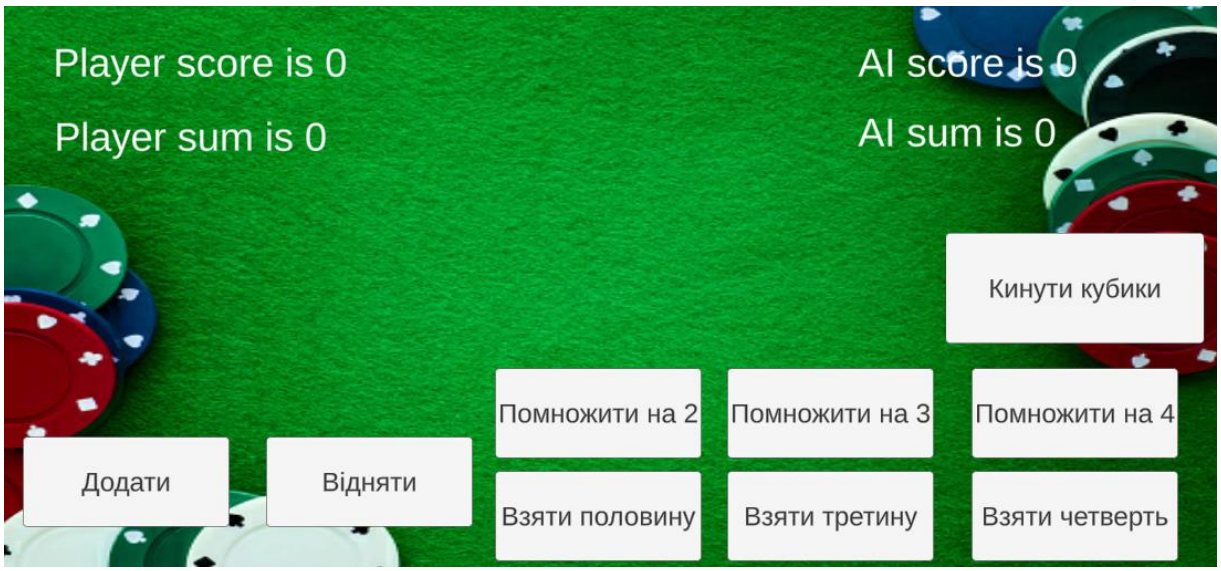
```
        canMakeMovePlayer = true;
    }
    else if(!isPlayer && !canMakeMoveAI && AIIterations < 5)
    {
        AISum += (dice1num + dice2num) / 4;
        canMakeMoveAI = true;
    }

    ManageScore(isPlayer);

}

public void TakeThird(bool isPlayer)
{
    if (isPlayer && !canMakeMovePlayer && playerIterations < 5)
    {
        playerSum += (dice1num + dice2num) / 3;
        canMakeMovePlayer = true;
    }
    else if (!isPlayer && !canMakeMoveAI && AIIterations < 5)
    {
        AISum += (dice1num + dice2num) / 3;
        canMakeMoveAI = true;
    }

    ManageScore(isPlayer);
```

```
}
public void TakeHalf(bool isPlayer)
{
    if (isPlayer && !canMakeMovePlayer && playerIterations < 5)
    {
        playerSum += (dice1num + dice2num) / 2;
        canMakeMovePlayer = true;
    }
    else if (!isPlayer && !canMakeMoveAI && AIIterations < 5)
    {
        AISum += (dice1num + dice2num) / 2;
        canMakeMoveAI = true;
    }

    ManageScore(isPlayer);

}

public void ManageScore(bool isPlayer)
{
    if (isPlayer)
    {
        if (playerSum % 13 == 0)
        {
            playerScore = playerSum/13;
```

```
        }
    }
    else
    {
        if (AISum % 13 == 0)
        {
            AIScore = AISum / 13;
        }
    }

    if(playerIterations == 5)
    {
        canMakeMovePlayer = false;
        AITurn();
    }

    if(AIIterations == 5)
    {
        canMakeMoveAI = false;
    }

    UpdateSum(isPlayer);
    UpdateScore(isPlayer);

}
```

}

Приклади роботи




Player score is 3
Player sum is
39
AI score is 0
AI sum is 0
Кинути кубики
Помножити на 2
Помножити на 3
Помножити на 4
Додати
Відняти
Взяти половину
Взяти третину
Взяти четверть


Player score is 3
Player sum is
55
AI score is 0
AI sum is 0
Кинути кубики
Помножити на 2
Помножити на 3
Помножити на 4
Додати
Відняти
Взяти половину
Взяти третину
Взяти четверть


Player score is 3
Player sum is
55
AI score is 6
AI sum is 78
Кинути кубики
Помножити на 2
Помножити на 3
Помножити на 4
Додати
Відняти
Взяти половину
Взяти третину
Взяти четверть

ВИСНОВОК

В рамках даної лабораторної роботи дослідив і навчився працювати з ігровими алгоритмами. Зокрема зробив невелику гру “Тринадцять” в якій реалізував алгоритм Minimax.