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-13 Нещерет В. О.* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Сопов О. О.* _____
(прізвище, ім'я, по батькові)

Київ 2022

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	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

3 ВИКОНАННЯ

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

MainActivity.kt

```
package com.example.lab6

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.navigation.NavController
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.setupActionBarWithNavController
import com.example.lab6.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

    private lateinit var navController: NavController

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        val binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val navHostFragment =
            supportFragmentManager.findFragmentById(R.id.nav_host_fragment) as
NavHostFragment
        navController = navHostFragment.navController

        setupActionBarWithNavController(navController)
    }

    override fun onSupportNavigateUp(): Boolean {
        return navController.navigateUp() || super.onSupportNavigateUp()
    }
}
```

DifficultyChooseFragment.kt

```
package com.example.lab6

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.navigation.findNavController
import com.example.lab6.databinding.FragmentDifficultyChooseBinding

class DifficultyChooseFragment : Fragment() {

    private lateinit var binding: FragmentDifficultyChooseBinding
    private var difficulty: Int = 0

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentDifficultyChooseBinding.inflate(inflater, container,
```

```
false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.easyButton.setOnClickListener {
            difficulty = 0
            navigateToGameFragment()
        }
        binding.normalButton.setOnClickListener {
            difficulty = 1
            navigateToGameFragment()
        }
        binding.hardButton.setOnClickListener {
            difficulty = 2
            navigateToGameFragment()
        }
    }

    private fun navigateToGameFragment() {

        val action =
DifficultyChooseFragmentDirections.difficultyToGame(difficulty)
        binding.root.findNavController().navigate(action)
    }

}
```

GameFragment.kt

```
package com.example.lab6

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.GridLayoutManager
import com.example.lab6.databinding.FragmentGameBinding


class GameFragment : Fragment() {

    private lateinit var binding: FragmentGameBinding

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentGameBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val difficulty = arguments?.getInt("difficulty")!!

        val recyclerView = binding.boardRecyclerview
        val squares = createSquares()
        val adapter = ChessBoardAdapter(squares, difficulty)
        recyclerView.adapter = adapter
```

```
        val gridLayoutManager = GridLayoutManager(requireContext(), 8)
        gridLayoutManager.orientation = GridLayoutManager.VERTICAL

        recyclerView.layoutManager = gridLayoutManager
    }

    private fun createSquares(): List<ChessSquare> {
        val squares = mutableListOf<ChessSquare>()
        for (x in 0 until 8) {
            for (y in 0 until 8) {
                squares.add(ChessSquare(x, y))
            }
        }

        for (x in 56 until 64 step 2) {
            squares[x].piece = PieceType.HOUND
        }
        squares[3].piece = PieceType.FOX

        return squares
    }

}
```

ChessBoardAdapter.kt

```
package com.example.lab6

import android.annotation.SuppressLint
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageButton
import android.widget.ImageView
import androidx.navigation.findNavController
import androidx.recyclerview.widget.RecyclerView

class ChessBoardAdapter(private val squares: List<ChessSquare>, private val
difficulty : Int)
    : RecyclerView.Adapter<ChessBoardAdapter.ViewHolder>() {

    val viewHolders : MutableList<ViewHolder> = mutableListOf()
    private var activePiecePos : Int = 0
    private val activeMovesPos : MutableList<ChessSquare> = mutableListOf()

    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val imageView: ImageView =
itemView.findViewById(R.id.chess_square_image)
        val pieceButton: ImageButton = itemView.findViewById(R.id.piece_button)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.chess_square_view, parent,
false)
        val vh = ViewHolder(view)
        viewHolders.add(vh)
        return vh
    }

    override fun getItemCount(): Int {
```

```
        return squares.size
    }

    override fun onBindViewHolder(holder: ViewHolder,
@SuppressLint("RecyclerView") position: Int) {

        if (position == 63) {
            foxMove(holder)
        }
        val square = squares[position]
        holder.imageView.setImageResource(getImageResource(square.x, square.y))

        if (square.piece == PieceType.FOX) {
            holder.pieceButton.setImageResource(R.drawable.fox)
            holder.pieceButton.visibility = View.VISIBLE
        }
        else if (square.piece == PieceType.HOUND) {
            holder.pieceButton.setImageResource(R.drawable.dog)
            holder.pieceButton.visibility = View.VISIBLE
        }

        holder.pieceButton.setOnClickListener {

            when (square.piece) {
                PieceType.AVAILABLE_MOVE -> {
                    clearActiveMoves()
                    squares[activePiecePos].piece = null
                    squares[position].piece = PieceType.HOUND

                    holder.pieceButton.setImageResource(R.drawable.dog)
                    holder.pieceButton.visibility = View.VISIBLE
                    viewHolders[activePiecePos].pieceButton.visibility =
View.GONE

                    if (!isHoundWon()) foxMove(holder)
                    else navigateToEndgameFragment(holder.imageView, true)
                }

                PieceType.HOUND -> {
                    clearActiveMoves()

                    activePiecePos = position
                    val moves = square.getAvailableMoves(squares)
                    activeMovesPos.addAll(moves)

                    for (move in moves) {
                        squares[move.x * 8 + move.y].piece =
PieceType.AVAILABLE_MOVE
                        viewHolders[move.x * 8 +
move.y].pieceButton.setImageResource(R.drawable.point)
                        viewHolders[move.x * 8 + move.y].pieceButton.visibility
= View.VISIBLE
                    }
                }
                else -> {
                    clearActiveMoves()
                    activePiecePos = 0
                }
            }
        }
    }

    fun foxMove(holder: ViewHolder) {
        val fox = squares.find { it.piece == PieceType.FOX }!!
```

```
            val moves = fox.getAvailableMoves(squares)
            var move = moves.random()

            for (i in 0..5) {
                move = moves.random()
                if (fox.x < move.x) break
            }

            squares[fox.x * 8 + fox.y].piece = null
            squares[move.x * 8 + move.y].piece = PieceType.FOX

            viewHolders[fox.x * 8 + fox.y].pieceButton.visibility = View.GONE
            viewHolders[move.x * 8 +
move.y].pieceButton.setImageResource(R.drawable.fox)
            viewHolders[move.x * 8 + move.y].pieceButton.visibility = View.VISIBLE

            if (isFoxWon()) {
                navigateToEndgameFragment(holder.imageView, false)
            }
        }

        private fun navigateToEndgameFragment(holder: View, isWon : Boolean, ) {
            val action = GameFragmentDirections.gameToEndgame(isWon, difficulty)
            holder.findNavController().navigate(action)
        }

        private fun isFoxWon() : Boolean {
            for (square in squares) {
                if (square.piece == PieceType.FOX) {
                    return square.x == 7
                }
            }
            return false
        }

        private fun isHoundWon() : Boolean {
            for (square in squares) {
                if (square.piece == PieceType.FOX) {
                    return square.getAvailableMoves(squares).isEmpty()
                }
            }
            return true
        }

        private fun clearActiveMoves() {
            activeMovesPos.forEach {
                viewHolders[it.x * 8 + it.y].pieceButton.visibility = View.GONE
                squares[it.x * 8 + it.y].piece = null
            }
            activeMovesPos.clear()
        }
        private fun getImageResource(x: Int, y: Int): Int {
            return if ((x + y) % 2 == 0) {
                R.drawable.white_square
            } else {
                R.drawable.brown_square
            }
        }
}
```

ChessSquare.kt

```
package com.example.lab6

enum class PieceType {
    FOX, HOUND, AVAILABLE_MOVE
}
class ChessSquare(val x: Int, val y: Int, var piece: PieceType? = null) {

    private fun isFox() = piece == PieceType.FOX

    private fun isHound() = piece == PieceType.HOUND

    fun getAvailableMoves(board : List<ChessSquare>): List<ChessSquare> {
        val moves = mutableListOf<ChessSquare>()
        if (isFox()) {
            moves.add(ChessSquare(x - 1, y - 1))
            moves.add(ChessSquare(x - 1, y + 1))
            moves.add(ChessSquare(x + 1, y - 1))
            moves.add(ChessSquare(x + 1, y + 1))
        } else if (isHound()) {
            moves.add(ChessSquare(x - 1, y - 1))
            moves.add(ChessSquare(x - 1, y + 1))
        }

        val availableMoves : MutableList<ChessSquare> = mutableListOf()
        for (move in moves) {
                if (move.x in 0..7 && move.y in 0..7 && board[move.x * 8 +
move.y].piece == null) {
                    availableMoves.add(move)
                }
            }

        return availableMoves
    }
}
```

EndgameFragment.kt

```
package com.example.lab6

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.findNavController
import com.example.lab6.databinding.FragmentEndgameBinding


class EndgameFragment : Fragment() {

    private lateinit var binding: FragmentEndgameBinding

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentEndgameBinding.inflate(inflater, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
```

```
        val isWon = arguments?.getBoolean("isWon")
        val difficulty = arguments?.getInt("difficulty")!!.toInt()

        if (isWon == true) {
            binding.endgameTitle.text = getString(R.string.win_title)
        } else {
            binding.endgameTitle.text = getString(R.string.lose_title)
        }

        binding.toStartButton.setOnClickListener {
            val action = EndgameFragmentDirections.endgameToDifficulty()
            binding.root.findNavController().navigate(action)
        }
        binding.toGameButton.setOnClickListener {
            val action = EndgameFragmentDirections.endgameToGame(difficulty)
            binding.root.findNavController().navigate(action)
        }
    }
}
```

3.1.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

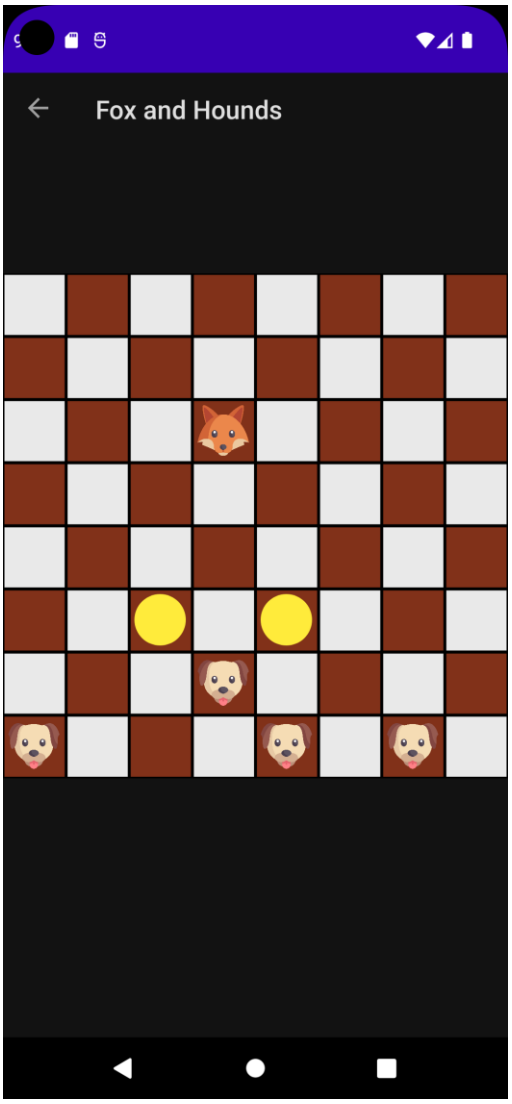

Рисунок 3.1 – Екран гри з вибором ходу

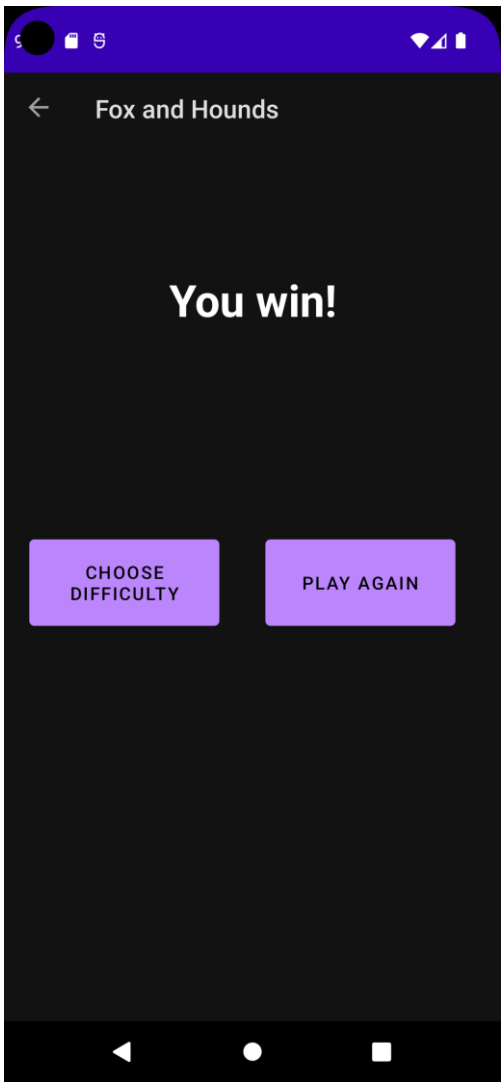

Рисунок 3.2 – Екран завершення гри

ВИСНОВОК

В рамках даної лабораторної роботи я виконав програмну реалізацію гри «Fox and Hounds» з комп’ютерним противником. Реалізував інтерфейс та алгоритм.

КРИТЕРІЇ ОЦІНЮВАННЯ

При здачі лабораторної роботи до 25.12.2022 включно максимальний бал дорівнює – 5. Після 25.12.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

– програмна реалізація – 95%;

– висновок – 5%.

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