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

«Проєктування алгоритмів»

«Пошук в умовах протидії, ігри з повною інформацією, ігри з елементом випадковості, ігри з неповною інформацією»

Варіант 3

Виконав студент ІП-14 Гайдучек Максим Андрійович

(шифр, прізвище, ім'я, по батькові)

Перевірив Соколовський Владислав Володимирович

( прізвище, ім'я, по батькові)

Київ 2022

**Лабораторна робота №6**

**Тема роботи:** Пошук в умовах протидії, ігри з повною інформацією, ігри з елементом випадковості, ігри з неповною інформацією.

**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.

**Варіант:** 3

**Завдання**: Гра «Генерал»

**Хід роботи**

1. Код алгоритму

package com.example.lab6.algorithms;  
  
import com.example.lab6.Combination;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class MinimaxAlgorithm {  
  
 private MinimaxAlgorithm() {  
 }  
  
 public static MinimaxResult findOptimal(int score, boolean[] changes, List<Integer> dices, boolean maxing, int depth) {  
 if (depth <= 0) {  
 return new MinimaxResult(score, changes);  
 }  
  
 MinimaxResult result = new MinimaxResult(Integer.*MIN\_VALUE*, null);  
 boolean[] caseChanges;  
 if (maxing) {  
 for (int i = 0; i <= 5; i++) {  
 for (String change : *createChanges*("", 5, i, 0)) {  
 caseChanges = new boolean[5];  
 for (int j = 0; j < 5; j++) {  
 caseChanges[j] = change.charAt(j) == '1';  
 }  
 MinimaxResult caseResult = *findOptimal*(score, caseChanges, dices, false, depth - 1);  
 if (result.compareTo(caseResult) < 0) {  
 result = caseResult;  
 }  
 }  
 }  
 } else {  
 for (List<Integer> combination : *createDicesCombinations*(new ArrayList<>(), dices, changes, 5, 0)) {  
 Combination caseCombination = Combination.*calcCombination*(combination);  
 int caseScore = caseCombination.getPoints(depth);  
 MinimaxResult caseResult = *findOptimal*(caseScore, changes, combination, true, depth);  
 if (result.compareTo(caseResult) < 0) {  
 result = caseResult;  
 }  
 }  
 }  
  
 return result;  
 }  
  
 private static List<String> createChanges(String head, int n, int ones, int usedOnes) {  
 if (head.length() == n) {  
 return List.*of*(head);  
 }  
  
 List<String> changes = new ArrayList<>();  
 if (head.length() - usedOnes < n - ones) {  
 changes.addAll(*createChanges*(head + "0", n, ones, usedOnes));  
 }  
 if (usedOnes < ones) {  
 changes.addAll(*createChanges*(head + "1", n, ones, usedOnes + 1));  
 }  
 return changes;  
 }  
  
 private static List<List<Integer>> createDicesCombinations(List<Integer> head, List<Integer> dices,  
 boolean[] changes, int n, int count) {  
 if (head.size() == n) {  
 return List.*of*(head);  
 }  
 if (!changes[count]) {  
 List<Integer> combination = new ArrayList<>(head);  
 combination.add(dices.get(count));  
 return *createDicesCombinations*(combination, dices, changes, n, count + 1);  
 }  
  
 List<List<Integer>> combinations = new ArrayList<>();  
 for (int i = 1; i <= 6; i++) {  
 List<Integer> combination = new ArrayList<>(head);  
 combination.add(i);  
 combinations.addAll(*createDicesCombinations*(combination, dices, changes, n, count + 1));  
 }  
 return combinations;  
 }  
}

package com.example.lab6.algorithms;  
  
import java.util.Arrays;  
  
public class MinimaxResult implements Comparable<MinimaxResult> {  
  
 private final int score;  
 private final boolean[] changes;  
  
 MinimaxResult(int score, boolean[] changes) {  
 this.score = score;  
 this.changes = changes;  
 }  
  
 public boolean[] getChanges() {  
 return changes;  
 }  
  
 @Override  
 public int compareTo(MinimaxResult o) {  
 if (score == Integer.*MIN\_VALUE*) return -1;  
 return score - o.score;  
 }  
  
 @Override  
 public String toString() {  
 return "MinimaxResult{" +  
 "score=" + score +  
 ", changes=" + Arrays.*toString*(changes) +  
 '}';  
 }  
}

1. Приклади роботи алгоритму

На рисунках 2.1 і 2.2 показані приклади роботи програми.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.1 – Приклад роботи алгоритму

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.2 – Приклад роботи алгоритму

**Висновок**: під час виконання даної лабораторної роботи було створено гру «Генерал» для гри користувача з комп’ютером (ботом). Було розроблено графічний інтерфейс користувача за допомогою JavaFX. Штучний інтелект вибору варіанту ходу бота було реалізовано на основі алгоритму мінімакс.