МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
КАФЕДРА 25

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Доцент к.т.н.		Е. М. Линский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ ПО КУРСОВОЙ РАБОТЕ

”ИГРА В ХЕКСАГОН (ГЕКСАГОН)”

по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

Студент гр. №	2351		Е. А. Лысов
		подпись, дата	инициалы, фамилия

Санкт-Петербург, 2024

Содержание

Постановка задачи

Задачей данной курсовой работы является разработка программы, которая моделирует игру в Гексагон, создает игровое поле и анализирует стратегии для достижения победы. Программа должна учитывать правила игры и предоставлять пользователю возможность визуализировать игровое поле, а также проводить анализ наилучших ходов.

Игра в Гексагон осуществляется на специальной ромбической доске, состоящей из шестиугольных ячеек, которая может быть любого размера, но традиционно используется размер n×n (чаще всего 11×11). На доске каждое поле в центре граничит с шестью соседними полями, а угловые поля имеют 2 или 3 соседа.

Для игрока важным свойством является возможность построения цепи из своих фишек, соединяющей две стороны его цвета (красную или синюю). Задача программы — анализировать возможные стратегии и находить оптимальные ходы с целью создания такой цепи.

Из литературы, которая помогает понять правила и стратегии игры, можно обратиться к книге Сильвии Назар «Игры разума», где подробно описываются принципы и аналитика игр, включая Гексагон.

Примеры решения задачи

Пример 1: Простой случай

На доске 5x5 у красного игрока фишки расположены следующим образом:

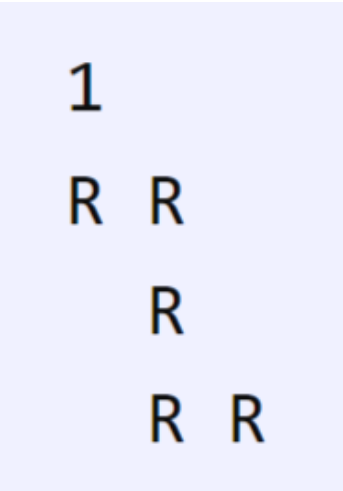

Рис. 1: Расположение фишек красного игрока. Где R — красные фишки и 1 — свободное поле.

Где R — красные фишки и 1 — свободное поле. Красный игрок может выиграть, расположив свою следующую фишку, чтобы соединить две красные стороны.

Пример 2: Неоптимальный ход

На доске 5x5 у синего игрока фишки расположены так:

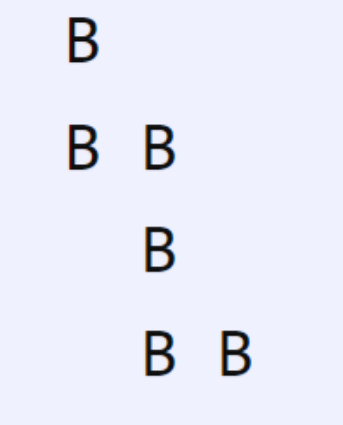

Рис. 2: Расположение фишек синего игрока.

Синему игроку нужно учесть, что размещение следующей фишки не должно дать красному игроку возможность создать цепь из своих фишек.

Эти примеры демонстрируют важность стратегического мышления и анализа ходов, что и должно быть реализовано в разработанной программе.

Описание алгоритма

Основные идеи алгоритма: Алгоритм игры в Гексагон строится на стратегии поиска оптимальных ходов для обоих игроков, а также на анализе состояния игрового поля. Основные идеи включают:

1. Анализ соседей. Каждое поле имеет от 2 до 6 соседей. Игроки должны учитывать, как их ход влияет на доступные варианты для противника.

2. Поиск цепей. Необходимость находить все возможные цепи для каждого игрока, начиная с границ их сторон.

3. Минимизация ходов противника. Алгоритм выявляет наилучшие ответы на ходы соперника, блокируя их возможности.

4. Рекурсивный поиск. Использование методов поиска с возвратом (backtracking) для оценки возможных состояний.

Описание структур данных:

- Доска представляется двумерным массивом для представления игрового поля. Элементы массива могут быть пустыми, красными или синими.

- Для каждого поля находится список его соседей, что позволяет быстро изменять состояние поля и проверять возможные ходы.

- Для хранения текущих состояний во время поиска используется стек.

Шаги алгоритма:

1. Инициализация доски.

2. Поочередные ходы игроков.

3. Проверка состояния доски после каждого хода на наличие цепей.

4. Если найдено решение, завершить игру; если нет, перейти к следующему ходу.

Пошаговое выполнение алгоритма на примере

Начальная ситуация:

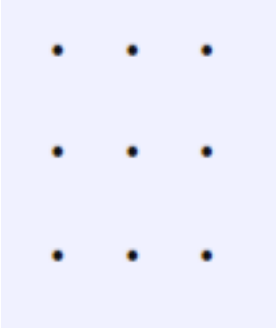

Рис. 3: Начальная ситуация.

Первый ход (Синий):

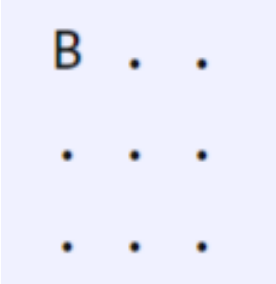

Рис. 4: Первый ход.

Второй ход (Красный):

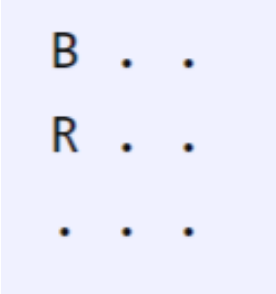

Рис. 5: Второй ход.

Третий ход (Синий):

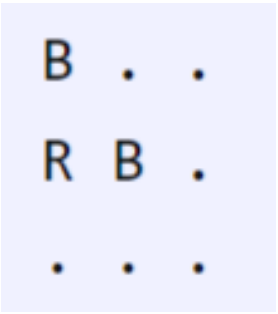

Рис. 6: Третий ход.

Четвертый ход (Красный):

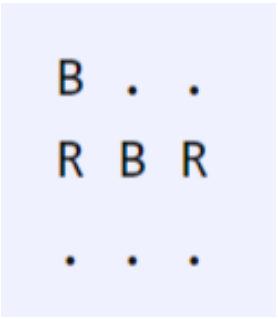

Рис. 7: Четвертый ход.

Псевдокод

Класс HexGame

Переменные:

board: двумерный массив Player (размер BOARD_SIZE x BOARD_SIZE)

currentPlayer: Player (RED или BLUE)

firstMoveDone: Логическая переменная

isBoardFlipped: Логическая переменная

Конструктор HexGame

board = [Создать BOARD_SIZE x BOARD_SIZE, заполнить NONE]

currentPlayer = BLUE

firstMoveDone = ЛОЖЬ

isBoardFlipped = ЛОЖЬ

Метод displayBoard

Вывести " " // Заголовок столбцов

Для i от 0 до BOARD_SIZE

Вывести i + " " // Номер строки

Для j от 0 до BOARD_SIZE

Если isBoardFlipped == ИСТИНА

player = getFlippedPlayer(board[j][i]) // Получить перевернутого игрока

Иначе

player = board[i][j]

Если player == RED

Вывести "R "

Иначе Если player == BLUE

Вывести "B "

Иначе

Вывести ". "

```
        Вывести новую строку

Метод makeMove(row, col)
    Если row и col валидны и board[row][col] == NONE
        board[row][col] = currentPlayer // Установить ход текущего игрока
        Если firstMoveDone == ЛОЖЬ
            firstMoveDone = ИСТИНА
            Вывести (" совершил первый ход. Повернуть доску? (y/n): ")
            choice = ВводИспользователя()
            Если choice == 'y' или choice == 'Y'
                isBoardFlipped = ИСТИНА
                currentPlayer = (currentPlayer == BLUE) ? RED : BLUE
                Вывести "Доска перевернута. Игрок сменился")
        currentPlayer = (currentPlayer == BLUE) ? RED : BLUE
    Иначе
        Вывести "Некорректный ход. Попробуйте снова."

Метод getCurrentPlayer
    Вернуть currentPlayer

Метод checkWin
    Вернуть (checkWinForPlayer(RED) или checkWinForPlayer(BLUE)) // Проверка победы

Метод winner
    Если checkWinForPlayer(RED)
        Вернуть RED
    Если checkWinForPlayer(BLUE)
        Вернуть BLUE
    Вернуть NONE

Метод getFlippedPlayer(player)
```

Если isBoardFlipped == ИСТИНА

Если player == RED

Вернуть BLUE

Иначе Если player == BLUE

Вернуть RED

Вернуть NONE

Вернуть player

Метод checkWinForPlayer(player)

Создать массив visited (BOARD_SIZE x BOARD_SIZE, значениями ЛОЖЬ)

Создать очередь q

Если player == RED

Для i от 0 до BOARD_SIZE

Если board[0][i] == RED

q.Добавить((0, i))

visited[0][i] = ИСТИНА

Иначе Если player == BLUE

Для i от 0 до BOARD_SIZE

Если board[i][0] == BLUE

q.Добавить((i, 0))

visited[i][0] = ИСТИНА

Определить направления для проверки соседей (направления = [(0, 1), (1, 0), (1, -1), (0, -1), (-1, 0), (-1, 1)])

Пока q не пуста

p = q.УдалитьПервый() // Получить координаты из очереди

Если (RED и первая == BOARD_S - 1) или (BLUE и вторая == BOARD_S - 1)

Вернуть ИСТИНА

Конец Если

Для каждого dir в направлениях

nx = p.первая_координата + dir.первая_координата

ny = p.вторая_координата + dir.вторая_координата

Если (nx валиден и ny валиден и board[nx][ny] == player и visited[nx][ny] == ЛОЖЬ)

visited[nx][ny] = ИСТИНА

q.Добавить((nx, ny))

Конец Для

Конец Пока

Вернуть ЛОЖЬ

Сложность алгоритма

Анализ сложности алгоритма будет зависеть от размеров доски и количества возможных ходов. Наиболее простой расчет дает сложность:

- Проверка всех соседей для каждого хода требует $O(6)$ времени, так как у каждого поля не более 6 соседей. Однако $O(6)$ не используется, так как 6 — константа, а значит, сложность этой операции относится к $O(1)$.
- Рекурсивный поиск состояния в худшем случае ведет к $O(n^2)$ состояниям на доске $n \times n$.

Таким образом, после оптимизации, общая сложность алгоритма $O(n^2)$.

Инструкция пользователя

Программа запускается пользователем из командной строки. Командная строка принимает три параметра: имя проекта с кодом (например, ‘hexagon.exe‘), имя файла, в котором находятся исходные данные, то есть поочередные ходы игроков в формате первая координата x, а вторая координата y (например, ‘input.txt‘); имя файла, в который будет записан результат после выполнения программы методом полного перебора (например, ‘output.txt‘). Входные и выходные файлы должны быть в формате .txt.

Тестовые примеры

Тест 1

Входные данные:

```
0 0
y
0 1
10 0
0 2
9 0
0 3
7 0
0 4
3 2
0 5
3 3
0 6
7 9
0 7
8 9
0 8
5 10
0 9
3 10
0 10
```

Выходные данные:

```
Current Player: Blue
   0 1 2 3 4 5 6 7 8 9 10
0 R . . . . . B . B B
 1 R . . . . . .
  2 R . . B . . . . . .
   3 R . . B . . . . . .
    4 R . . . . . .
     5 R . . . . . .
      6 R . . . . . .
       7 R . . . . . .
        8 R . . . . . .
         9 R . . . . . B B . .
          10 . . . . B . . . . .
Current Player: Red
   0 1 2 3 4 5 6 7 8 9 10
0 R . . . . . B . B B
 1 R . . . . . .
  2 R . . B . . . . . .
   3 R . . B . . . . . .
    4 R . . . . . .
     5 R . . . . . .
      6 R . . . . . .
       7 R . . . . . .
        8 R . . . . . .
         9 R . . . . . B B . .
          10 . . . B . B . . . . .
Current Player: Blue
   0 1 2 3 4 5 6 7 8 9 10
0 R . . . . . B . B B
 1 R . . . . . .
  2 R . . B . . . . . .
   3 R . . B . . . . . .
    4 R . . . . . .
     5 R . . . . . .
      6 R . . . . . .
       7 R . . . . . .
        8 R . . . . . .
         9 R . . . . . B B . .
          10 R . . B . B . . . . .
Congratulations! Blue Wins!
```

Тест 2

Входные данные:

```
0 0
n
1 0
0 1
1 1
0 2
1 2
0 3
1 3
0 4
1 4
0 5
1 5
0 6
1 6
0 7
1 7
0 8
1 8
0 9
1 9
10 10
1 10
```

Выходные данные:

```
Current Player: Blue
   0 1 2 3 4 5 6 7 8 9 10
0 B B B B B B B B B B .
 1 R R R R R R R R R .
  2 . . . . . . . . . .
   3 . . . . . . . . . .
    4 . . . . . . . . . .
     5 . . . . . . . . . .
      6 . . . . . . . . . .
       7 . . . . . . . . . .
        8 . . . . . . . . . .
         9 . . . . . . . . . .
          10 . . . . . . . . . B
  RED zone: Row 0 to 4  |  BLUE zone: Row 5 to 10
Current Player: Red
   0 1 2 3 4 5 6 7 8 9 10
0 B B B B B B B B B B .
 1 R R R R R R R R R R
  2 . . . . . . . . . .
   3 . . . . . . . . . .
    4 . . . . . . . . . .
     5 . . . . . . . . . .
      6 . . . . . . . . . .
       7 . . . . . . . . . .
        8 . . . . . . . . . .
         9 . . . . . . . . . .
          10 . . . . . . . . . B
  RED zone: Row 0 to 4  |  BLUE zone: Row 5 to 10
Current Player: Blue
```

Тест 3

Входные данные:

```
0 0
n
0 1
1 1
0 2
2 2
0 3
3 3
0 4
4 4
0 5
5 5
0 6
6 6
0 7
7 7
0 8
8 8
0 9
9 9
5 5
10 10
```

Выходные данные:

```
Current Player: Red
   0 1 2 3 4 5 6 7 8 9 10
0 B R R R R R R R R R .
 1 . B . . . . . . . . .
  2 . . B . . . . . . . .
   3 . . . B . . . . . . .
    4 . . . . B . . . . . .
     5 . . . . . B . . . . .
      6 . . . . . . B . . . .
       7 . . . . . . . B . . .
        8 . . . . . . . . B . .
         9 . . . . . . . . . B .
          10 . . . . . . . . . . .
  RED zone: Row 0 to 4  |  BLUE zone: Row 5 to 10
Current Player: Red
   0 1 2 3 4 5 6 7 8 9 10
0 B R R R R R R R R R R
 1 . B . . . . . . . . .
  2 . . B . . . . . . . .
   3 . . . B . . . . . . .
    4 . . . . B . . . . . .
     5 . . . . . B . . . . .
      6 . . . . . . B . . . .
       7 . . . . . . . B . . .
        8 . . . . . . . . B . .
         9 . . . . . . . . . B .
          10 . . . . . . . . . . B
  RED zone: Row 0 to 4  |  BLUE zone: Row 5 to 10
Current Player: Blue
Congratulations! Blue Wins!
```

Список литературы

1. Уэзерелл Ч. Этюды для программистов. – 1982.

2. Седжвик Р. Фундаментальные алгоритмы на С. Анализ/Структуры данных/Сортировка //Поиск–СПб.: ДиаСофтЮП. – 2001.

3. Назар С. Игры разума. История жизни Джона Нэша, гениального математика и лауреата Нобелевской премии //М.: АСТ: CORPUS. – 2017. – Т. 747.