МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра теории вероятностей и компьютерного моделирования

**Лабораторная работа № 6**

**По спецкурсу «Теория сложности алгоритмов»**

**Игровые задачи**

Выполнил: Покхарел П.К.

Группа: М8О-101М-22, Вариант 4

Преподаватель: Рассказова В.А.

Москва, 2023

**Задание.** Для модифицированного поля игры в «крестики-нолики»

1. построить программными средствами ориентированный граф состояний игры;

2. определить существует ли выигрышная стратегия для первого игрока;

3. реализовать данную стратегию (если она существует) или стратегию приводящую к гарантированной ничьей (если выигрышной стратегии не существует) в виде интерактивной игры, в которой первый ход делает компьютер, а оппонентом выступает игрок.

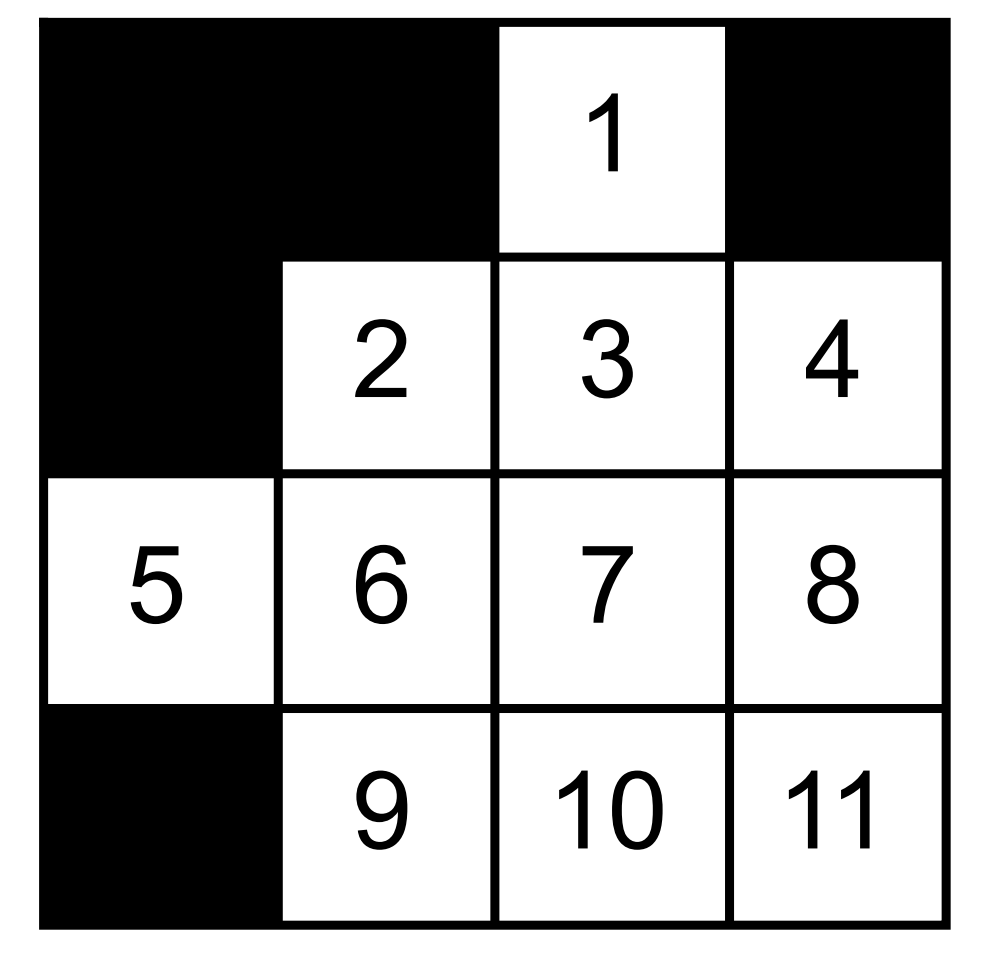
Поле для игры представлено на рисунке 1.

Рис. 1 - поле для игры в “крестики-нолики”

**Состояния игры**

На рисунке 2 представлен ориентированный граф состояний игры.

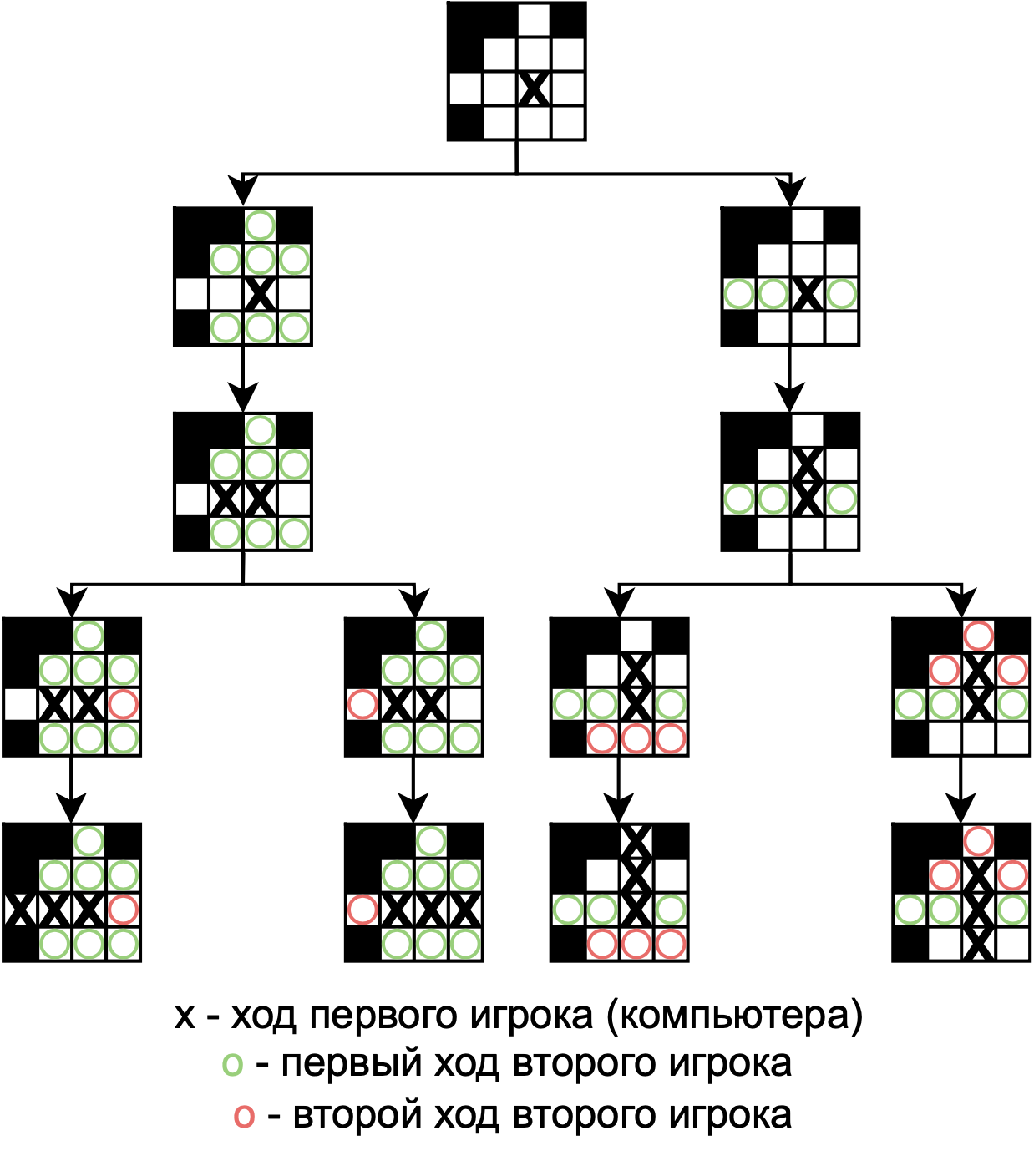
****

Рис. 2 - диаграмма состояний игры с выигрышной стратегией первого игрока (компьютера) при всевозможных ходах второго игрока

**Выигрышная стратегия**

Выигрышная стратегия существует - для того чтобы гарантированно победить в данной игре необходимо:

1. Поставить первый крестик на клетку с номером “7”, а затем заполнить крестиками одну из двух четырехклеточных полос в зависимости от первого хода второго игрока

2. Поставить крестик на клетку с номером “6” или “3”

3. После второго хода второго игрока первый игрок выигрывает, поставив крестик на клетку с номером

- “1” или “10” - если предыдущий крестик поставлен на клетку “3”

- “5” или “8” - если предыдущий крестик поставлен на клетку “6”

**Программная реализация**

def return\_next\_turns(turns\_list):

new\_turns\_list = []

for new\_pos in allow\_positions:

if new\_pos not in turns\_list:

new\_turns\_list.append(turns\_list + [new\_pos])

return new\_turns\_list

from copy import deepcopy

win\_triple = [{2, 3, 4}, {6, 7, 8}, {9, 10, 11}, {2, 6, 9}, {3, 7, 10}, {4, 8, 11}, {5, 6, 7}, {1, 3, 7}, {2, 7, 11}, {4, 7, 9}]

allow\_positions = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

allow\_positions\_real = [(0, 2), (1, 1), (1, 2), (1, 3), (2, 0), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)]

game\_len = len(allow\_positions)

all\_games\_list = [[x] for x in allow\_positions]

all\_games\_list1 = []

win\_list = []

win\_count = [0] \* 11

lose\_start\_positions = set([])

i = 0

while i < 5: #game\_len-1:

for pos in all\_games\_list:

#Проверим, что крестики еще не приняли выигрышную позицию

if True in [x.issubset(set(pos[::2])) for x in win\_triple]:

win\_list.append(pos)

win\_count[pos[0]-1] += 1

#Проверим, что нолики еще не приняли выигрышную позицию

elif True in [x.issubset(set(pos[1::2])) for x in win\_triple]:

lose\_start\_positions.add(pos[0])

pass

else:

for x in return\_next\_turns(pos):

all\_games\_list1.append(x)

all\_games\_list = deepcopy(all\_games\_list1)

all\_games\_list1 = []

i += 1

print('Крестики-нолики')

print('')

board\_help = [[' #', ' #', ' 1',' #'], [' #',' 2',' 3',' 4'], [' 5',' 6',' 7',' 8'], [' #',' 9','10','11']]

print('Чтобы отметить нолик, вам нужно выбрать число от 1 до 11, которое еще не занято')

print('Компьютер делает первый ход')

print('')

for y in board\_help:

print(y)

print('')

best\_position = win\_count.index(max(win\_count)) + 1

strategy\_list = [x for x in win\_list if x[0] == best\_position]

# print(strategy\_list)

game\_list = [best\_position]

m = 1

board = [['##','##','##','##'], ['##','##','##','##'], ['##','##','##','##'], ['##','##','##','##']]

discard\_list = [best\_position-1]

best\_position\_r = allow\_positions\_real[best\_position-1]

board[best\_position\_r[0]][best\_position\_r[1]] = ' X'

print('Партия начинается. Компьютер делает ход...')

print('')

for y in board:

print(y)

print('')

while m < 5:

# if m != 0:

# for y in board\_help:

# print(y)

new\_turn = int(input('Ваш ход... '))

print('')

game\_list.append(new\_turn)

new\_position\_r = allow\_positions\_real[new\_turn-1]

board[new\_position\_r[0]][new\_position\_r[1]] = ' 0'

board\_help[new\_position\_r[0]][new\_position\_r[1]] = ' 0'

new\_strategy\_list = [x for x in strategy\_list if x[m] == new\_turn]

# print(new\_strategy\_list[0])

strategy\_list = deepcopy(new\_strategy\_list)

m += 1

for y in board:

print(y)

win\_count\_c = [0] \* 11

# print('count')

for pos in new\_strategy\_list:

win\_count\_c[pos[m]-1] += 1

# print(win\_count\_c)

# for x in discard\_list:

# win\_count[x] = 0

# print(win\_count)

# new\_strategy\_list1 = deepcopy(new\_strategy\_list)

# new\_strategy\_list1.sort(key=len)

best\_position\_c = win\_count\_c.index(max(win\_count\_c)) + 1

# best\_position = new\_strategy\_list1[0][m]

# best\_position = new\_strategy\_list[0][m]

# print(best\_position\_c)

# print(new\_strategy\_list)

# discard\_list.append(best\_position-1)

best\_position\_r = allow\_positions\_real[best\_position\_c-1]

board[best\_position\_r[0]][best\_position\_r[1]] = ' X'

new\_strategy\_list = [x for x in strategy\_list if x[m] == best\_position\_c]

strategy\_list = deepcopy(new\_strategy\_list)

print('Компьютер делает ход... ')

print('')

for y in board:

print(y)

# print(m)

# m += 2

m += 1

if m == 5:

print('Вы проиграли :(')