**Министерство образования и науки Украины**

**Национальный технический университет Украины "Киевский политехнический институт имени Игоря Сикорского"**

**Факультет информатики и вычислительной техники**

**Кафедра автоматизированных систем обработки**

**информации и управления**

**ОТЧЕТ**

по лабораторной работе № 6 по дисциплине

«Проектирование и анализ вычислительных алгоритмов»

„ **Поиск в условиях противодействия** ”

**Выполнил**

(шифр, фамилия, имя, отчество)

*Кушка М.О., ІП-61*

**Проверил**

(фамилия, имя, отчество )

*Головченко М.Н.*

Киев 2018

Содержание

[1 Цель лабораторной работы 3](#_Toc510983936)

[2 Задание 4](#_Toc510983937)

[3 Выполнение 7](#_Toc510983938)

[3.1 Псевдокод алгоритма 7](#_Toc510983939)

[3.2 Анализ временной сложности 7](#_Toc510983940)

[3.3 Программная реализация алгоритма 7](#_Toc510983941)

[3.3.1 Исходный код 7](#_Toc510983942)

[3.3.2 Примеры работы 7](#_Toc510983943)

[3.4 Испытания алгоритма 8](#_Toc510983944)

[3.4.1 Временные оценочные характеристики 8](#_Toc510983945)

[3.4.2 Графики зависимости временных оценочных характеристик и времени поиска от размерности структур 9](#_Toc510983946)

[Выводы 10](#_Toc510983947)

[Критерии оценивания 11](#_Toc510983948)

# Цель лабораторной работы

Цель работы – изучить основные подходы к формализации алгоритмов нахождения решений задач в условиях.

# Задание

Согласно варианту (таблица 2.1) реализовать визуальное игровое приложение для игры пользователя с компьютерным оппонентом. Для реализации стратегии игры компьютерного оппонента использовать алгоритм альфа-бета-отсечений.

Реализовать три уровня сложности (легкий, средний, сложный) +1балл.

Сделать обобщенный вывод по лабораторной работе.

Таблица 2.1 – Варианты

|  |  |
| --- | --- |
| **№** | **Вариант** |
| 1 | Баше https://ru.wikipedia.org/wiki/Баше\_(игра) |
| 2 | Hexapawn https://ru.wikipedia.org/wiki/Hexapawn |
| 3 | Точки https://ru.wikipedia.org/wiki/Точки\_(игра) |
| 4 | Dots and Boxes https://ru.wikipedia.org/wiki/Палочки\_(игра) |
| 5 | Сим https://ru.wikipedia.org/wiki/Сим\_(игра) |
| 6 | Snakes http://www.papg.com/show?3AE4 |
| 7 | Cram https://en.wikipedia.org/wiki/Cram\_(game) |
| 8 | Chomp http://www.papg.com/show?3AEA |
| 9 | Obstruction http://www.papg.com/show?2XMX |
| 10 | Gale http://www.papg.com/show?1TPI |
| 11 | Гомоку https://ru.wikipedia.org/wiki/Гомоку |
| 12 | Ним https://ru.wikipedia.org/wiki/Ним\_(игра) |
| 13 | Col http://www.papg.com/show?2XLY |
| 14 | Hackenbush http://www.papg.com/show?1TMP |
| 15 | Snort http://www.papg.com/show?2XM1 |
| 16 | Race Track http://www.papg.com/show?1TPE |
| 17 | 3D Noughts and Crosses http://www.papg.com/show?1TND |
| 18 | Domineering http://www.papg.com/show?1TX6 |
| 19 | Баше https://ru.wikipedia.org/wiki/Баше\_(игра) |
| 20 | Hexapawn https://ru.wikipedia.org/wiki/Hexapawn |
| 21 | Точки https://ru.wikipedia.org/wiki/Точки\_(игра) |
| 22 | Dots and Boxes https://ru.wikipedia.org/wiki/Палочки\_(игра) |
| 23 | Сим https://ru.wikipedia.org/wiki/Сим\_(игра) |
| 24 | Snakes http://www.papg.com/show?3AE4 |
| 25 | Cram https://en.wikipedia.org/wiki/Cram\_(game) |
| 26 | Chomp http://www.papg.com/show?3AEA |
| 27 | Obstruction http://www.papg.com/show?2XMX |
| 28 | Gale http://www.papg.com/show?1TPI |
| 29 | Гомоку https://ru.wikipedia.org/wiki/Гомоку |
| 30 | Ним https://ru.wikipedia.org/wiki/Ним\_(игра) |

# Выполнение

## Программная реализация

### Исходный код

##############

## NIM game ##

##############

# from tkinter import Tk, Canvas, Frame, Button, LEFT, RIGHT, messagebox, CURRENT, Label, Entry, Checkbutton

from tkinter import \*

from random import randint

def on\_click(event): #this deals with actions from clicks based on the name of the button clicked on

if canvas.find\_withtag(CURRENT):

global last\_piece, piece\_name

piece\_name = canvas.gettags(CURRENT)[0]

group\_name = piece\_name[0]

if pieces == [7,5,3]:

last\_piece = None

try:

if piece\_name == "AI\_first":

AI\_to\_play()

elif group\_name != last\_piece and last\_piece != None and piece\_name != 'DONE':

# display ILLEGAL MOVE in the game canvas for 1.5 seconds if the user tries to pick pieces from different piles on the same turn

canvas.create\_text(355, 40, text="ILLEGAL MOVE", font="Purisa", tags="ILLEGAL\_WARNING", fill="black",command=None)

canvas.update\_idletasks()

canvas.after(1500)

canvas.delete("ILLEGAL\_WARNING")

else:

if piece\_name == 'DONE' and last\_piece != 'DONE' and last\_piece != None:

last\_piece = None

AI\_to\_play() # this is for the computer's turn when the user has clicked the Done button

elif piece\_name == 'DONE' and last\_piece == None:

# do not let the user click the done button more than once in a row. Displays the message for 1.5 seconds

canvas.create\_text(355, 40, text="YOU HAVE NOT MADE ANY MOVES", font="Purisa", tags="DOUBLE\_DONE",fill="black", command=None)

canvas.update\_idletasks()

canvas.after(1500)

canvas.delete("DOUBLE\_DONE")

elif piece\_name == 'WON\_BUTTON':

pass

else:

canvas.delete('AI\_first')

update\_board(piece\_name)

canvas.delete(piece\_name)

last\_piece = piece\_name[0]

if sum(pieces) == 0:

game\_over('computer')

except NameError:

last\_piece = group\_name

if piece\_name == 'DONE':

AI\_to\_play() # this is the computer's turn when the user has clicked the Done button

elif piece\_name == 'WON\_BUTTON':

pass

else:

update\_board(piece\_name)

canvas.delete(piece\_name)

# this should only happen on first go and is to catch the case where the last button press is not defined

last\_piece = piece\_name[0]

if sum(pieces) == 0:

game\_over('computer')

def create\_pieces():

# ititialise the mathematical representations of the board and declare as global variable so it can easily be updated within the update function

global pieces, board, piece\_name

pieces = [7, 5, 3]

board = [[1,1,1,1,1,1,1],[1,1,1,1,1],[1,1,1]]

piece\_name = 'NEW\_GAME'

circle\_size = 50

linecolour = "black"

fillcolour = "blue"

# Group A is on the left. It is a group of 7

A1x = 50

A1y = 50

A2x = 110

A2y = 50

A3x = 20

A3y = 105

A4x = 80

A4y = 105

A5x = 140

A5y = 105

A6x = 50

A6y = 160

A7x = 110

A7y = 160

canvas.create\_oval(A1x, A1y, A1x+circle\_size, A1y+circle\_size, outline=linecolour,fill=fillcolour,tags="A1")

canvas.create\_oval(A2x, A2y, A2x+circle\_size, A2y+circle\_size, outline=linecolour,fill=fillcolour,tags="A2")

canvas.create\_oval(A3x, A3y, A3x+circle\_size, A3y+circle\_size, outline=linecolour,fill=fillcolour,tags="A3")

canvas.create\_oval(A4x, A4y, A4x+circle\_size, A4y+circle\_size, outline=linecolour,fill=fillcolour,tags="A4")

canvas.create\_oval(A5x, A5y, A5x+circle\_size, A5y+circle\_size, outline=linecolour,fill=fillcolour,tags="A5")

canvas.create\_oval(A6x, A6y, A6x+circle\_size, A6y+circle\_size, outline=linecolour,fill=fillcolour,tags="A6")

canvas.create\_oval(A7x, A7y, A7x+circle\_size, A7y+circle\_size, outline=linecolour,fill=fillcolour,tags="A7")

# Group B is in the centre. It is a group of 5

B1x = 330

B1y = 65

B2x = 280

B2y = 105

B3x = 380

B3y = 105

B4x = 300

B4y = 160

B5x = 360

B5y = 160

canvas.create\_oval(B1x, B1y, B1x+circle\_size, B1y+circle\_size, outline=linecolour,fill=fillcolour,tags="B1")

canvas.create\_oval(B2x, B2y, B2x+circle\_size, B2y+circle\_size, outline=linecolour,fill=fillcolour,tags="B2")

canvas.create\_oval(B3x, B3y, B3x+circle\_size, B3y+circle\_size, outline=linecolour,fill=fillcolour,tags="B3")

canvas.create\_oval(B4x, B4y, B4x+circle\_size, B4y+circle\_size, outline=linecolour,fill=fillcolour,tags="B4")

canvas.create\_oval(B5x, B5y, B5x+circle\_size, B5y+circle\_size, outline=linecolour,fill=fillcolour,tags="B5")

# Group C is on the right. It is a group of 3

C1x = 570

C1y = 105

C2x = 540

C2y = 160

C3x = 600

C3y = 160

canvas.create\_oval(C1x, C1y, C1x+circle\_size, C1y+circle\_size, outline=linecolour,fill=fillcolour,tags="C1")

canvas.create\_oval(C2x, C2y, C2x+circle\_size, C2y+circle\_size, outline=linecolour,fill=fillcolour,tags="C2")

canvas.create\_oval(C3x, C3y, C3x+circle\_size, C3y+circle\_size, outline=linecolour,fill=fillcolour,tags="C3")

def create\_DONE\_button():

canvas.create\_rectangle(580,2,680,45,outline="black",fill="gray80",tags="DONE")

canvas.create\_text(630,23,text="I'm done with\n my turn",font="Purisa",tags="DONE")

def create\_computer\_go\_first\_button():

canvas.create\_rectangle(580,48,680,91,outline="black",fill="gray80",tags="AI\_first")

canvas.create\_text(630,70,text="The computer\n can go first",font="Purisa",tags="AI\_first")

def create\_operation\_buttons():

# create the buttons to start the game and show the rules

operation\_frame = Frame()

operation\_frame.pack(fill="both", expand=True)

Start = Button(operation\_frame, text='Click here to start a new game', height=2, command=start\_game, bg='white',fg='navy')

Start.pack(fill="both", expand=True, side=LEFT)

Rules = Button(operation\_frame, text='Click here to see the rules', command=show\_rules, height=2, bg='navy',fg='white')

Rules.pack(fill="both", expand=True, side=RIGHT)

def start\_game():

# this turns all the ovals into buttons to be activated by a mouse click. Function then jumps to on\_click.

# lvl = Toplevel()

# Label(lvl, text='Message!').pack()

global box

box = Frame()

w = Button(box, text="Low lvl", width=10, command=low\_lvl)

w.pack(side=LEFT, padx=5, pady=5)

w = Button(box, text="Middle lvl", width=10, command=middle\_lvl, default=ACTIVE)

w.pack(side=LEFT, padx=5, pady=5)

w = Button(box, text="High lvl", width=10, command=high\_lvl)

w.pack(side=LEFT, padx=5, pady=5)

# bind("&lt;Return>", self.ok)

# bind("&lt;Escape>", self.cancel)

box.pack()

create\_pieces()

create\_DONE\_button()

create\_computer\_go\_first\_button()

canvas.delete('WON\_BUTTON')

canvas.bind("<Button-1>",func=on\_click)

def low\_lvl():

global box, lvl

lvl = 1

box.forget()

def middle\_lvl():

global box, lvl

lvl = 2

box.forget()

def high\_lvl():

global box, lvl

lvl = 3

box.forget()

def show\_rules():

rules\_intro = 'Welcome to Nim, a game with more strategy than may first appear!\n'

game\_play = 'Playing Nim involves each player taking pieces from the game screen in turns.\n'

rule1 = 'Your goal is to leave your opponent with the last piece remaining on the screen.\n'

rule2 = 'You may only take from one pile each turn.\n'

rule3 = 'You can take as many pieces as you want each turn.\n\n'

operation = 'When you are done with your turn, please click the button in the top right corner to let the computer know that it can play.'

rule\_message = rules\_intro+game\_play+rule1+rule2+rule3+operation

messagebox.showinfo("How to play Nim", rule\_message)

def update\_board(piece\_names):

# the board had two representations, the "pieces" representation is [7,5,3].

# the "board" representation has all the pieces and looks like [[1,1,1,1,1,1,1],[1,1,1,1,1],[1,1,1]].

if type(piece\_names) == str: #need to tell if there is a single or multiple updates to be done

update\_board\_pieces(piece\_names)

else:

for item in range(0, len(piece\_names)): # multiple updates are required when the AI makes its moves

piece\_name = piece\_names[item]

update\_board\_pieces(piece\_name)

def update\_board\_pieces(piece\_name):

# this part adjusts the mathematical representations of the board, being the [7,5,3] and [[1,1,1,1,1,1,1],[1,1,1,1,1],[1,1,1]] arrays.

# these arrays are global variables defined when the board is created

group\_name = piece\_name[0]

if group\_name == 'A':

pieces[0] -= 1

elif group\_name == 'B':

pieces[1] -= 1

elif group\_name == 'C':

pieces[2] -= 1

if piece\_name == 'A1':

board[0][0] = 0

elif piece\_name == 'A2':

board[0][1] = 0

elif piece\_name == 'A3':

board[0][2] = 0

elif piece\_name == 'A4':

board[0][3] = 0

elif piece\_name == 'A5':

board[0][4] = 0

elif piece\_name == 'A6':

board[0][5] = 0

elif piece\_name == 'A7':

board[0][6] = 0

elif piece\_name == 'B1':

board[1][0] = 0

elif piece\_name == 'B2':

board[1][1] = 0

elif piece\_name == 'B3':

board[1][2] = 0

elif piece\_name == 'B4':

board[1][3] = 0

elif piece\_name == 'B5':

board[1][4] = 0

elif piece\_name == 'C1':

board[2][0] = 0

elif piece\_name == 'C2':

board[2][1] = 0

elif piece\_name == 'C3':

board[2][2] = 0

def AI\_to\_play():

# this finds the computer's action using the strategies defined

next\_move = find\_next\_move()

print(next\_move)

canvas.delete("AI\_first")

# this applies the strategy and consist of the computer building a list of board moves it must make to apply the strategy has determined is best

if finish == False:

# delta is the difference is the current state and the future state of the board

# delta should only every have 1 number of the 3 that is greater than zero. eg. [0,3,0] tells the program to remove 3 pieced from pile B

delta = [pieces[0] - next\_move[0], pieces[1] - next\_move[1], pieces[2] - next\_move[2]]

pieces\_to\_take = []

if delta[0] > 0:

# take from A

piece\_index = 0

number\_of\_pieces\_to\_take = delta[0]

while number\_of\_pieces\_to\_take > 0:

if board[0][piece\_index] == 1:

board[0][piece\_index] = 0

pieces[0] -= 1

piece\_index += 1

number\_of\_pieces\_to\_take -= 1

pieces\_to\_take.append('A' + str(piece\_index))

else:

piece\_index += 1

elif delta[1] > 0:

# take from B

piece\_index = 0

number\_of\_pieces\_to\_take = delta[1]

while number\_of\_pieces\_to\_take > 0:

if board[1][piece\_index] == 1:

board[1][piece\_index] = 0

pieces[1] -= 1

piece\_index += 1

number\_of\_pieces\_to\_take -= 1

pieces\_to\_take.append('B' + str(piece\_index))

else:

piece\_index += 1

elif delta[2] > 0:

# take from C

piece\_index = 0

number\_of\_pieces\_to\_take = delta[2]

while number\_of\_pieces\_to\_take > 0:

if board[2][piece\_index] == 1:

board[2][piece\_index] = 0

pieces[2] -= 1

piece\_index += 1

number\_of\_pieces\_to\_take -= 1

pieces\_to\_take.append('C' + str(piece\_index))

else:

piece\_index += 1

# this is the computer actually making the moves iteratively for each move it has decided

if finish == False:

for piece in pieces\_to\_take:

canvas.itemconfig(piece, fill="yellow") # change the colour of the pieces the AI selects to yellow before deleting them

canvas.update\_idletasks()

canvas.after(500) #500ms delay to allow the user to see the pieces change colour before the AI deletes them

canvas.delete(piece) # delete each piece the AI selects

if sum(pieces) == 0: #check if the user has won

game\_over('user')

def game\_over(who\_won): # displays the final message of who won

button\_width = 190

button\_height = 40

BX = 260

BY = 110

canvas.delete('DONE')

canvas.create\_rectangle(BX, BY, BX + button\_width, BY + button\_height, outline="black", fill="grey80",

tags="WON\_BUTTON", command=None)

if who\_won == 'user':

canvas.create\_text(BX + 95, BY + 20, text="!!! YOU WON !!!", font="Purisa", tags="WON\_BUTTON",

fill="black", command=None)

elif who\_won == 'computer':

canvas.create\_text(BX + 95, BY + 20, text="THE COMPUTER WON", font="Purisa", tags="WON\_BUTTON",

fill="red", command=None)

# this is a very prescriptive strategy where each move has been typed explicitly.

# anywhere there is a choice (randint) is where the computer does not have a clear strategy and may try and trick the user into making a mistake

def find\_next\_move():

global finish, lvl

if lvl == 1:

if randint(0, 1) == 0:

next\_move = [1, 0, 0]

if lvl == 2:

if randint(0, 2) == 0:

next\_move = [1, 0, 0]

finish = False

next\_move = []

choice = randint(1,3)

choice1 = randint(1,2)

if pieces == [7, 5, 3]: #this is the opening move if you ask the AI to play first. It will pick one of two options

if choice1 < 2:

next\_move = [7, 4, 3]

else:

next\_move = [7, 5, 2]

elif pieces == [7, 5, 2]: #every other choice is a matter of 3 options

if choice == 1:

next\_move = [6, 5, 2]

elif choice == 2:

next\_move = [4, 5, 2]

else:

next\_move = [7, 4, 2]

elif pieces == [7, 5, 1]: # if there is a clear best move then the computer will always play that

next\_move = [4, 5, 1]

elif pieces == [7, 5, 0]:

next\_move = [5, 5, 0]

elif pieces == [7, 4, 3]:

if choice == 1:

next\_move = [6, 4, 3]

elif choice == 2:

next\_move = [7, 4, 2]

else:

next\_move = [7, 4, 1]

elif pieces == [7, 4, 2]:

next\_move = [6, 4, 2]

elif pieces == [7, 4, 1]:

next\_move = [5, 4, 1]

elif pieces == [7, 4, 0]:

next\_move = [4, 4, 0]

elif pieces == [7, 3, 3]:

next\_move = [0, 3, 3]

elif pieces == [7, 3, 2]:

next\_move = [1, 3, 2]

elif pieces == [7, 3, 1]:

next\_move = [2, 3, 1]

elif pieces == [7, 3, 0]:

next\_move = [3, 3, 0]

elif pieces == [7, 2, 3]:

next\_move = [1, 2, 3]

elif pieces == [7, 2, 2]:

next\_move = [2, 2, 2]

elif pieces == [7, 2, 1]:

next\_move = [3, 2, 1]

elif pieces == [7, 2, 0]:

next\_move = [2, 2, 0]

elif pieces == [7, 1, 3]:

next\_move = [2, 1, 3]

elif pieces == [7, 1, 2]:

next\_move = [3, 1, 2]

elif pieces == [7, 1, 1]:

next\_move = [1, 1, 1]

elif pieces == [7, 1, 0]:

next\_move = [0, 1, 0]

elif pieces == [7, 0, 3]:

next\_move = [3, 0, 3]

elif pieces == [7, 0, 2]:

next\_move = [2, 0, 2]

elif pieces == [7, 0, 1]:

next\_move = [0, 0, 1]

elif pieces == [7, 0, 0]:

next\_move = [1, 0, 0]

elif pieces == [6, 5, 3]:

if choice == 1:

next\_move = [6, 4, 3]

elif choice == 2:

next\_move = [6, 5, 2]

else:

next\_move = [4, 5, 3]

elif pieces == [6, 5, 2]:

next\_move = [6, 4, 2]

elif pieces == [6, 5, 1]:

next\_move = [4, 5, 1]

elif pieces == [6, 5, 0]:

next\_move = [5, 5, 0]

elif pieces == [6, 4, 3]:

next\_move = [6, 4, 2]

elif pieces == [6, 4, 2]:

if choice == 1:

next\_move = [6, 4, 1]

elif choice == 2:

next\_move = [5, 4, 2]

else:

next\_move = [3, 4, 2]

elif pieces == [6, 4, 1]:

next\_move = [5, 4, 1]

elif pieces == [6, 4, 0]:

next\_move = [4, 4, 0]

elif pieces == [6, 3, 3]:

next\_move = [0, 3, 3]

elif pieces == [6, 3, 2]:

next\_move = [1, 3, 2]

elif pieces == [6, 3, 1]:

next\_move = [2, 3, 1]

elif pieces == [6, 3, 0]:

next\_move = [3, 3, 0]

elif pieces == [6, 2, 3]:

next\_move = [1, 2, 3]

elif pieces == [6, 2, 2]:

next\_move = [0, 2, 2]

elif pieces == [6, 2, 1]:

next\_move = [3, 2, 1]

elif pieces == [6, 2, 0]:

next\_move = [2, 2, 0]

elif pieces == [6, 1, 3]:

next\_move = [2, 1, 3]

elif pieces == [6, 1, 2]:

next\_move = [3, 1, 2]

elif pieces == [6, 1, 1]:

next\_move = [1, 1, 1]

elif pieces == [6, 1, 0]:

next\_move = [0, 1, 0]

elif pieces == [6, 0, 3]:

next\_move = [3, 0, 3]

elif pieces == [6, 0, 2]:

next\_move = [2, 0, 2]

elif pieces == [6, 0, 1]:

next\_move = [0, 0, 1]

elif pieces == [6, 0, 0]:

next\_move = [1, 0, 0]

elif pieces == [5, 5, 3]:

next\_move = [5, 5, 0]

elif pieces == [5, 5, 2]:

next\_move = [5, 5, 0]

elif pieces == [5, 5, 1]:

next\_move = [5, 5, 0]

elif pieces == [5, 5, 0]:

if choice == 1:

next\_move = [5, 2, 0]

elif choice == 2:

next\_move = [3, 5, 0]

else:

next\_move = [2, 5, 0]

elif pieces == [5, 4, 3]:

next\_move = [5, 4, 1]

elif pieces == [5, 4, 2]:

next\_move = [5, 4, 1]

elif pieces == [5, 4, 1]:

if choice == 1:

next\_move = [5, 3, 1]

elif choice == 2:

next\_move = [3, 4, 1]

else:

next\_move = [4, 4, 1]

elif pieces == [5, 4, 0]:

next\_move = [4, 4, 0]

elif pieces == [5, 3, 3]:

next\_move = [0, 3, 3]

elif pieces == [5, 3, 2]:

next\_move = [1, 3, 2]

elif pieces == [5, 3, 1]:

next\_move = [2, 3, 1]

elif pieces == [5, 3, 0]:

next\_move = [3, 3, 0]

elif pieces == [5, 2, 3]:

next\_move = [1, 2, 3]

elif pieces == [5, 2, 2]:

next\_move = [0, 2, 2]

elif pieces == [5, 2, 1]:

next\_move = [3, 2, 1]

elif pieces == [5, 2, 0]:

next\_move = [2, 2, 0]

elif pieces == [5, 1, 3]:

next\_move = [2, 1, 3]

elif pieces == [5, 1, 2]:

next\_move = [3, 1, 2]

elif pieces == [5, 1, 1]:

next\_move = [1, 1, 1]

elif pieces == [5, 1, 0]:

next\_move = [0, 1, 0]

elif pieces == [5, 0, 3]:

next\_move = [3, 0, 3]

elif pieces == [5, 0, 2]:

next\_move = [2, 0, 2]

elif pieces == [5, 0, 1]:

next\_move = [0, 0, 1]

elif pieces == [5, 0, 0]:

next\_move = [1, 0, 0]

elif pieces == [4, 5, 3]:

next\_move = [4, 5, 1]

elif pieces == [4, 5, 2]:

next\_move = [4, 5, 1]

elif pieces == [4, 5, 1]:

if choice == 1:

next\_move = [3, 5, 1]

elif choice == 2:

next\_move = [2, 5, 1]

else:

next\_move = [4, 3, 1]

elif pieces == [4, 5, 0]:

next\_move = [4, 4, 0]

elif pieces == [4, 4, 3]:

next\_move = [4, 4, 0]

elif pieces == [4, 4, 2]:

next\_move = [4, 4, 0]

elif pieces == [4, 4, 1]:

next\_move = [4, 4, 0]

elif pieces == [4, 4, 0]:

if choice == 1:

next\_move = [4, 2, 0]

elif choice == 2:

next\_move = [3, 4, 0]

else:

next\_move = [4, 1, 0]

elif pieces == [4, 3, 3]:

next\_move = [0, 3, 3]

elif pieces == [4, 3, 2]:

next\_move = [1, 3, 2]

elif pieces == [4, 3, 1]:

next\_move = [2, 3, 1]

elif pieces == [4, 3, 0]:

next\_move = [3, 3, 0]

elif pieces == [4, 2, 3]:

next\_move = [1, 2, 3]

elif pieces == [4, 2, 2]:

next\_move = [0, 2, 2]

elif pieces == [4, 2, 1]:

next\_move = [3, 2, 1]

elif pieces == [4, 2, 0]:

next\_move = [2, 2, 0]

elif pieces == [4, 1, 3]:

next\_move = [2, 1, 3]

elif pieces == [4, 1, 2]:

next\_move = [3, 1, 2]

elif pieces == [4, 1, 1]:

next\_move = [1, 1, 1]

elif pieces == [4, 1, 0]:

next\_move = [0, 1, 0]

elif pieces == [4, 0, 3]:

next\_move = [3, 0, 3]

elif pieces == [4, 0, 2]:

next\_move = [2, 0, 2]

elif pieces == [4, 0, 1]:

next\_move = [0, 0, 1]

elif pieces == [4, 0, 0]:

next\_move = [1, 0, 0]

elif pieces == [3, 5, 3]:

next\_move = [3, 0, 3]

elif pieces == [3, 5, 2]:

next\_move = [3, 1, 2]

elif pieces == [3, 5, 1]:

next\_move = [3, 2, 1]

elif pieces == [3, 5, 0]:

next\_move = [3, 3, 0]

elif pieces == [3, 4, 3]:

next\_move = [3, 0, 3]

elif pieces == [3, 4, 2]:

next\_move = [3, 1, 2]

elif pieces == [3, 4, 1]:

next\_move = [3, 2, 1]

elif pieces == [3, 4, 0]:

next\_move = [3, 3, 0]

elif pieces == [3, 3, 3]:

next\_move = [3, 0, 3]

elif pieces == [3, 3, 2]:

next\_move = [3, 3, 0]

elif pieces == [3, 3, 1]:

next\_move = [3, 3, 0]

elif pieces == [3, 3, 0]:

if choice == 1:

next\_move = [3, 2, 0]

elif choice == 2:

next\_move = [1, 3, 0]

else:

next\_move = [3, 1, 0]

elif pieces == [3, 2, 3]:

next\_move = [3, 0, 3]

elif pieces == [3, 2, 2]:

next\_move = [0, 2, 2]

elif pieces == [3, 2, 1]:

if choice == 1:

next\_move = [1, 2, 1]

elif choice == 2:

next\_move = [3, 0, 1]

else:

next\_move = [2, 2, 1]

elif pieces == [3, 2, 0]:

next\_move = [2, 2, 0]

elif pieces == [3, 1, 3]:

next\_move = [3, 0, 3]

elif pieces == [3, 1, 2]:

if choice == 1:

next\_move = [1, 1, 2]

elif choice == 2:

next\_move = [3, 0, 2]

else:

next\_move = [2, 1, 2]

elif pieces == [3, 1, 1]:

next\_move = [1, 1, 1]

elif pieces == [3, 1, 0]:

next\_move = [0, 1, 0]

elif pieces == [3, 0, 3]:

if choice == 1:

next\_move = [2, 0, 3]

elif choice == 2:

next\_move = [3, 0, 2]

else:

next\_move = [1, 0, 3]

elif pieces == [3, 0, 2]:

next\_move = [2, 0, 2]

elif pieces == [3, 0, 1]:

next\_move = [0, 0, 1]

elif pieces == [3, 0, 0]:

next\_move = [1, 0, 0]

elif pieces == [2, 5, 3]:

next\_move = [2, 1, 3]

elif pieces == [2, 5, 2]:

next\_move = [2, 0, 2]

elif pieces == [2, 5, 1]:

next\_move = [2, 3, 1]

elif pieces == [2, 5, 0]:

next\_move = [2, 2, 0]

elif pieces == [2, 4, 3]:

next\_move = [2, 1, 3]

elif pieces == [2, 4, 2]:

next\_move = [2, 0, 2]

elif pieces == [2, 4, 1]:

next\_move = [2, 3, 1]

elif pieces == [2, 4, 0]:

next\_move = [2, 2, 0]

elif pieces == [2, 3, 3]:

next\_move = [0, 3, 3]

elif pieces == [2, 3, 2]:

next\_move = [2, 0, 2]

elif pieces == [2, 3, 1]:

if choice == 1:

next\_move = [2, 2, 1]

elif choice == 2:

next\_move = [0, 3, 1]

else:

next\_move = [2, 1, 1]

elif pieces == [2, 3, 0]:

next\_move = [2, 2, 0]

elif pieces == [2, 2, 3]:

next\_move = [2, 2, 0]

elif pieces == [2, 2, 2]:

next\_move = [2, 0, 2]

elif pieces == [2, 2, 1]:

next\_move = [2, 2, 0]

elif pieces == [2, 2, 0]:

if choice == 1:

next\_move = [2, 0, 0]

elif choice == 2:

next\_move = [1, 2, 0]

else:

next\_move = [2, 1, 0]

elif pieces == [2, 1, 3]:

if choice == 1:

next\_move = [2, 1, 2]

elif choice == 2:

next\_move = [2, 1, 1]

else:

next\_move = [1, 1, 3]

elif pieces == [2, 1, 2]:

next\_move = [2, 0, 2]

elif pieces == [2, 1, 1]:

next\_move = [1, 1, 1]

elif pieces == [2, 1, 0]:

next\_move = [0, 1, 0]

elif pieces == [2, 0, 3]:

next\_move = [2, 0, 2]

elif pieces == [2, 0, 2]:

if choice == 1:

next\_move = [2, 0, 1]

elif choice == 2:

next\_move = [1, 0, 2]

else:

next\_move = [2, 0, 0]

elif pieces == [2, 0, 1]:

next\_move = [0, 0, 1]

elif pieces == [2, 0, 0]:

next\_move = [1, 0, 0]

elif pieces == [1, 5, 3]:

next\_move = [1, 2, 3]

elif pieces == [1, 5, 2]:

next\_move = [1, 3, 2]

elif pieces == [1, 5, 1]:

next\_move = [1, 1, 1]

elif pieces == [1, 5, 0]:

next\_move = [1, 0, 0]

elif pieces == [1, 4, 3]:

next\_move = [1, 2, 3]

elif pieces == [1, 4, 2]:

next\_move = [1, 3, 2]

elif pieces == [1, 4, 1]:

next\_move = [1, 1, 1]

elif pieces == [1, 4, 0]:

next\_move = [1, 0, 0]

elif pieces == [1, 3, 3]:

next\_move = [0, 3, 3]

elif pieces == [1, 3, 2]:

if choice == 1:

next\_move = [1, 2, 2]

elif choice == 2:

next\_move = [1, 3, 1]

else:

next\_move = [1, 1, 2]

elif pieces == [1, 3, 1]:

next\_move = [1, 1, 1]

elif pieces == [1, 3, 0]:

next\_move = [1, 0, 0]

elif pieces == [1, 2, 3]:

if choice == 1:

next\_move = [1, 2, 2]

elif choice == 2:

next\_move = [1, 2, 1]

else:

next\_move = [1, 1, 3]

elif pieces == [1, 2, 2]:

next\_move = [0, 2, 2]

elif pieces == [1, 2, 1]:

next\_move = [1, 1, 1]

elif pieces == [1, 2, 0]:

next\_move = [1, 0, 0]

elif pieces == [1, 1, 3]:

next\_move = [1, 1, 1]

elif pieces == [1, 1, 2]:

next\_move = [1, 1, 1]

elif pieces == [1, 1, 1]:

if choice == 1:

next\_move = [1, 0, 1]

elif choice == 2:

next\_move = [1, 1, 0]

else:

next\_move = [0, 1, 1]

elif pieces == [1, 1, 0]:

next\_move = [1, 0, 0]

elif pieces == [1, 0, 3]:

next\_move = [1, 0, 0]

elif pieces == [1, 0, 2]:

next\_move = [1, 0, 0]

elif pieces == [1, 0, 1]:

next\_move = [1, 0, 0]

elif pieces == [1, 0, 0]:

next\_move = [0, 0, 0]

elif pieces == [0, 5, 3]:

next\_move = [0, 3, 3]

elif pieces == [0, 5, 2]:

next\_move = [0, 2, 2]

elif pieces == [0, 5, 1]:

next\_move = [0, 0, 1]

elif pieces == [0, 5, 0]:

next\_move = [0, 1, 0]

elif pieces == [0, 4, 3]:

next\_move = [0, 3, 3]

elif pieces == [0, 4, 2]:

next\_move = [0, 2, 2]

elif pieces == [0, 4, 1]:

next\_move = [0, 0, 1]

elif pieces == [0, 4, 0]:

next\_move = [0, 1, 0]

elif pieces == [0, 3, 3]:

if choice == 1:

next\_move = [0, 3, 2]

elif choice == 2:

next\_move = [0, 2, 3]

else:

next\_move = [0, 3, 1]

elif pieces == [0, 3, 2]:

next\_move = [0, 2, 2]

elif pieces == [0, 3, 1]:

next\_move = [0, 0, 1]

elif pieces == [0, 3, 0]:

next\_move = [0, 1, 0]

elif pieces == [0, 2, 3]:

next\_move = [0, 2, 1]

elif pieces == [0, 2, 2]:

if choice == 1:

next\_move = [0, 1, 2]

elif choice == 2:

next\_move = [0, 2, 1]

else:

next\_move = [0, 2, 0]

elif pieces == [0, 2, 1]:

next\_move = [0, 0, 1]

elif pieces == [0, 2, 0]:

next\_move = [0, 1, 0]

elif pieces == [0, 1, 3]:

next\_move = [0, 1, 0]

elif pieces == [0, 1, 2]:

next\_move = [0, 1, 0]

elif pieces == [0, 1, 1]:

next\_move = [0, 1, 0]

elif pieces == [0, 1, 0]:

next\_move = [0, 0, 0]

elif pieces == [0, 0, 3]:

next\_move = [0, 0, 1]

elif pieces == [0, 0, 2]:

next\_move = [0, 0, 1]

elif pieces == [0, 0, 1]:

next\_move = [0, 0, 0]

elif pieces == [0, 0, 0]:

finish = True

return next\_move

#this is the main program

root = Tk()

root.title('Nim')

canvas = Canvas(root, width=680, height=250)

canvas.pack()

create\_pieces()

create\_operation\_buttons()

root.mainloop()

### Примеры работы

На рисунках 3.1 и 3.2 показаны примеры работы программы.

Рисунок 3.1

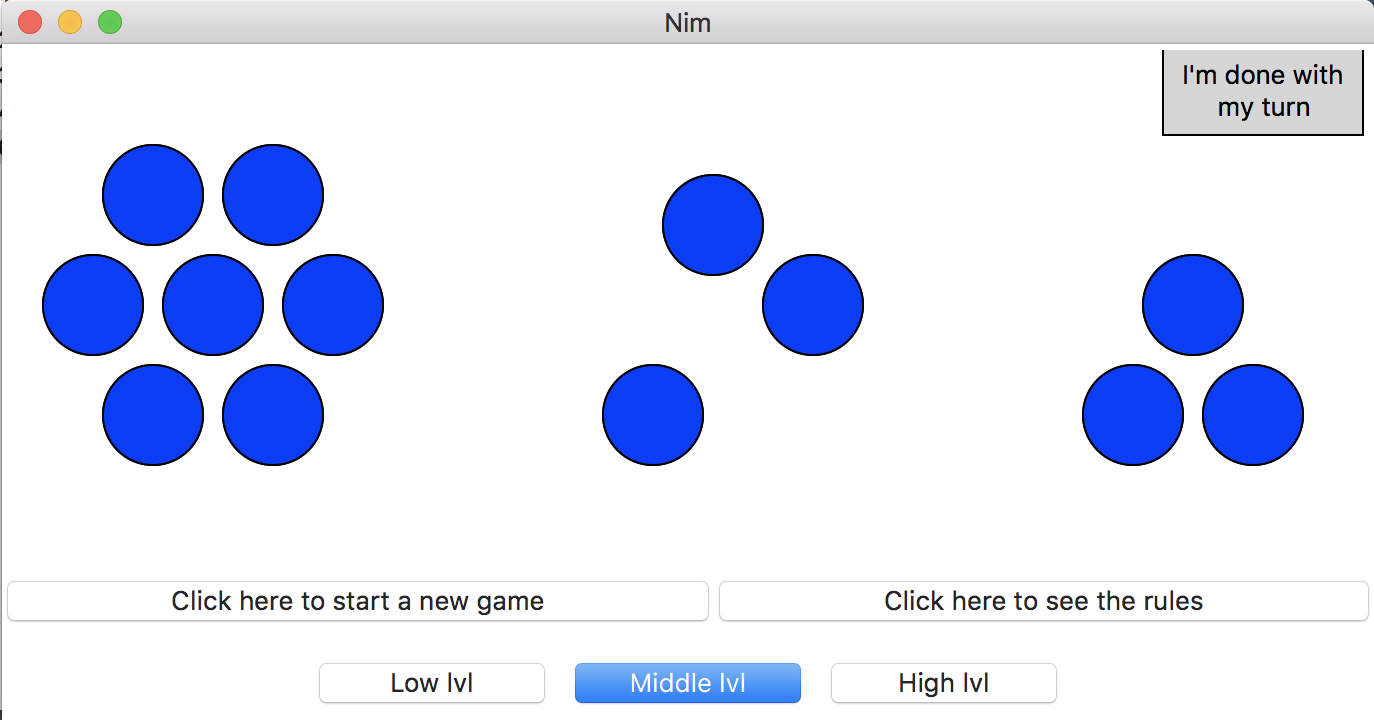
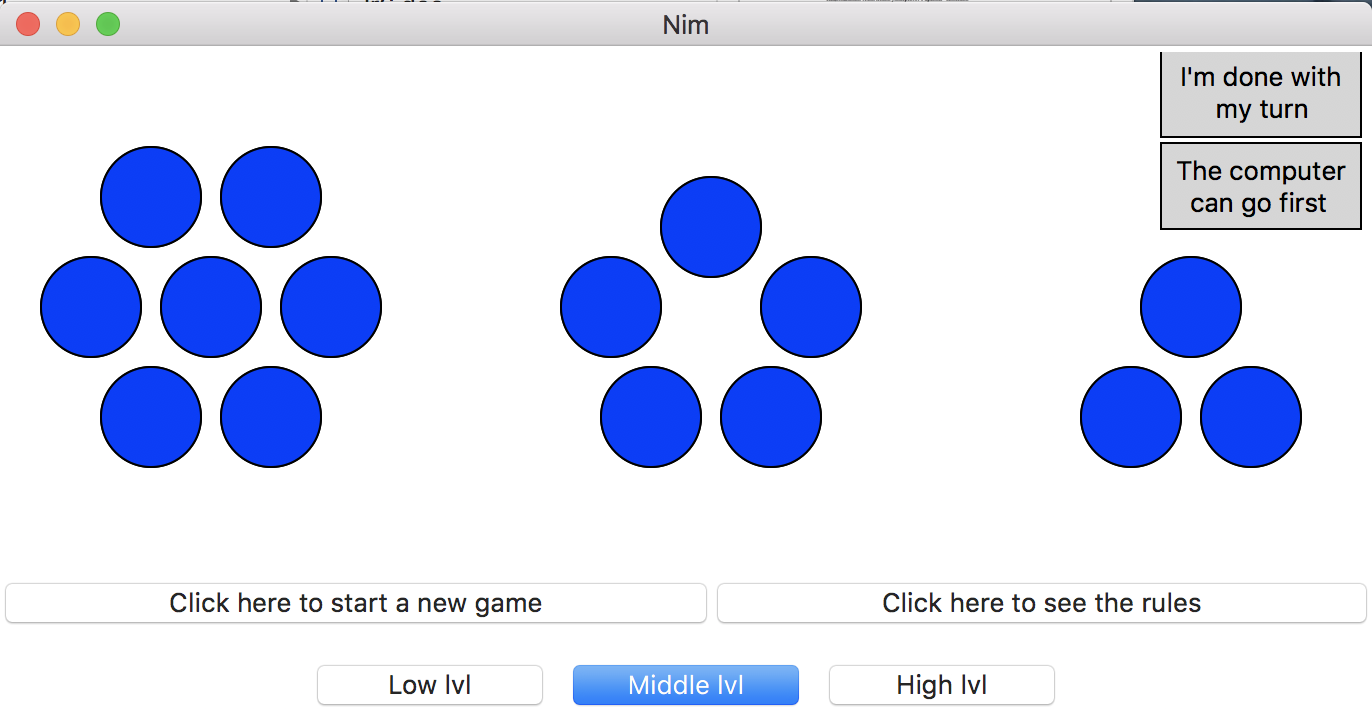


Рисунок 3.2



Выводы

В рамках данной лабораторной работы изучил математическую игру Ним и выполнил программную реализацию одной из его разновидностей.