МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа № 5

**Клеточные автоматы. Игра «Жизнь».**

Выполнил студент 595 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Лаптев

Проверил:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.Н. Уланов

Лабораторная работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2021

**Цель работы:** ознакомиться с понятием клеточного автомата, получить навыки создания клеточных автоматов, создать и протестировать клеточный автомат на примере игры «Жизнь».

**Задание.**

Реализовать программу, моделирующую игру «Жизнь».

Задать центральную часть поля случайной конфигурацией с размером случайной области от 3х3 до 10х10.

Многократными запусками программы найти начальные конфигурации, приводящие к:

• вымиранию,

• стабильной конфигурации,

• периодической конфигурации,

• конфигурации, развивающийся не менее 100 поколений.

Сохранить начальные конфигурации файл с возможностью задания затем из него начальной конфигурации моделирования для демонстрации правильности подбора.

**Решение.**

Модель игры была реализована согласно всем правилам игры на ЯПВУ Python.

Клеточный автомат задаётся следующими правилами.

Клетки на квадратной доске могут находиться в двух состояниях: «живое» и «мёртвое».

«Живая» клетка выживает на очередном временном шаге, если имеет только 2 или 3 живых соседа. Если соседей меньше двух, клетка умирает из-за обособленности, а если больше трех, то из-за скученности (перенаселения).

«Мёртвая» оживает на очередном шаге только в том случае, если имеет 3 живых соседа.

У каждой клетки 8 соседей: клетки, имеющие с ней общие стороны или вершины. Изменение состояния всех клеток происходит одновременно. В игре «Жизнь» встречаются самые разнообразные конфигурации «живых» клеток:

• конфигурации, которые вымирают за конечное число шагов;

• устойчивые или стационарные конфигурации, то есть конфигурации, которые в точности воспроизводятся на каждом временном шаге;

• периодически меняющиеся конфигурации, то есть те, которые претерпев ряд изменений, через несколько шагов возвращаются в исходное состояние, после чего процесс повторяется вновь;

• движущиеся конфигурации;

• генераторы – конфигурации, порождающие новые конфигурации.

Так как минимальный возможный размер поля 3х3, то разумнее всего для начальных конфигураций брать небольшое количество «живых» клеток. Проше всего рассчитать конфигурации для трёх клеток, поэтому такое количество и было взято за основу, при этом, для некоторых конфигураций начальное количество «живых» клеток взято другое.

Стоит также отметить, что для начальных условий с тремя «живыми» клетками не существует варианта решения для конфигурации, развивающейся не менее 100 поколений, которая не представляла бы собой стабильную либо периодическую конфигурацию. В связи с этим, для подобной конфигурации было выбрано начальное условие, в котором использованы 10 «живых» клеток, т.к. только для поля с максимальными размерами удалось подобрать количество и расположение «живых» клеток, развивающихся не менее 100 поколений.

Исходя из этих рассуждений, были выделены начальные условия для каждой из начальных конфигураций, для которых надо было найти решения.

Начальные положения для «живых» клеток (обозначены символом «#»):

Вымирание за конечное количество шагов:

field[1][0] = '#'

field[2][1] = '#'

field[2][2] = '#'

. . . . . . . . . .

# . . . . . . . . .

. # # . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

Стабильная конфигурация (данный вариант использует 5 «живых» клеток, хотя можно обойтись и тремя:

field[1][0] = '#'

field[2][1] = '#'

field[2][2] = '#'

field[0][0] = '#'

field[1][2] = '#'

# . . . . . . . . .

# . # . . . . . . .

. # # . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

Периодическая конфигурация:

field[1][0] = '#'

field[1][1] = '#'

field[1][2] = '#'

. . . . . .

# # # . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Конфигурация, развивающаяся не менее 100 поколений:

field[3][4] = '#'

field[4][5] = '#'

field[2][5] = '#'

field[2][4] = '#'

field[2][3] = '#'

field[5][4] = '#'

field[5][6] = '#'

field[6][4] = '#'

field[6][5] = '#'

field[6][6] = '#'

. . . . . . . . . .

. . . . . . . . . .

. . . # # # . . . .

. . . . # . . . . .

. . . . . # . . . .

. . . . # . # . . .

. . . . # # # . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

Приведённые параметры можно самостоятельно задать в приведённом коде, начальные значения также присутствуют в текстовом файле.

**Приложение**

import math

import random

vertical = random.randint(2, 9)

horizontal = vertical

n = vertical

m = horizontal

quantitySteps = 100

mode = 1

field = [[0 for x in range(vertical + 1)] for y in range(horizontal + 1)]

newField = [[0 for x in range(vertical + 1)] for y in range(horizontal + 1)]

def show():

for i in range(vertical + 1):

for j in range(horizontal + 1):

print(field[i][j], end = ' ')

print()

def generate():

help1 = math.floor((vertical - n) / 2)

help2 = math.floor((horizontal - m) / 2)

a = round(help1)

b = round(help2)

for i in range(a, (a + n + 1)):

for j in range(b, (b + m + 1)):

if (field[i][j] == random.random() % 2):

field[i][j] = '#'

else:

field[i][j] = '.'

def exchange():

for i in range (vertical + 1):

for j in range (horizontal + 1):

field[i][j] = newField[i][j]

newField[i][j] = '.'

for i in range (vertical + 1):

for j in range (horizontal + 1):

field[i][j] = '.'

newField[i][j] = '.'

if (mode):

field[3][4] = '#'

field[4][5] = '#'

field[2][5] = '#'

field[2][4] = '#'

field[2][3] = '#'

field[5][4] = '#'

field[5][6] = '#'

field[6][4] = '#'

field[6][5] = '#'

field[6][6] = '#'

else:

generate()

for t in range (quantitySteps):

step = input('next step press whitespace')

show()

if (t != quantitySteps - 1):

for i in range (vertical + 1):

for j in range (horizontal + 1):

counter = 0

if ((i - 1) != (-1) and (j - 1) != (-1) and field[i - 1][j - 1] == '#'):

counter += 1

if ((i - 1) != (-1) and field[i - 1][j] == '#'):

counter += 1

if ((i - 1) != (-1) and (j + 1) <= horizontal and field[i - 1][j + 1] == '#'):

counter += 1

if ((j - 1) != (-1) and field[i][j - 1] == '#'):

counter += 1

if ((j + 1) <= horizontal and field[i][j + 1] == '#'):

counter += 1

if ((i + 1) <= vertical and (j - 1) != (-1) and field[i + 1][j - 1] == '#'):

counter += 1

if ((i + 1) <= vertical and field[i + 1][j] == '#'):

counter += 1

if ((i + 1) <= vertical and (j + 1) <= horizontal and field[i + 1][j + 1] == '#'):

counter += 1

if (field[i][j] == '#' and (counter == 2 or counter == 3)):

newField[i][j] = '#'

if ((field[i][j] == '.') and (counter == 3)):

newField[i][j] = '#'

if (field[i][j] == '.' and counter != 3):

newField[i][j] = '.'

counter = 0

exchange()