**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №3**

**Дисциплина: Теория надежности**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: доц. каф. ПО и ВТАС

Кабалянц Петр Степанович

**Белгород 2020**

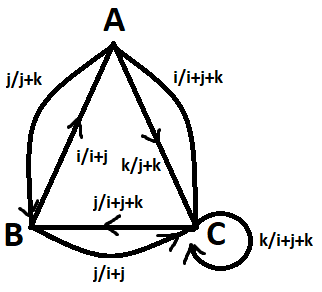
**Задания для выполнения к работе**

Имеется треугольник с вершинами A, B, C. В начальный момент времени муха находится в вершине A. Каждую секунду муха перелетает в другую вершину или остается на месте. Ее поведение задается графом марковского процесса:

1) Необходимо определить среднее время, через которое муха вернётся в вершину А.

2) Написать программу, которая имитирует поведение мухи и выводит среднее количество переходов до первого возвращения в точку А.

3) Сравните результаты.



**Ход выполнения работы**

Составим матрицу переходных вероятностей:

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| A | 0 | 10/11 | 1/11 |
| B | 3/13 | 0 | 10/13 |
| C | 3/14 | 10/14 | 1/14 |

Решим уравнение , преобразовав его к виду . В результате получим:

t = [0,421 0,137 0,442]

Для того, чтобы получить среднее время возврата в начальную позицию , необходимо найти обратную величину от :

= [2,375 7,308 2,262], т. е. среднее время возвращения в точку А, если начали в ней же, равняется 2.375 ед. времени.

Смоделируем полет мухи со стартовой позицией в точке А, и повторим его 1000 раз.

В результате моделирования было установлено, что муха возвращалась через 2,365 шагов, что на 0.009 больше, чем значение, вычисленное теоретически.

Изображение выглядит как снимок экрана, компьютер

Автоматически созданное описание

*Приложение*

# This Python file uses the following encoding: utf-8

import random

import math

import numpy as np

i = 3

j = 10

k = 1

# из / в

aa = 0

ab = k / (i + k)

ac = i / (i + k)

ba = j / (i + j)

bb = 0

bc = i / (i + j)

ca = j / (i + j + k)

cb = k / (i + j + k)

cc = i / (i + j + k)

# A B C

P = [[aa, ab, ac],

[ba, bb, bc],

[ca, cb, cc]]

print("Р = ", P)

tP = np.transpose(np.copy(P))

tP[np.diag\_indices\_from(tP)] -= 1

tP[-1] = [1, 1, 1]

res = np.array([0, 0, 1])

x = np.linalg.solve(tP, res)

print("t = ", x)

theorJumps = 1 / x

print("Среднее время возврата = ", theorJumps)

def jump(currentState):

r = random.random()

if r < P[currentState][0]:

#print(r, 0)

return 0

if r < P[currentState][0] + P[currentState][1]:

#print(r, 1)

return 1

else:

#print(r, 2)

return 2

firstState = 0

sz = 1000

result = []

print("Проведем эксперимент: заставим муху {} раз попрыгать по треугольнику".format(sz))

for i in range (0, sz):

state = firstState

jumps = 1

state = jump(state)

while state != firstState:

state = jump(state)

jumps += 1

result.append(jumps)

returnTime = sum(result) / len(result)

print("На вершину А муха возвращалась через {} шагов".format(returnTime))

print("Согласно теории, муха должна была возвращаться через {} шагов. Разница между этими величинами составила {}".format(round(theorJumps[0], 4), abs(returnTime - theorJumps[0])))