**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 №6**

**Дисциплина: Теория надежности**

**Тема: Анализ надежности сложной резервируемой системы**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: доц. каф. ПО и ВТАС

Кабалянц Петр Степанович

**Белгород 2020**

**Задания для выполнения к работе**

1. Построить структурную схему системы.

2. Выполнить анализ системы методом, основанным на теоремах теории вероятности.

3. Выполнить анализ системы методом минимальных путей и минимальных сечений.

4. Составить для системы функцию алгебры логики.

5. Применить алгоритм разрезания для вычисления вероятности безотказной работы системы.

6. Составить программу, моделирующую работу системы при разном режиме обслуживания и случайном времени отказов и восстановлений.

**Ход выполнения работы**

Вариант 9

Система состоит из 3 узлов: элемент 1 резервирован с кратностью m=3, а элементы 2 и 3 с кратностью m=3/2.

Изображение выглядит как счетчик

Автоматически созданное описание

Вероятность отказа узла равна произведению вероятности отказа основного и дублирующего элемента, тогда вероятность безотказный работы узла

Вероятность безотказной работы системы равна произведению этих вероятностей:

,

а вероятность отказа:

.

Минимальные пути:

1-5-6, 1-7-8, 1-9, 2-5-6, 2-7-8, 2-9, 3-5-6, 3-7-8, 3-9, 4-5-6, 4-7-8, 4-9;

Минимальные сечения:

1-2-3-4, 5-7-9, 5-8-9, 6-7-9, 6-8-9.

Функция работоспособности:

Функция неработоспособности:

Изображение выглядит как снимок экрана, монитор, внутренний, компьютер

Автоматически созданное описание

*Приложение*

import numpy

import random

from itertools import combinations

n = 9 # количество элементов

P = []

Q = []

for i in range(0, 9):

tmp = random.random() \* 0.1

P.append(1 - tmp)

Q.append(tmp)

P.append(1 - tmp)

Q.append(tmp)

P = [0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4]

Q = [0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6]

print("Вероятности безотказной работы элементов P = {}\nВероятности отказов Q = {}\n".format(P, Q))

min\_ways = [{1, 5, 6}, {1, 7, 8}, {1,9}, {2, 5, 6}, {2, 7, 8}, {2,9},

{3, 5, 6}, {3, 7, 8}, {3,9}, {4, 5, 6}, {4, 7, 8}, {4,9},]

min\_cuts = [{1, 2, 3, 4}, {5, 7, 9}, {5, 8, 9}, {6, 7, 9}, {6, 8, 9}]

P\_tv = (1 - Q[0]\*Q[1]\*Q[2]\*Q[3])\*(1 - ((1 - P[4]\*P[5])\*(1 - P[6]\*P[7]))\*Q[8])

Q\_tv = 1 - P\_tv

print("Анализ системы на основе теорем теории вероятности: P = (1-q0q1)(1-q2q3)(1-q4q5) = {}; Q = 1 - P = {}\n".format(P\_tv, Q\_tv))

print("Анализ системы методом минимальных путей и сечений:")

print("Минимальные пути: {}\nМинимальные сечения: {}\n".format(min\_ways, min\_cuts))

def getStrExpression(combinations, ch, sign):

res\_str = ""

for comb in combinations:

res\_str += " " + sign + " "

for j in range(n):

if j in comb:

res\_str += "{}{}".format(ch, j)

return res\_str

def uniCompinations(combinations, k, set\_):

res = []

for comb in combinations:

t = set()

for j in range(len(set\_)):

if j in comb:

t = t | set\_[j]

res.append(t)

return res

def computeCombinations(combinations, p\_q):

res = 0

for comb in combinations:

t = 1

for j in range(n):

if j in comb:

t \*= p\_q[j]

res += t

return res

ways\_num = [i for i in range(0, len(min\_ways))]

exp\_str = ""

exp\_res = 0

for k in range(1, len(min\_ways) + 1):

comb = list(combinations(ways\_num, k))

comb = uniCompinations(comb, k, min\_ways)

if (k % 2 != 0):

exp\_res += computeCombinations(comb, P)

else:

exp\_res -= computeCombinations(comb, P)

str\_ = getStrExpression(comb, "p", "+" if (k % 2 != 0) else "-")

exp\_str += str\_

exp\_str = "P =" + exp\_str[2:]

#print("Вероятность безотказной работы методом минимальных путей и сечений: {}".format(exp\_str))

#print("P = {}\n".format(exp\_res))

cuts\_num = [i for i in range(0, len(min\_cuts))]

exp\_str = ""

exp\_res = 0

for k in range(1, len(min\_cuts) + 1):

comb = list(combinations(cuts\_num, k))

comb = uniCompinations(comb, k, min\_cuts)

if (k % 2 != 0):

exp\_res += computeCombinations(comb, Q)

else:

exp\_res -= computeCombinations(comb, Q)

str\_ = getStrExpression(comb, "q", "+" if (k % 2 != 0) else "-")

exp\_str += str\_

exp\_str = "Q =" + exp\_str[2:]

print("Вероятность отказа методом минимальных путей и сечений: {}".format(exp\_str))

print("Q = {}\n".format(exp\_res))

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

def schemFunc(X):

res = X[0] and X[2] and X[4] or X[0] and X[2] and X[5] or X[0] and X[3] and X[4] or X[0] and X[3] and X[5]

res = res or X[1] and X[2] and X[4] or X[1] and X[2] and X[5] or X[1] and X[3] and X[4] or X[1] and X[3] and X[5]

return res

def step():

res = []

for i in range (n):

res.append(False if random.random() > P[i] else True)

return res

for k in range(1):

if (i != 0):

P = []

Q = []

for i in range(0, int(n/2)):

tmp = random.random() \* 0.1

P.append(1 - tmp)

Q.append(tmp)

P.append(1 - tmp)

Q.append(tmp)

P\_tv = (1 - Q[0]\*\*2)\*(1 - Q[2]\*\*2)\*(1-Q[4]\*\*2)

Q\_tv = 1 - P\_tv

print("Теоретически: P = {}; Q = {}".format(P\_tv, Q\_tv))

N = 1000

ok = 0

for i in range(N):

X = step()

if (schemFunc(X)):

ok += 1

print("Вероятность безотказной работы в результате эксперимента: ", ok/N)

print("Вероятность отказа в результате эксперимента: ", (N-ok)/N, "\n")