**Задания к работе**

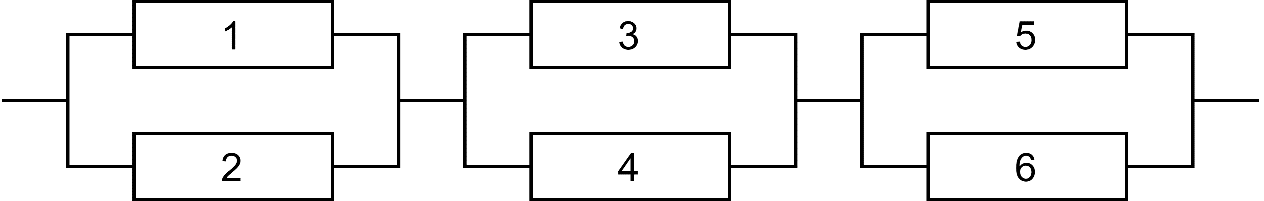
1. Построить структурную схему системы;
2. Выполнить анализ системы методом, основанным на теоремах теории вероятности;
3. Выполнить анализ системы методом минимальных путей и минимальных сечений;
4. Составить для системы функцию алгебры логики;
5. Применить алгоритм разрезания для вычисления вероятности безотказной работы системы;
6. Составить программу, моделирующую работу системы при разном режиме обслуживания и случайном времени отказов и восстановлений.

**Задание варианта №4**

Система состоит из 3 узлов, каждый из которых резервируется дублирующим элементом.

**Выполнение**

1. 1.



1. 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: .

1. Минимальные пути: 1-3-5, 1-3-6, 1-4-5, 1-4-6, 2-3-5, 2-3-6, 2-4-5, 2-4-6;

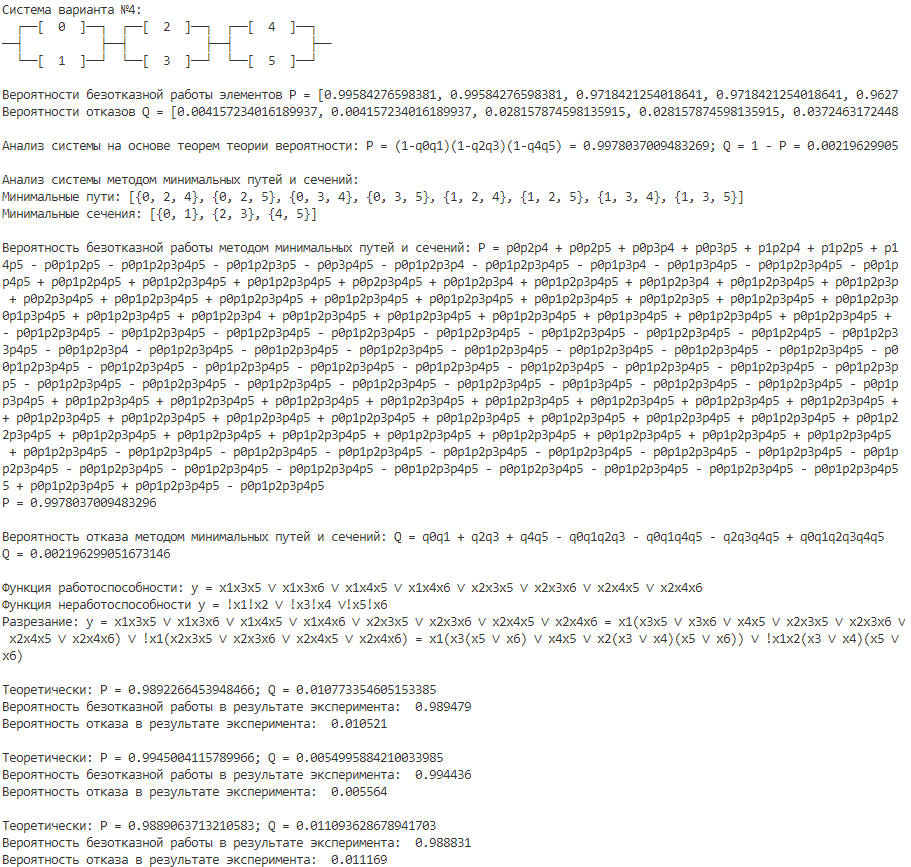
Минимальные сечения: 1-2, 3-4, 5-6.

1. Функция работоспособности:

Функция неработоспособности:

1. Разрезание:

Пример работы программы



Вывод: в ходе лабораторной работы была построена структурная схема системы, проведён её анализ методами теории вероятности, методом минимальных путей и сечений, для системы была составлена функция алгебры логики и к этой функции был применён алгоритм разрезания. Для полученной системы была написана программа, выполняющая анализ системы и моделирующая работу системы для трёх разных режимов. Результаты эксперимента совпадают с теоретическими данными с точностью до десятитысячных долей единицы.

Код программы

import numpy

import random

from itertools import combinations

print('''Система варианта №4:

  ┌──[  0  ]──┐  ┌──[  2  ]──┐  ┌──[  4  ]──┐

──┤           ├──┤           ├──┤           ├──

  └──[  1  ]──┘  └──[  3  ]──┘  └──[  5  ]──┘

''')

n = 6   # количество элементов

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)

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

min\_ways = [{0, 2, 4}, {0, 2, 5}, {0, 3, 4}, {0, 3, 5}, {1, 2, 4}, {1, 2, 5}, {1, 3, 4}, {1, 3, 5}]

min\_cuts = [{0, 1}, {2, 3}, {4, 5}]

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

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))

print("Функция работоспособности: y = x1x3x5 ∨ x1x3x6 ∨ x1x4x5 ∨ x1x4x6 ∨ x2x3x5 ∨ x2x3x6 ∨ x2x4x5 ∨ x2x4x6")

print("Функция неработоспособности y = !x1!x2 ∨ !x3!x4 ∨!x5!x6")

print("Разрезание: y = x1x3x5 ∨ x1x3x6 ∨ x1x4x5 ∨ x1x4x6 ∨ x2x3x5 ∨ x2x3x6 ∨ x2x4x5 ∨ x2x4x6 = x1(x3x5 ∨ x3x6 ∨ x4x5 ∨ x2x3x5 ∨ x2x3x6 ∨ x2x4x5 ∨ x2x4x6) ∨ !x1(x2x3x5 ∨ x2x3x6 ∨ x2x4x5 ∨ x2x4x6) = x1(x3(x5 ∨ x6) ∨ x4x5 ∨ x2(x3 ∨ x4)(x5 ∨ x6)) ∨ !x1x2(x3 ∨ x4)(x5 ∨ x6)\n")

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

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(3):

    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 = 1000000

    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")