**Условие задачи**

Вычислительная система состоит из трех ЭВМ. С заданным интервалом мин. в систему поступают задания, которые с вероятносятми: идут на первую ЭВМ, с адресуются второй ЭВМ, а все остальные идут на обработку на третью ЭВМ. Перед каждой ЭВМ имеется очередь заданий, длина которой не ограничена. После обработки задания на первой ЭВМ оно с вероятностью поступает в очередь ко второй ЭВМ и с вероятностью – в очередь к третье ЭВМ. После обработки на второй или третьей ЭВМ задание считается выполненным.

Продолжительность обработки заданий на разных ЭВМ характеризуется интервалами времени: , мин. , мин. мин.

Разработать программу, моделирующую процесс функционирования вычислительной системы массового обслуживания с неограниченной очередью.

**Схема, описывающая работу программы**

Изображение выглядит как диаграмма, зарисовка, рисунок, Технический чертеж

Автоматически созданное описание

Рисунок 1 – блок-схема программы

**Результаты работы программы**

Пример 1. Контрольные значения

Итоги работы ЭВМ 1:

Количество поступивших заданий: 81

Общее время работы: 6 ч. 7 мин.

Процент простоя: 39.14 %

Очередь: 0

Среднее время ожидания в очереди: 2 мин.

Итоги работы ЭВМ 2:

Количество поступивших заданий: 79

Общее время работы: 4 ч. 45 мин.

Процент простоя: 52.74 %

Очередь: 0

Среднее время ожидания в очереди: 1 мин.

Итоги работы ЭВМ 3:

Количество поступивших заданий: 121

Общее время работы: 9 ч. 50 мин.

Процент простоя: 2.16 %

Очередь: 5

Среднее время ожидания в очереди: 9 мин.

Общие итоги работы системы:

Общее время работы системы: 10 ч. 3 мин.

Среднее время выпонения задачи: 4.34 мин.

Среднее число каналов под обслуживанием: 2.06

Средний процент простоя: 31.35 %

Абсолютная пропускная способность: 0.78

Среднее время ожидания заявки в очереди 4.0 мин.

Среднее время пребывания заявки в системе: 8.34 мин.

Краткие итоги:

- Общее время работы – примерно 10 часов.

- ЭВМ 1 и ЭВМ2 значительную часть времени находятся в простое.

- Среднее число каналов под обслуживанием – 2

Некоторые рекомендации по улучшению работы программы:

1 – Уменьшение времени поступления задачи в систему.

2 – Увеличить вероятность поступления задач на ЭВМ1 и ЭВМ2.

3 – Увеличить вероятность поступления задачи из ЭВМ1 в ЭВМ3.

Пример 2. Данные, введенные с клавиатуры с учетом рекомендаций

Количество заданий: 200

Интервалы поступления заданий в систему: 2 0

Вероятности попадания задания на 1, 2 и 3 ЭВМ: 0.4 0.4 0.2

Вероятности попадания из ЭВМ 1 в ЭВМ 2 и ЭВМ 3: 0.5 0.5

Время выполнения задания на ЭВМ 1: 4 1

Время выполнения задания на ЭВМ 2: 3 1

Время выполнения задания на ЭВМ 3: 3 2

Итоги работы ЭВМ 1:

Количество поступивших заданий: 81

Общее время работы: 5 ч. 49 мин.

Процент простоя: 14.67 %

Очередь: 2

Среднее время ожидания в очереди: 6 мин.

Итоги работы ЭВМ 2:

Количество поступивших заданий: 114

Общее время работы: 6 ч. 7 мин.

Процент простоя: 10.27 %

Очередь: 3

Среднее время ожидания в очереди: 4 мин.

Итоги работы ЭВМ 3:

Количество поступивших заданий: 86

Общее время работы: 4 ч. 40 мин.

Процент простоя: 31.54 %

Очередь: 0

Среднее время ожидания в очереди: 3 мин.

Общие итоги работы системы:

Общее время работы системы: 6 ч. 49 мин.

Среднее время выпонения задачи: 3.59 мин.

Среднее число каналов под обслуживанием: 2.44

Средний процент простоя: 18.83 %

Абсолютная пропускная способность: 0.92

Среднее время ожидания заявки в очереди 4.33 мин.

Среднее время пребывания заявки в системе: 7.93 мин.

Краткие итоги:

- Общее время работы – уменьшилось до 7 часов.

- Все машины большую часть времени работают.

- Среднее число каналов под обслуживанием – 2,52

**Вывод**

В ходе работы была разработана программа. моделирующая процесс функционирования вычислительной системы массового обслуживания с неограниченной очередью.

**Код программы**

import sys

import random

from random import randint

num = int(input("Выберите действие: \n1. Ввести значения с клавиатуры. \n2. Запустить контрольные значения. \n"))

if num == 1:

kol\_tasks = int(input('Количество заданий: '))

b0, a0 = map(int, input('Интервалы поступления заданий в систему: ').split())

p1, p2, p3 = map(float, input('Вероятности попадания задания на 1, 2 и 3 ЭВМ: ').split())

p\_evm\_1\_to\_2, p\_evm\_1\_to\_3 = map(float, input('Вероятности попадания из ЭВМ 1 в ЭВМ 2 и ЭВМ 3: ').split())

b1, a1 = map(int, input('Время выполнения задания на ЭВМ 1: ').split())

b2, a2 = map(int, input('Время выполнения задания на ЭВМ 2: ').split())

b3, a3 = map(int, input('Время выполнения задания на ЭВМ 3: ').split())

elif num == 2:

kol\_tasks = 200

p1 = 0.4

p2 = 0.3

p3 = 0.3

p\_evm\_1\_to\_2 = 0.3

p\_evm\_1\_to\_3 = 0.7

a0, b0 = 1, 3

a1, b1 = 1, 4

a2, b2 = 1, 3

a3, b3 = 2, 5

else:

print("Вы неправы!")

sys.exit()

def add\_task(task, now\_time, min\_time, max\_time):

time\_work = randint(min\_time, max\_time)

if len(task) == 0:

task.append([time\_work, now\_time])

else:

task.append([time\_work, now\_time])

return task

def del\_task(task, now\_time):

task.pop(0)

if len(task) == 0:

return task

else:

task[0] = [task[0][0], now\_time]

return task

def task\_wait\_time(task):

res = [0, 0, 0]

if(task[0][0] != 0):

res[0] = int(task[0][0] / task[0][1])

if (task[1][0] != 0):

res[1] = int(task[1][0] / task[1][1])

if (task[2][0] != 0):

res[2] = int(task[2][0] / task[2][1])

return res

def machine():

#время работы

next\_task\_min, next\_task\_max = b0 - a0, b0 + a0

evm\_1\_min, evm1\_max = b1 - a1, b1 + a1

evm\_2\_min, evm\_2\_max = b2 - a2, b2 + a2

evm\_3\_min, evm\_3\_max = b3 - a3, b3 + a3

# счётчики

now\_time = 0

new\_task\_time = 0

last\_task\_time = 0

evm\_1 = []

evm\_2 = []

evm\_3 = []

evm\_1\_task\_complete = 0

evm\_2\_task\_complete = 0

evm\_3\_task\_complete = 0

queue = [0, 0, 0]

time\_task\_waiting = [[0, 0], [0, 0], [0, 0]]

task\_number = 0

completed\_tasks = 0

while completed\_tasks < kol\_tasks:

if now\_time - last\_task\_time == new\_task\_time:

task\_number += 1

last\_task\_time = now\_time

new\_task\_time = randint(next\_task\_min, next\_task\_max)

new\_task = random.choices((1, 2, 3), weights=[p1, p2, p3])

if new\_task == [1]:

evm\_1 = add\_task(evm\_1, now\_time, evm\_1\_min, evm1\_max)

elif new\_task == [2]:

evm\_2 = add\_task(evm\_2, now\_time, evm\_2\_min, evm\_2\_max)

else:

evm\_3 = add\_task(evm\_3, now\_time, evm\_3\_min, evm\_3\_max)

# Задача поступает на ЭВМ 1

if len(evm\_1) != 0:

if now\_time - evm\_1[0][1] == evm\_1[0][0]:

if len(evm\_1) > 1:

time\_task\_waiting[0][0] += now\_time - evm\_1[1][1]

time\_task\_waiting[0][1] += 1

evm\_1 = del\_task(evm\_1, now\_time)

evm\_1\_task\_complete += 1

# Передаем задачу на следующую ЭВМ

rand = random.choices((1, 2), weights=[p\_evm\_1\_to\_2, p\_evm\_1\_to\_3])

if(rand == [1]):

evm\_2 = add\_task(evm\_2, now\_time, evm\_2\_min, evm\_2\_max)

else:

evm\_3 = add\_task(evm\_3, now\_time, evm\_3\_min, evm\_3\_max)

else:

queue[0] += 1

# задача поступила на ЭВМ 2

if len(evm\_2) != 0:

if len(evm\_2) > 1:

time\_task\_waiting[1][0] += now\_time - evm\_2[1][1]

time\_task\_waiting[1][1] += 1

if now\_time - evm\_2[0][1] == evm\_2[0][0]:

evm\_2 = del\_task(evm\_2, now\_time)

evm\_2\_task\_complete +=1

completed\_tasks += 1

else:

queue[1] += 1

# задача поступила на ЭВМ 3

if len(evm\_3) != 0:

if now\_time - evm\_3[0][1] == evm\_3[0][0]:

if len(evm\_3) > 1:

time\_task\_waiting[2][0] += now\_time - evm\_3[1][1]

time\_task\_waiting[2][1] += 1

evm\_3 = del\_task(evm\_3, now\_time)

evm\_3\_task\_complete += 1

completed\_tasks += 1

else:

queue[2] += 1

now\_time += 1

avg\_task\_wait = task\_wait\_time(time\_task\_waiting)

# task\_compl\_time = [(now\_time - time\_no\_queue[0]) / evm\_1\_task\_complete, (now\_time - time\_no\_queue[1]) / evm\_2\_task\_complete, (now\_time - time\_no\_queue[2]) / evm\_3\_task\_complete]

return [now\_time, len(evm\_1), len(evm\_2), len(evm\_3), now\_time - queue[0], now\_time - queue[1], now\_time - queue[2], evm\_1\_task\_complete, evm\_2\_task\_complete, evm\_3\_task\_complete,queue[0], queue[1], queue[2], avg\_task\_wait[0], avg\_task\_wait[1], avg\_task\_wait[2]]#task\_compl\_time[0] + (task\_compl\_time[1] \* p\_evm\_1\_to\_2 + task\_compl\_time[2] \* p\_evm\_1\_to\_3), task\_compl\_time[1], task\_compl\_time[2]]

a = 0

data = machine()

queue1, queue2, queue3 = data[1], data[2], data[3]

time\_in\_queue1, time\_in\_queue2, time\_in\_queue3, = data[13], data[14], data[15]

stop\_perc\_1 = round(100 \* data[10] / data[0], 2)

stop\_perc\_2 = round(100 \* data[11] / data[0], 2)

stop\_perc\_3 = round(100 \* data[12] / data[0], 2)

stop\_perc = round((stop\_perc\_1 + stop\_perc\_2 + stop\_perc\_3) / 3, 2)

A = 1 / b1 + 1 / b2 + 1 / b3

avg\_work\_time = (data[4] / data[7] + data[5] / data[8] + data[6] / data[9]) / 3

time\_in\_system = avg\_work\_time + (time\_in\_queue1 + time\_in\_queue2 + time\_in\_queue3) / 3

print('\nИтоги работы ЭВМ 1:')

print('Количество поступивших заданий:', data[7])

print('Общее время работы:', data[4] // 60, 'ч.', data[4] % 60, 'мин.')

print('Процент простоя:', stop\_perc\_1, '%')

print('Очередь:', queue1)

print('Среднее время ожидания в очереди:', time\_in\_queue1, 'мин.')

print('\nИтоги работы ЭВМ 2:')

print('Количество поступивших заданий:', data[8])

print('Общее время работы:', data[5] // 60, 'ч.', data[5] % 60, 'мин.')

print('Процент простоя:', stop\_perc\_2, '%')

print('Очередь:', queue2)

print('Среднее время ожидания в очереди:', time\_in\_queue2, 'мин.')

print('\nИтоги работы ЭВМ 3:')

print('Количество поступивших заданий:', data[9])

print('Общее время работы:', data[6] // 60, 'ч.', data[6] % 60, 'мин.')

print('Процент простоя:', stop\_perc\_3, '%')

print('Очередь:', queue3)

print('Среднее время ожидания в очереди:', time\_in\_queue3, 'мин.')

print("\nОбщие итоги работы системы:")

print('Общее время работы системы:', data[0] // 60, 'ч.', data[0] % 60, 'мин.')

print('Среднее время выпонения задачи:', round(avg\_work\_time, 2), 'мин.')

print('Среднее число каналов под обслуживанием:', round((data[4] + data[5] + data[6]) / data[0], 2))

print('Средний процент простоя:', stop\_perc, '%')

print('Абсолютная пропускная способность:', round(A, 2))

print('Среднее время ожидания заявки в очереди', round((data[13] + data[14] + data[15]) / 3, 2), 'мин.')

print('Среднее время пребывания заявки в системе:', round(time\_in\_system, 2), 'мин.')