Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра вычислительных систем

**Лабораторная работа №4**

«Организация очереди»

Выполнил:

студент гр. МГ-211 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Бурдуковский И.А./

подпись

Проверил:

Профессор

кафедры ВС \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Родионов А.С./

Новосибирск

2023 г.

Оглавление

[Задание 3](#_Toc133933349)

[Выполнение работы 4](#_Toc133933350)

# Задание

Пусть есть некоторая очередь.

t – момент поступления i-го требования в систему;

τ – интервал поступления;

s – время обслуживания;

w – время ожидания i-го клиента до начала обслуживания.

Необходимо определить, какой вариант организации очереди будет давать наименьшее время ожидания. Наилучшим будет считаться вариант, имеющий наименьшую дисперсию. При этом необходимо учитывать, что отношение математического ожидания значений интервала поступления к математическому ожиданию значений времени обслуживания было меньше нуля.

# Выполнение работы

Программа генерирует случайные интервалы поступления от 0 до 4 и время обслуживания от 0 до 3.

Затем вычисляются t по формуле: ti = ti-1+ τi

Далее находятся W по формуле: wi = max(0, wi-1 + si-1 - τi)

Рассмотрим три варианта сортировки массива для трёх разных случаев:

1. Оба параметра (τ и s) – случайны

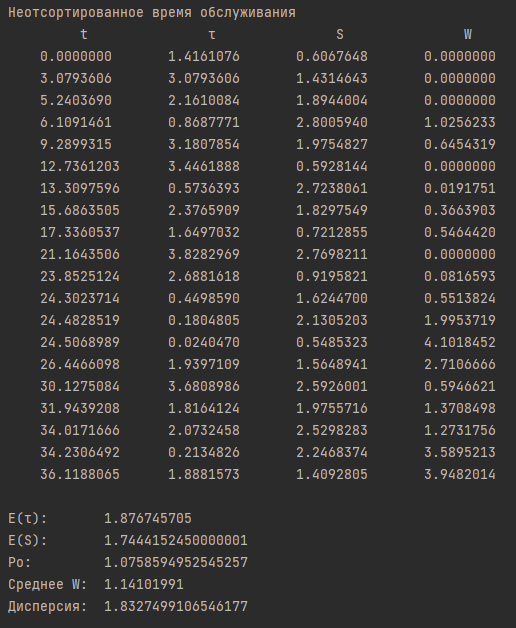


Рис. 1 – Расчет дисперсии и среднего для неотсортированной очереди

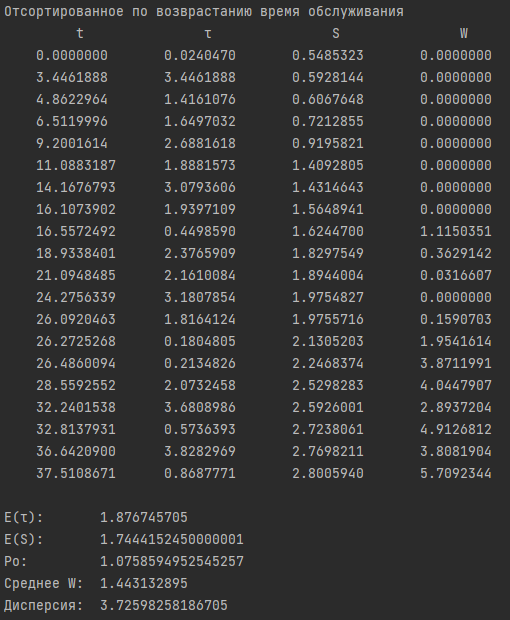


Рис. 2 – Расчет дисперсии и среднего для очереди, отсортированной по возрастанию времени обслуживания

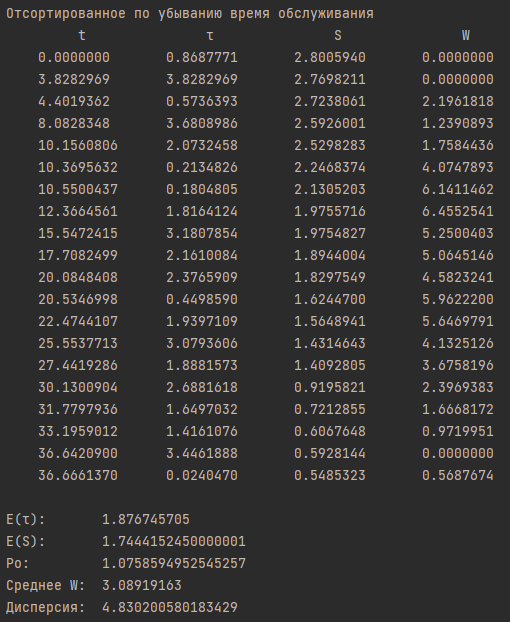
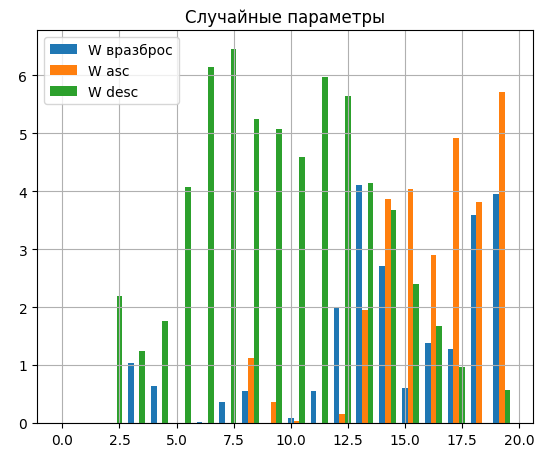


Рис. 3 – Расчет дисперсии и среднего для очереди, отсортированной по убыванию времени обслуживания

Для случайных интервалов поступления и случайного времени обслуживания наименьшая дисперсия и среднее время ожидания получаются в случае неотсортированной очереди.



На гистограмме также наблюдается, что, в среднем, время ожидания меньше в случае неотсортированной очереди. Но можно заметить, что в случае отсортированной очереди по возрастанию времени обслуживания – заметное время ожидания появляется только во второй половине очереди, тогда как при неотсортированной – время ожидания присутствует почти на всей протяженности очереди.

1. Интервал поступления (τ) постоянен, а время обслуживания (s) – случайно

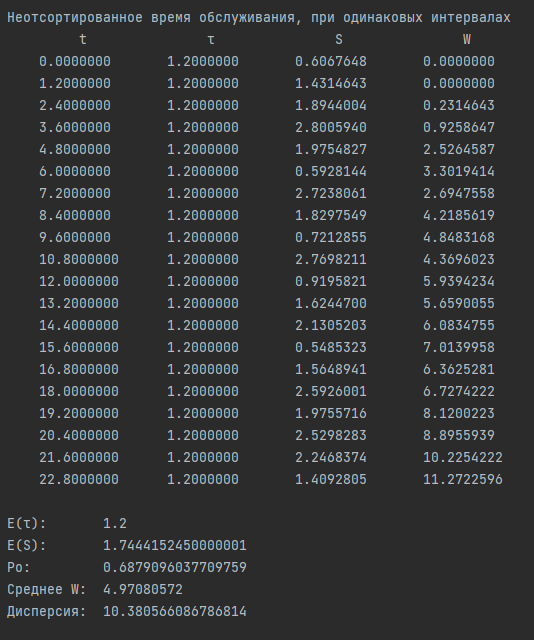


Рис. 4 – Расчет дисперсии и среднего для неотсортированной очереди

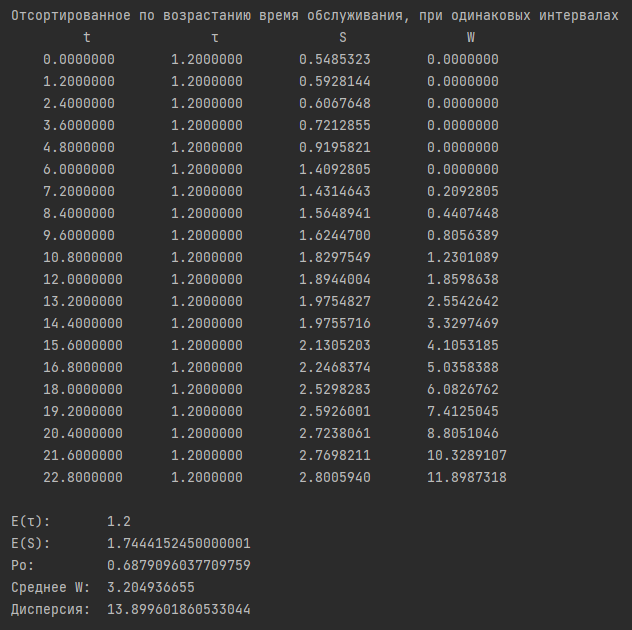


Рис. 5 – Расчет дисперсии и среднего для очереди, отсортированной по возрастанию времени обслуживания

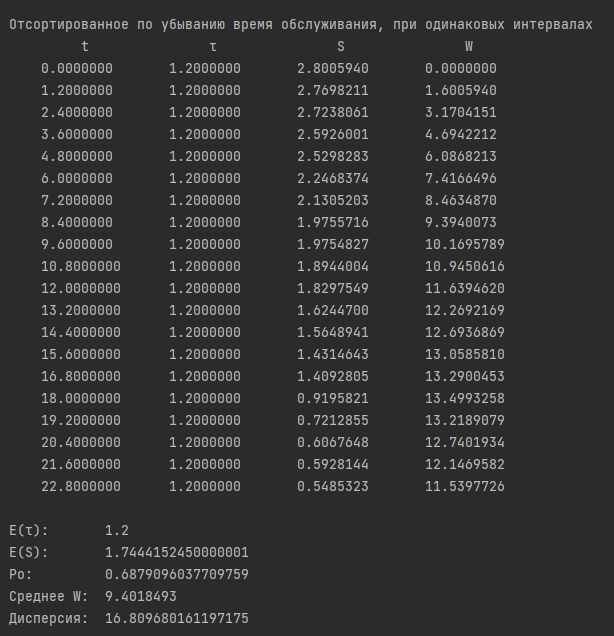
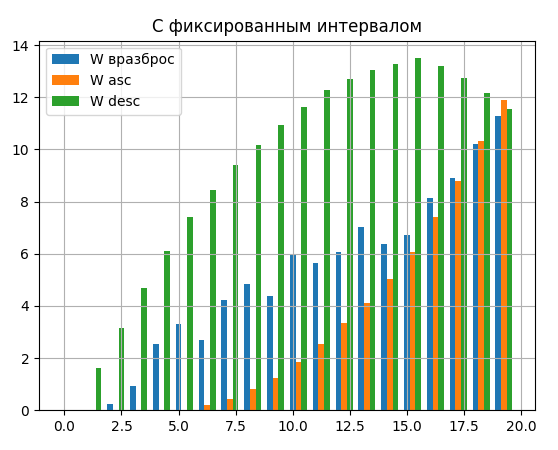


Рис. 6 – Расчет дисперсии и среднего для очереди, отсортированной по убыванию времени обслуживания

Для постоянных интервалов поступления и случайного времени обслуживания наименьшая дисперсия получается в случае неотсортированной очереди, а наименьшее среднее время ожидания – в случае отсортированной по возрастанию времени обслуживания.



На гистограмме наблюдается, что, время ожидания меньше в случае отсортированной по возрастанию очереди.

1. Интервал поступления (τ) случаен, а время обслуживания (s) – постоянно

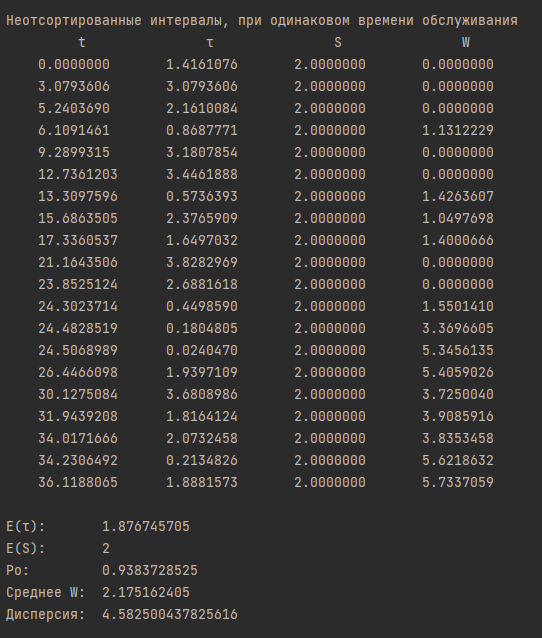


Рис. 7 – Расчет дисперсии и среднего для неотсортированной очереди

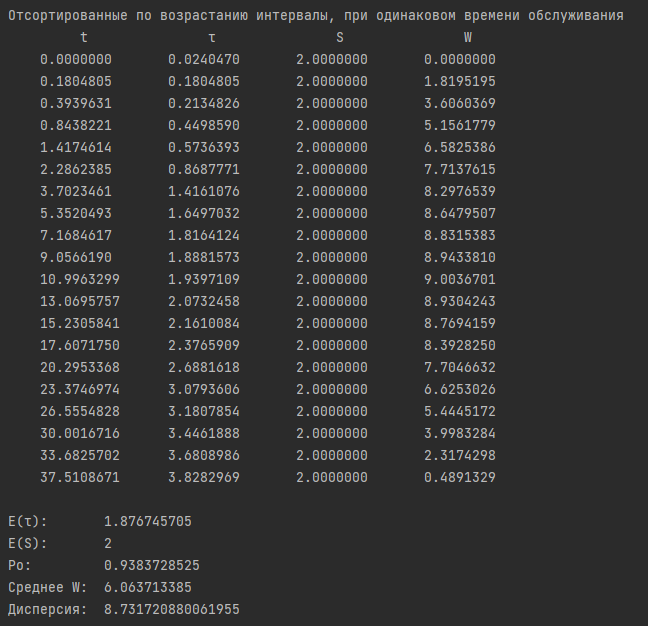


Рис. 8 – Расчет дисперсии и среднего для очереди, отсортированной по возрастанию интервалов поступления

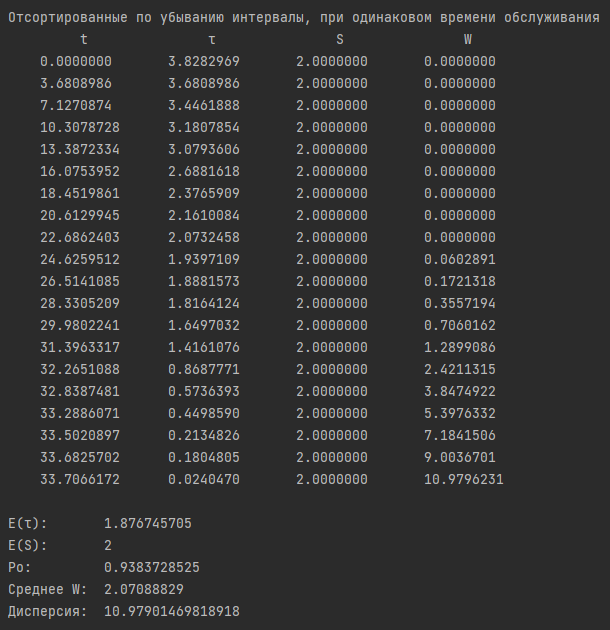
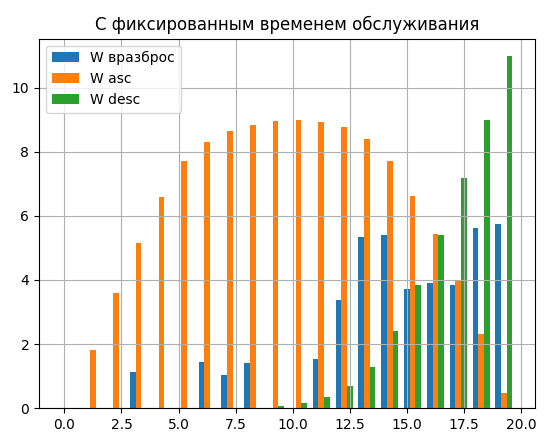


Рис. 9 – Расчет дисперсии и среднего для очереди, отсортированной по убыванию интервалов поступления

Для случайных интервалов поступления и постоянного времени обслуживания наименьшая дисперсия получается в случае неотсортированной очереди, а наименьшее среднее время ожидания – в случае отсортированной по убыванию интервалов поступления.



Из гистограммы получаем, что наилучшим результатом будет являться случай с отсортированным временем поступления по убыванию, т.к. в таком случае высокое время ожидания будет возникать только к концу очереди.

Таким образом, при заданных рамках генерации τ и S, наименьшую дисперсию среди всех вариантов продемонстрировал случай, когда интервал и время обслуживания случайны, а очередь неотсортированна. Наименьшее среднее время ожидания также был продемонстрирован этим случаем.

# Листинг

import random  
import copy  
from statistics import variance, pvariance, mean  
from operator import itemgetter  
import numpy as np  
import matplotlib.pyplot as plt  
  
n = 10  
size = 20  
max\_interval = 4  
max\_service\_time = 3  
  
  
def generate\_queue():  
 matrix = []  
 for i in range(size):  
 interval = round(random.uniform(0, max\_interval), 7)  
 service\_time = round(random.uniform(0, max\_service\_time), 7)  
 line = {'interval': interval, 'service\_time': service\_time, 'add\_moment': 0, 'await\_time': 0}  
 matrix.append(line)  
 return matrix  
  
  
def set\_all\_values(matrix, key, value):  
 for i in range(len(matrix)):  
 matrix[i][key] = value  
  
  
def calculate\_awaits(matrix):  
 i = 1  
 matrix[0]['add\_moment'] = 0  
 matrix[0]['await\_time'] = 0  
 while i < len(matrix):  
 matrix[i]['add\_moment'] = round(matrix[i-1]['add\_moment'] + matrix[i]['interval'], 7)  
 matrix[i]['await\_time'] = max(0, round(matrix[i-1]['await\_time'] + matrix[i-1]['service\_time'] - matrix[i]['interval'], 7))  
 i += 1  
  
  
def print\_matrix(matrix):  
 print("\t\t t \t\t\t τ \t\t\t\t S \t\t\t\t W")  
 for i in range(len(matrix)):  
 print("\t%7.7f" % (matrix[i]['add\_moment']), end=' ')  
 print("\t%7.7f" % (matrix[i]['interval']), end=' ')  
 print("\t%7.7f" % (matrix[i]['service\_time']), end=' ')  
 print("\t%7.7f" % (matrix[i]['await\_time']), end=' ')  
 print()  
 print()  
  
  
def estimate\_matrix(matrix):  
 # среднее арифметическое интервала / мат ожидание  
 interval\_middle = mean(matrix[i]['interval'] for i in range(len(matrix)))  
 # среднее арифметическое времени обслуживания / мат ожидание  
 service\_time\_middle = mean(matrix[i]['service\_time'] for i in range(len(matrix)))  
 # отношение  
 Po = interval\_middle/service\_time\_middle  
 # среднее арифметическое времени ожидания  
 await\_time\_middle = mean(matrix[i]['await\_time'] for i in range(len(matrix)))  
 # дисперсия генеральной совокупности  
 dispersion = pvariance([matrix[i]['await\_time'] for i in range(len(matrix))])  
  
 print(f"E(τ): {interval\_middle}")  
 print(f"E(S): {service\_time\_middle}")  
 print(f"Po: {Po}")  
 print(f"Среднее W: {await\_time\_middle}")  
 print(f"Дисперсия: {dispersion}")  
 print()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 w\_unsorted = []  
 w\_sorted\_asc = []  
 w\_sorted\_desc = []  
 width = 0.25  
  
 print('Неотсортированное время обслуживания')  
 matrix = generate\_queue()  
 calculate\_awaits(matrix)  
 print\_matrix(matrix)  
 estimate\_matrix(matrix)  
 w\_unsorted = [matrix[i]['await\_time'] for i in range(len(matrix))]  
  
 print('Отсортированное по возврастанию время обслуживания')  
 matrix\_sorted = copy.deepcopy(matrix)  
 matrix\_sorted.sort(key=itemgetter('service\_time'))  
 calculate\_awaits(matrix\_sorted)  
 print\_matrix(matrix\_sorted)  
 estimate\_matrix(matrix\_sorted)  
 w\_sorted\_asc = [matrix\_sorted[i]['await\_time'] for i in range(len(matrix\_sorted))]  
  
 print('Отсортированное по убыванию время обслуживания')  
 matrix\_sorted = copy.deepcopy(matrix)  
 matrix\_sorted.sort(key=itemgetter('service\_time'), reverse=True)  
 calculate\_awaits(matrix\_sorted)  
 print\_matrix(matrix\_sorted)  
 estimate\_matrix(matrix\_sorted)  
 w\_sorted\_desc = [matrix\_sorted[i]['await\_time'] for i in range(len(matrix\_sorted))]  
  
 fig, axs = plt.subplots()  
 axs.bar(np.arange(len(w\_unsorted)), w\_unsorted, width, label='W вразброс')  
 axs.bar(np.arange(len(w\_sorted\_asc))+width, w\_sorted\_asc, width, label='W asc')  
 axs.bar(np.arange(len(w\_sorted\_desc))+width\*2, w\_sorted\_desc, width, label='W desc')  
 plt.title('Случайные параметры')  
 plt.grid()  
 plt.legend()  
 plt.show()  
  
 print('Неотсортированное время обслуживания, при одинаковых интервалах')  
 matrix\_same\_intervals = copy.deepcopy(matrix)  
 set\_all\_values(matrix\_same\_intervals, 'interval', 1.2)  
 calculate\_awaits(matrix\_same\_intervals)  
 print\_matrix(matrix\_same\_intervals)  
 estimate\_matrix(matrix\_same\_intervals)  
 w\_unsorted = [matrix\_same\_intervals[i]['await\_time'] for i in range(len(matrix\_same\_intervals))]  
  
 print('Отсортированное по возрастанию время обслуживания, при одинаковых интервалах')  
 matrix\_sorted = matrix\_same\_intervals  
 matrix\_sorted.sort(key=itemgetter('service\_time'))  
 calculate\_awaits(matrix\_sorted)  
 print\_matrix(matrix\_sorted)  
 estimate\_matrix(matrix\_sorted)  
 w\_sorted\_asc = [matrix\_sorted[i]['await\_time'] for i in range(len(matrix\_sorted))]  
  
 print('Отсортированное по убыванию время обслуживания, при одинаковых интервалах')  
 matrix\_sorted.sort(key=itemgetter('service\_time'), reverse=True)  
 calculate\_awaits(matrix\_sorted)  
 print\_matrix(matrix\_sorted)  
 estimate\_matrix(matrix\_sorted)  
 w\_sorted\_desc = [matrix\_sorted[i]['await\_time'] for i in range(len(matrix\_sorted))]  
  
 fig, axs = plt.subplots()  
 axs.bar(np.arange(len(w\_unsorted)), w\_unsorted, width, label='W вразброс')  
 axs.bar(np.arange(len(w\_sorted\_asc))+width, w\_sorted\_asc, width, label='W asc')  
 axs.bar(np.arange(len(w\_sorted\_desc))+width\*2, w\_sorted\_desc, width, label='W desc')  
 plt.title('С фиксированным интервалом')  
 plt.grid()  
 plt.legend()  
 plt.show()  
  
 print('Неотсортированные интервалы, при одинаковом времени обслуживания')  
 matrix\_same\_service\_time = copy.deepcopy(matrix)  
 set\_all\_values(matrix\_same\_service\_time, 'service\_time', 2)  
 calculate\_awaits(matrix\_same\_service\_time)  
 print\_matrix(matrix\_same\_service\_time)  
 estimate\_matrix(matrix\_same\_service\_time)  
 w\_unsorted = [matrix\_same\_service\_time[i]['await\_time'] for i in range(len(matrix\_same\_service\_time))]  
  
 print('Отсортированные по возрастанию интервалы, при одинаковом времени обслуживания')  
 matrix\_sorted = matrix\_same\_service\_time  
 matrix\_sorted.sort(key=itemgetter('interval'))  
 calculate\_awaits(matrix\_sorted)  
 print\_matrix(matrix\_sorted)  
 estimate\_matrix(matrix\_sorted)  
 w\_sorted\_asc = [matrix\_sorted[i]['await\_time'] for i in range(len(matrix\_sorted))]  
  
 print('Отсортированные по убыванию интервалы, при одинаковом времени обслуживания')  
 matrix\_sorted.sort(key=itemgetter('interval'), reverse=True)  
 calculate\_awaits(matrix\_sorted)  
 print\_matrix(matrix\_sorted)  
 estimate\_matrix(matrix\_sorted)  
 w\_sorted\_desc = [matrix\_sorted[i]['await\_time'] for i in range(len(matrix\_sorted))]  
  
 fig, axs = plt.subplots()  
 axs.bar(np.arange(len(w\_unsorted)), w\_unsorted, width, label='W вразброс')  
 axs.bar(np.arange(len(w\_sorted\_asc))+width, w\_sorted\_asc, width, label='W asc')  
 axs.bar(np.arange(len(w\_sorted\_desc))+width\*2, w\_sorted\_desc, width, label='W desc')  
 plt.title('С фиксированным временем обслуживания')  
 plt.grid()  
 plt.legend()  
 plt.show()