МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 42

ОТЧЁТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

| старший преподаватель |  |  |  | С.Ю. Гуков |
| --- | --- | --- | --- | --- |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

| ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №3 |
| --- |
| Обработка сетевых пакетов |
| по курсу: |
| АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ |
|  |

РАБОТУ ВЫПОЛНИЛ

| СТУДЕНТ гр. № | 4329 |  |  |  | Д.С. Шаповалова |
| --- | --- | --- | --- | --- | --- |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

ОГЛАВЛЕНИЕ

[Цель работы 3](#_30j0zll)

[Постановка задачи 3](#_1fob9te)

[Схема алгоритма решения 4](#_3znysh7)

[Полное описание реализованной функции 4](#_gqc1nhfmq022)

[Листинг программы 5](#_2et92p0)

[Результат выполнения программы. 5](#_tyjcwt)

[ВЫВОДЫ 6](#_3dy6vkm)

# Цель работы

Реализовать обработчик сетевых пакетов.

# Постановка задачи

Задание: Написать программ, симулирующую обработку сетевых пакетов. Необходимо использовать два любых дополнительных усложнения программы из представленных. Конкретный текст задания приведён в таблице 1.

Таблица 1. Индивидуальное задание

| № | Текст задания | Вход | Выход |
| --- | --- | --- | --- |
| 0 | Реализовать симулятор обработки сетевых пакетов. Есть процессор. Если процессор начинает обрабатывать пакет i (что занимает время duration[i]), он не прерывается и не останавливается до тех пор, пока не обработает пакет. имеется сетевой буфер размера size. До начала обработки пакеты хранятся в буфере. Если он полностью заполнен в момент поступления пакета (есть size пакетов, поступивших ранее, которые до сих пор не обработаны), этот пакет отбрасывается и уже не будет обработан.  Компьютер обрабатывает пакеты в порядке их поступления. Он начинает обрабатывать следующий пакет из буфера сразу после того, как обработает текущий пакет  Доп.условия:   1. Случайные сбои. Добавить вероятность случайного сбоя при обработке пакета. Если пакет не обработан из-за сбоя, он перемещается в конец очереди с удвоением времени обработки или отбрасывается. 2. Ограничение по времени обработки. Нужно ввести максимальное время, которое пакет может провести в буфере до начала обработки. Если это время истекает, пакет отбрасывается. | Первая строка входа содержит размер буфера size и число пакетов n. Каждая из следующих n строк содержит два числа: время arrival[i] прибытия i-го пакета и время duration[i], необходимое на его обработку. Гарантируется, что arrival[1] ≤ arrival[2] ≤ ··· ≤ arrival[n]. При этом может оказаться, что arrival[i−1] = arrival[i]. В таком случае считаем, что пакет i−1 поступил раньше пакета i. | Для каждого из n пакетов выведите время, когда процессор начал его обрабатывать, или −1, если пакет был отброшен. |

# Схема алгоритма решения

1. Ввод данных в коде
2. Функция обработки
   1. Инициализация переменных
   2. Наполнение очереди пакетов
   3. Обработка пакетов
   4. Возврат результатов

# Полное описание реализованной функции

Функция обработки:

1. Инициализация переменных:

*current\_time* - текущее время обработки.

*result* - список для хранения результатов обработки (время начала обработки или -1 для отбрасываемых пакетов).

*buffer* - очередь для хранения активных пакетов, которые обрабатываются.

*arrivals* - очередь для поступающих пакетов.

1. Наполнение очереди пакетов

Пакеты помещаются в очередь *arrivals* с использованием их времени прибытия, длительности и приоритета.

1. Обработка пакетов

Пока есть пакеты в очереди *arrivals*, выполняется следующее:

1. Извлекается пакет из очереди.
2. Удаляются завершенные пакеты из *buffer*.
3. Проверяется, достаточно ли места в буфере:

**Если место есть:**

1. Устанавливается время начала обработки (максимум из текущего времени и времени прибытия).
2. **Реализуется 2 усложнение.** Проверяется, превышает ли время ожидания пакет от установленного времени начала до времени его прибытия максимальное допустимое значение max\_wait\_time. Если да, пакет отбрасывается, и в результат добавляется -1.
3. **Реализуется 1 усложнение.** Генерируется случайное число для проверки на сбой. Если сбой произошел, пакет откладывается (добавляется в конец очереди с удвоенной длительностью).
4. Если пакет успешно обработан, обновляется текущее время и пакет добавляется в буфер.

**Если места нет**, то пакет отбрасывается и выводится -1.

1. Возврат результатов - списка результатов обработки пакетов.

# Листинг программы

import heapq

import random

def packet\_processing(size, n, packets, failure\_probability=0.1, max\_wait\_time=5):

current\_time = 0

result = []

buffer = [] # Это будет хранить (время\_окончания\_обработки, длительность, время\_прихода)

arrivals = [] # Очередь для поступающих пакетов

# Добавляем пакеты в arrivals с приоритетом

for arrival, duration in packets:

heapq.heappush(arrivals, (arrival, duration))

while arrivals:

arrival, duration = heapq.heappop(arrivals)

# Вычисляем полное время, которое пакет пробыл в системе

total\_wait\_time = current\_time - arrival

# Удаляем завершенные пакеты из буфера

while buffer and buffer[0][0] <= arrival:

buffer.pop(0)

# Проверяем размеры буфера

if len(buffer) < size:

# Устанавливаем время начала обработки

start\_time = max(current\_time, arrival)

# Проверяем время нахождения в системе

if total\_wait\_time > max\_wait\_time:

print("Пакет отброшен из-за превышения времени ожидания!")

result.append(-1) # Пакет отброшен

continue

if random.random() < failure\_probability:

# Произошел сбой

print("Сбой!")

result.append(-1)

# Удвоение времени обработки и добавление в очередь

new\_duration = min(duration \* 2, 100) # Ограничиваем максимальное время (например, 100)

heapq.heappush(arrivals, (arrival, new\_duration))

else:

# Пакет успешно обработан

result.append(start\_time)

# Обновляем текущее время

current\_time = start\_time + duration

# Добавляем пакет в буфер с временем окончания и временем прибытия

heapq.heappush(buffer, (current\_time, duration, arrival))

else:

result.append(-1) # Буфер переполнен

return result

# Пример использования

size, n = 2, 2 # размер буфера и кол-во пакетов

max\_wait\_time = 5 # максимальное время ожидания

input\_data = [

(0, 6),

(1, 1)

] # когда пришёл, вес

output = packet\_processing(size, n, input\_data, max\_wait\_time=max\_wait\_time-1)

for time in output:

print(time)

# Результат выполнения программы.

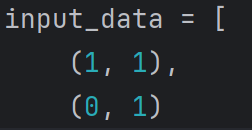


Рисунок 2.1.1 - 1-ый результат работы программы - Ввод

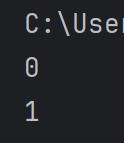


Рисунок 2.1.2 - 1-ой результат работы программы без сбоя - Вывод

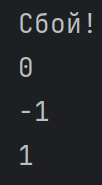


Рисунок 2.1.3 - 1-ой результат работы программы со сбоем - Вывод

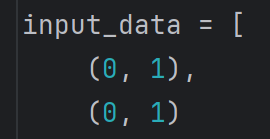


Рисунок 2.2.1 - 2-ой результат работы программы - Ввод

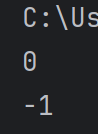


Рисунок 2.2.2 - 2-ой результат работы программы без сбоя - Вывод

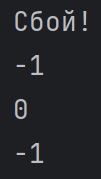


Рисунок 2.2.3 - 2-ой результат работы программы со сбоем- Вывод

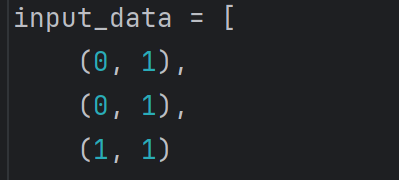


Рисунок 2.3.1 - 3-ой результат работы программы - Ввод, размер буфера 1

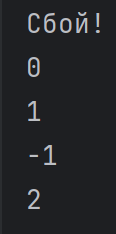


Рисунок 2.3.2 - 3-ой результат работы программы со сбоем - Вывод

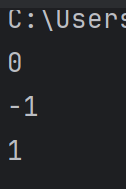


Рисунок 2.3.3 - 3-ой результат работы программы без сбоя - Вывод

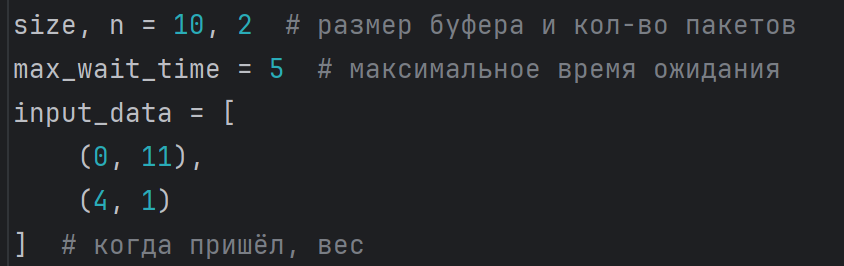


Рисунок 2.4.1 - 4 ввод

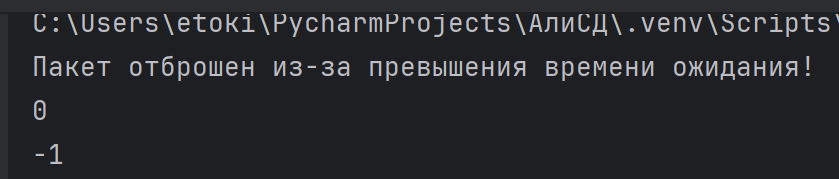


Рисунок 2.4.2 - 4 вывод

# 

# Выводы

В ходе выполнения лабораторной работы мной были освоены и изучены: понятие сетевой пакет, обработчик сетевых пакетов, сетевой буфер; алгоритм обработки сетевых пакетов. Написанная программа была протестирована, полученный результат соответствует значению в примере.