МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное образовательное учреждение высшего образования
САНКТ – ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

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

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

Обработка сетевых пакетов

по курсу: АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4321		К.А. Лебедев
	подпись, дата	инициалы, фамилия

Санкт-Петербург, 2024

СОДЕРЖАНИЕ

1 Цель работы 3
2 Задание..... 4
3 Ход разработки 5
4 Исходный код программы 6
5 Результаты работы программы 9
6 Вывод..... 10

1 Цель работы

Реализовать обработчик сетевых пакетов.

2 Задание

Программа может быть написана на любом языке программирования. Вход. Размер буфера size и число пакетов n, а также две последовательности arrival1, ..., arrivaln и duration1, ..., durationn, обозначающих время поступления и длительность обработки n пакетов. Выход. Для каждого из данных n пакетов необходимо вывести время начала его обработки или −1, если пакет не был обработан (это происходит в случае, когда пакет поступает в момент, когда в буфере компьютера уже находится size пакетов). Реализовать симулятор обработки сетевых пакетов.

Для i-го пакета известно время его поступления arrivali, а также время durationi, необходимое на его обработку. В вашем распоряжении имеется один процессор, который обрабатывает пакеты в порядке их поступления. Если процессор начинает обрабатывать пакет i (что занимает время durationi), он не прерывается и не останавливается до тех пор, пока не обработает пакет. У компьютера, обрабатывающего пакеты, имеется сетевой буфер размера size. До начала обработки пакеты хранятся в буфере.

Если буфер 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. Если несколько пакетов поступает в одно и то же время, они все будут сперва сохранены в буфер (несколько последних из них могут быть отброшены, если буфер заполнится).

Компьютер обрабатывает пакеты в порядке их поступления. Он начинает обрабатывать следующий пакет из буфера сразу после того, как обработает текущий пакет. Компьютер может простаивать, если все пакеты уже обработаны и в буфере нет пакетов. Пакет освобождает место в буфере сразу же, как компьютер заканчивает его обработку. Формат входа. Первая строка входа содержит размер буфера size и число пакетов n. Каждая из следующих n строк содержит два числа: время arrivali прибытия i-го пакета и время durationi, необходимое на его обработку.

Гарантируется, что arrival1 ≤ arrival2 ≤ ⋯ ≤ arrivaln. При этом может оказаться, что arrivali−1 = arrivali. В таком случае считаем, что пакет i−1 поступил раньше пакета i. Формат выхода. Для каждого из n пакетов выведите время, когда процессор начал его обрабатывать, или −1, если пакет был отброшен. Ограничения. Все числа во входе целые. $1 \le size \le 10^5$; $0 \le n \le 10^5$; $0 \le arrivali \le 10^6$; $0 \le durationi \le 10^3$; arrivali ≤ arrivali+1 для всех $1 \le i \le n-1$.

3 Ход разработки

Были реализованы класс задачи и функция для вычисления высоты дерева, написана утилита для прочтения чисел из командной строки на языке TypeScript, интегрирован компилятор в JavaScript, который сразу собирает TypeScript проект без предварительной компиляции.

4 Исходный код программы

```
import { readlineInterface } from "../utils/readline";

class Task {
  private bufferSize: number;
  private n: number;
  private maxTime: number | undefined;
  private arrival: Array<number>;
  private duration: Array<number>;
  private result: Array<number | string>;
  private currentTime: number;
  private buffer: Array<{ arrival: number; duration: number }>;

  constructor() {
    this.bufferSize = 0;
    this.maxTime = undefined;
    this.n = 0;
    this.arrival = [];
    this.duration = [];
    this.result = [];
    this.currentTime = 0;
    this.buffer = [];

    this.read(); // инициализация таски
  }

  getRandom() {
    return Math.floor(Math.random() * (11)) > 3;
  }

  getAnswer() {
    let index = 0;

    while (index < this.n || this.buffer.length > 0) {
      while (index < this.n && this.arrival[index] <= this.currentTime) {
        if(this.maxTime ? this.getRandom() : true) {
        if (this.buffer.length < this.bufferSize && (this.maxTime ?
this.duration[index] <= this.maxTime : true)) {
          this.buffer.push({
            arrival: this.arrival[index],
            duration: this.duration[index],
          });
          this.result[index] = this.currentTime;
        } else {
          this.result[index] = -1;
```

```
        }
      }else {
      }
        index++;
      }

      if (this.buffer.length > 0) {
        const packet = this.buffer.shift();
        if (packet) {
          this.currentTime += packet.duration;
        }
      } else {
        if (index < this.n) {
          this.currentTime = this.arrival[index];
        }
      }
    }

    this.result.forEach((start) => {
      console.log(start);
    });
  }

  read() {
    readlineInterface.question("", (length: string) => {
      this.bufferSize = Number(length.split(" ")[0]);
      this.n = Number(length.split(" ")[1]);
      this.maxTime = Number(length.split(" ")[2]);

      if (
        isNaN(this.bufferSize) ||
        this.bufferSize <= 0 ||
        isNaN(this.n) ||
        this.n < 0
      ) {
        console.log("Пожалуйста, введите корректное положительное
число.");
        readlineInterface.close();
        return;
      }

      this.arrival = [];
      this.duration = [];

      const readPackets = (count: number) => {
```

```
        if (count === 0) {
          this.getAnswer();
          readlineInterface.close();
          return;
        }

        readlineInterface.question("", (packet: string) => {
          const [arrival, duration] = packet.split(" ").map(Number);
          this.arrival.push(arrival);
          this.duration.push(duration);
          readPackets(count - 1);
        });
      };

      readPackets(this.n);
    });
  }
}

export default Task;
```

5 Результаты работы программы

Результат работы программы включает в себя вывод очереди команд

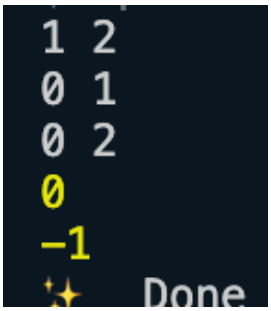

Рисунок 1 – тест программы №1

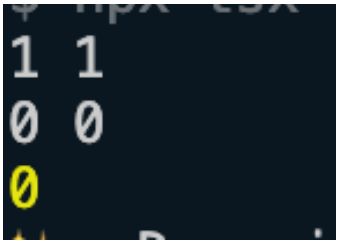

Рисунок 2 – тест программы №2

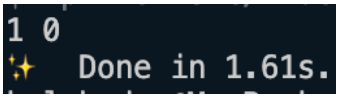

Рисунок 3 – тест программы №3

6 Вывод

Алгоритмы и структуры данных играют ключевую роль в эффективном решении задач в программировании. Они позволяют оптимизировать процессы и ускорить выполнение операций при правильном использовании.

Основная цель программы — создать реализацию буфера в соответствии с заданием. Буфер — область памяти, используемая для временного хранения данных при вводе или выводе. Обмен данными может происходить как с внешними устройствами, так и с процессами в пределах компьютера. Использование буфера дает возможность работать и обрабатывать поток данных частично.

Лабораторная работа выполнена, структура изучена.