**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

по дисциплине **«Алгоритмы и структуры данных»**

на тему: **«Биты и задачи комбинаторики»**

Выполнил: студент гр. ИП-21

М. С. Есис

Принял: преподаватель

В. Н. Шибеко

Дата сдачи отчета: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2024

**Цель работы**: ознакомиться с алгоритмами для выполнение задач связанных с комбинаторикой.

**Задания к лабораторной работе:**

**Задание 1**

Реализация задачи о рюкзаке

import itertools

def FindMax(masses, costs, maxMass):

    n = len(masses)

    maxValue = 0

    for r in range(1, n + 1):

        for combination in itertools.combinations(range(n), r):

            totalMass = sum(masses[i] for i in combination)

            totalCost = sum(costs[i] for i in combination)

        if totalMass < maxMass and totalCost > maxValue:

            maxValue = totalCost

    return [maxMass, maxValue]

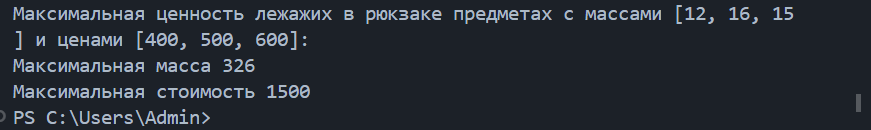
mass = [12, 16, 15]

value = [400, 500, 600]

maxMass = 326

result = FindMax(mass, value, maxMass)

print(f"Максимальная ценность лежажих в рюкзаке предметах с массами {mass} и ценами {value}:\nМаксимальная масса {result[0]}\nМаксимальная стоимость {result[1]}")



**Задание 2**

Реализация задачи о коммивояжёре

import itertools

def find\_optimal\_route(cities, matrix):

    permutations = list(itertools.permutations(cities))

    optimal\_route = None

    optimal\_distance = float('inf')

    counter = 0

    # Проходимся по всем перестановкам городов

    for permutation in permutations:

        distance = 0

        # Вычисляем суммарное расстояние для текущей перестановки

        for i in range(len(permutation) - 1):

            from\_city = permutation[i]

            to\_city = permutation[i + 1]

            distance += matrix[from\_city][to\_city]

        counter += 1

        print(f"Маршрут[{counter}]: {permutation}, общая дистанция: {distance}")

        # Если текущая перестановка имеет меньшую длину, обновляем оптимальный маршрут

        if distance < optimal\_distance:

            optimal\_distance = distance

            optimal\_route = permutation

    return [optimal\_route, optimal\_distance]

cities = [0, 1, 2, 3, 4]

matrix = [

    [float('inf'), 16, 17, 18, 19],

    [1, float('inf'), 3, 4, 16],

    [2, 1, float('inf'), 6, 17],

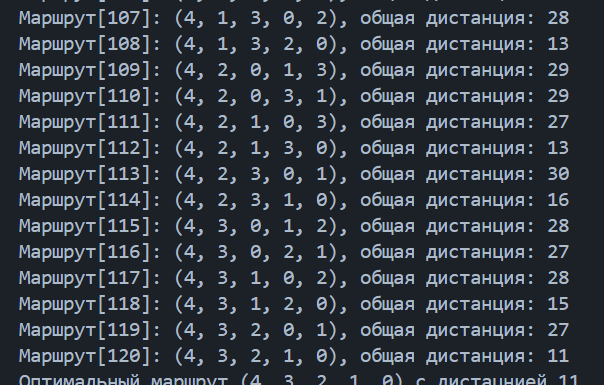
    [1, 2, 1, float('inf'), 18],

    [5, 6, 7, 8, float('inf')]

]

result = find\_optimal\_route(cities=cities, matrix=matrix)

print(f"Оптимальный маршрут {result[0]} с дистацнией {result[1]}")

****

**Вывод**: ознакомилась с алгоритмами для выполнение задач связанных с комбинаторикой.