

|  |  |
| --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное бюджетное образовательное учреждение высшего образования  **«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ**  **ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»** |  |

**Кафедра «Высшая математика»**

**Лабораторная работа № 3**

по дисциплине

«Структуры и алгоритмы обработки данных»

на тему:

**«**Методы динамического программирования**»**

**Выполнил:**

Учебная группа: 1бПМ   
ФИО: Борисов И. М.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель лабораторной работы:**

Должность: старший преподаватель

Звание: б/з

ФИО: Кутейников И. А.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Цель:

С помощью методов динамического программирования реализовать

алгоритм решения зада в соответствии с вариантом. Предусмотреть ввод

начальных условий и вывод решения.

Варианты:

1 Задача о разрезании стержня;

2 Перемножение цепочки матриц;

3 Наидлиннейшая общая последовательность;

Результаты лабораторной работы оформить в виде отчета с результатами

работы программы.

Формулировка задач и их аналитическое решение представлены в книге

Т.Кормен, Ч.Лейзерсон, Р.Ривест, К.Штайн - Алгоритмы. Построение и

анализ.

в главе 15

Задача 1 - 15.1

Задача 2 - 15.2

Задача 3 - 15.3

Код:

def matrix\_chain\_order(p):

    n = len(p)  # Количество матриц = длина массива размеров - 1

    m = [[0] \* n for \_ in range(n)]  # Таблица для хранения минимальных операций

    s = [[0] \* n for \_ in range(n)]  # Таблица для хранения индексов разбиения

    # m[i][i] = 0 для всех i, так как умножение одной матрицы не требует операций

    for i in range(n):

        m[i][i] = 0

    # l — длина цепочки матриц

    for l in range(2, n):  # Длина цепочки от 2 до n-1

        for i in range(n - l):  # Начало цепочки

            j = i + l - 1  # Конец цепочки

            m[i][j] = float('inf')

            for k in range(i, j):

                # q — стоимость умножения цепочек A\_i...A\_k и A\_k+1...A\_j

                q = m[i][k] + m[k + 1][j] + p[i] \* p[k + 1] \* p[j + 1]

                if q < m[i][j]:

                    m[i][j] = q

                    s[i][j] = k

    print\_order(s, 0, n - 2)  # Исправлено на нулевую индексацию

    print()

    return m[0][n - 2]

def print\_order(s, i, j):

    if i == j:

        print(f"A{i + 1}", end="")  # Коррекция индекса для вывода

    else:

        print("(", end="")

        print\_order(s, i, s[i][j])

        print\_order(s, s[i][j] + 1, j)

        print(")", end="")

if \_\_name\_\_ == "\_\_main\_\_":

    n = int(input("Введите количество матриц: "))  # Число матриц

    p = list(map(int, input("Введите размеры матриц: ").split()))  # Размеры матриц

    if len(p) != n + 1:

        print("Ошибка: количество введённых размеров должно быть равно количеству матриц + 1.")

    else:

        print(f"Минимальное количество умножений: {matrix\_chain\_order(p)}")

}

Результат:

Введите количество матриц: 3

Введите размеры матриц: 10 50 20 30

((A1A2)A3)

Минимальное количество умножений: 16000

Вывод:

В ходе данной лабораторной работы была решена задача оптимального перемножения матриц при помощи методов динамического программирования. Было получено понимание того, как работает динамическое программирование, а также, как при помощи полученных знаний оптимизировать решения множества задач.