**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(МИИТ)**

|  |
| --- |
| **ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Кафедра «Управление и защита информации»

**Лабораторная работа №6**

**«**Реализация алгоритмов умножения матриц**»**

по дисциплине

**«Методы программирования»**

**Выполнил:** ст. гр. ТКИ-311,

Виноградова В.Л.

**Проверил:**

к.т.н., доц. Логинова Л.Н.,

к.т.н., доц. Сафронов А.И.

**Москва – 2022 г.**

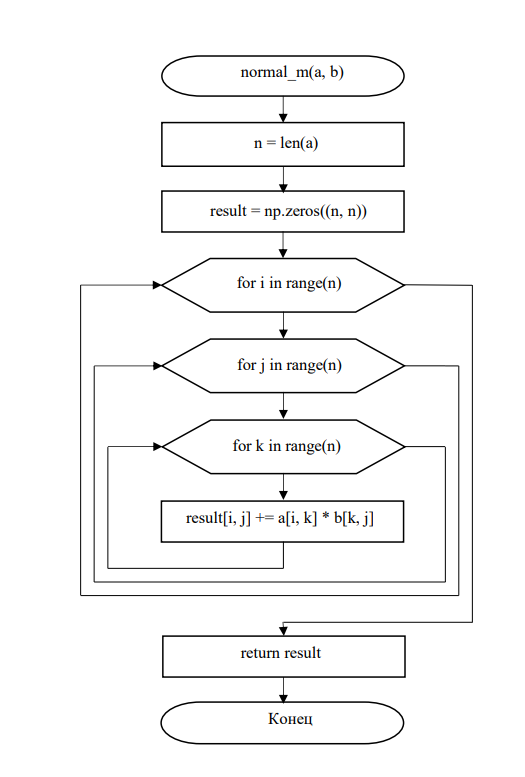
1. Цель работы

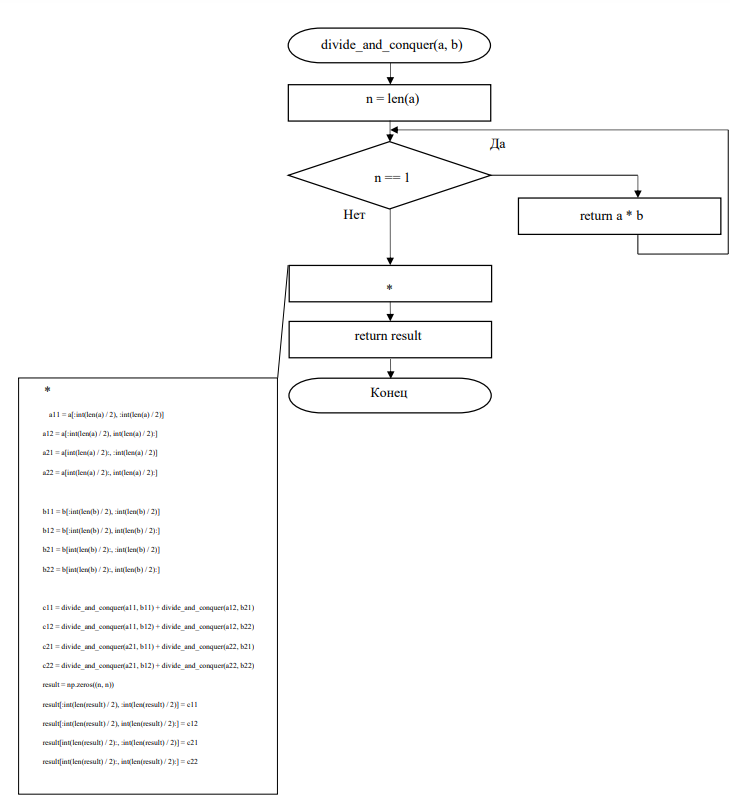
Изучить алгоритмы умножения двух матриц вручную, методом «Разделяй и властвуй, а также методом Штрассена. Научиться реализовывать данные алгоритмы, ориентируясь на псевдокод.

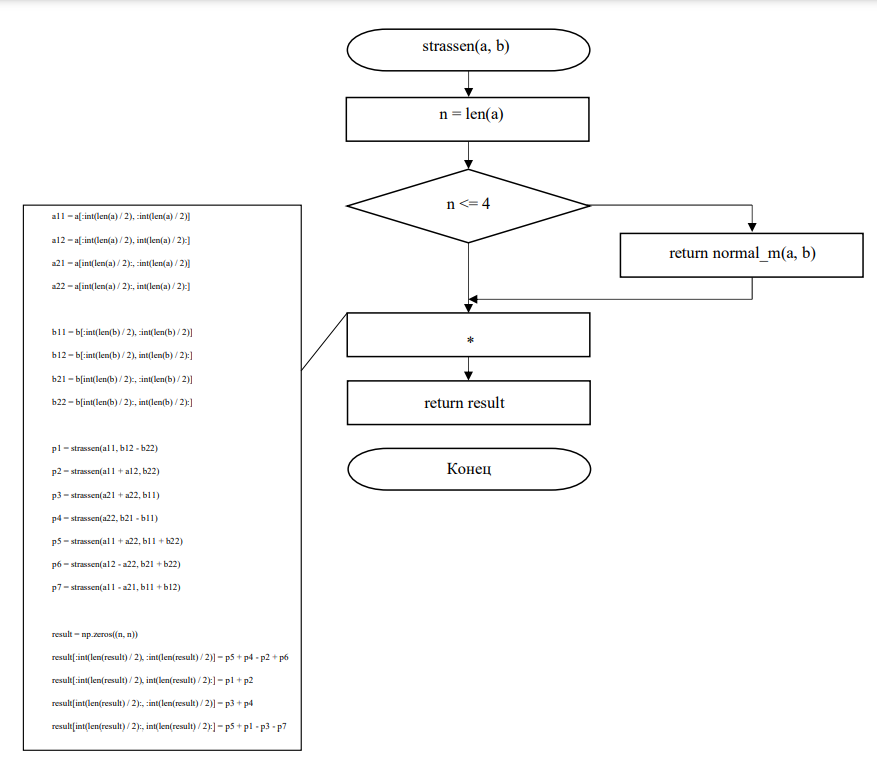
1. Постановка задачи

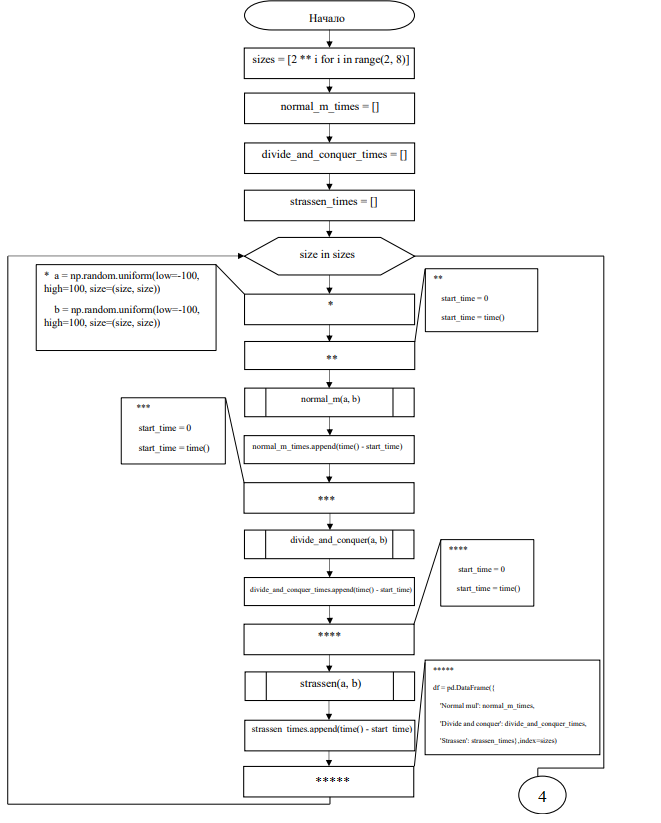
Реализовать 3 алгоритма умножения матриц на любом языке программирования. Сделать выводы на основе работы программ, сравнить реализованные методы.

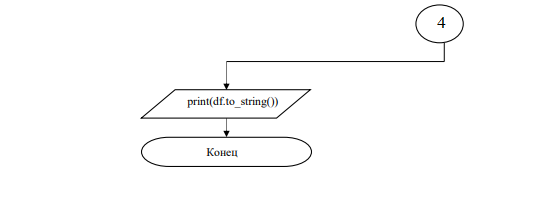
1. Блок-схема алгоритма











1. Тестовые примеры:

Необходимо перемножить матрицы

A = B =

1. Используем алгоритм полного перебора.

Матрица C = A \* B будет иметь следующий вид:

C =

1. Используем метод Штрассена.

= \*

= (( = -1 \* 15 = -15

= (( = 5 \* 13 = 65

= (= -1 \* 11 = -11

= ( = 4 \* 8 = 32

= ( = -2

= ( = 4 \* 2 = 8

= (= 6 \* 5 = 30

Вычисляем по формулам

= 50 – 32 + 8 = 26

= 32 - 2 = 30

= 30 + 8 = 38

= 76 – 2 – 30 = 44

C =

Необходимо умножить матрицу А на матрицу B:

A = B =

Используем алгоритм «Разделяй и властвуй»

Исходные матрицы делим на 4 части

Матрица А будет иметь следующие четверти:

A1 = , A2 = , A3 = , A4 =

Матрица B будет иметь следующие четверти:

B1 = , B2 = , B3 = , B4 =

Делим на 4 части каждую матрицу, полученную в результате первого деления:

Получим следующие четверти:

A11 = , A12 = , A13 = , A14 =

A21 = , A21 = , A23 = , A24 =

A31 = , A32 = , A33 = , A34 =

A41 = , A42 = , A43 = , A44 =

B11 = ,B12 = , B13 = , B14 =

B21 = , B21 = , B23 = , B24 =

B31 = , B32 = , B33 = , B34 =

B41 = , B42 = , B43 = , B44 =

Из полученных матриц

A11 = , A12 = , A13 = , A14 =

B11 = ,B12 = , B13 = , B14 =

формируем первое слагаемое первой четверти матрицы произведения:

C11 = A11\*B11 + A12\*B21 = 1 \* 5 + 2 \* 6 = 17

C12 = A11\*B12 + A12\*B14 = 1 \* 7 + 2 \* 8 = 23

C21 = A13\*B11 + A14\*B13 = 1 \* 5 + 4 \* 6 = 29

C22 = A13\*B12 + A14\*B14 = 1 \* 7 + 4 \* 8 = 39

С111 =

Далее из матриц

A21 = , A21 = , A23 = , A24 =

B31 = , B32 = , B33 = , B34 =

формируем второе слагаемое первой четверти матрицы произведения:

C11 = A21\*B31 + A22\*B31 = 3 \* 7 + 4 \* 8 = 53

C12 = A21\*B32 + A22\*B34 = 3 \* 5 + 4 \* 6 = 39

C21 = A23\*B31 + A24\*B33 = 3 \* 7 + 2 \* 8 = 37

C22 = A23\*B32 + A24\*B34 = 3 \* 5 + 2 \* 6 = 27

С112 =

Первая четверть матрицы равна:

С111 + С112 =

Далее по аналогии находим остальные четверти матрицы, получим

C =

1. Код программы на Python

from time import time

import numpy as np

import pandas as pd

def normal\_m(a, b):

n = len(a)

result = np.zeros((n, n))

for i in range(n):

for j in range(n):

for k in range(n):

result[i, j] += a[i, k] \* b[k, j]

return result

def divide\_and\_conquer(a, b):

n = len(a)

if n == 1:

return a \* b

else:

a11 = a[:int(len(a) / 2), :int(len(a) / 2)]

a12 = a[:int(len(a) / 2), int(len(a) / 2):]

a21 = a[int(len(a) / 2):, :int(len(a) / 2)]

a22 = a[int(len(a) / 2):, int(len(a) / 2):]

b11 = b[:int(len(b) / 2), :int(len(b) / 2)]

b12 = b[:int(len(b) / 2), int(len(b) / 2):]

b21 = b[int(len(b) / 2):, :int(len(b) / 2)]

b22 = b[int(len(b) / 2):, int(len(b) / 2):]

c11 = divide\_and\_conquer(a11, b11) + divide\_and\_conquer(a12, b21)

c12 = divide\_and\_conquer(a11, b12) + divide\_and\_conquer(a12, b22)

c21 = divide\_and\_conquer(a21, b11) + divide\_and\_conquer(a22, b21)

c22 = divide\_and\_conquer(a21, b12) + divide\_and\_conquer(a22, b22)

result = np.zeros((n, n))

result[:int(len(result) / 2), :int(len(result) / 2)] = c11

result[:int(len(result) / 2), int(len(result) / 2):] = c12

result[int(len(result) / 2):, :int(len(result) / 2)] = c21

result[int(len(result) / 2):, int(len(result) / 2):] = c22

return result

def strassen(a, b):

n = len(a)

if n <= 4:

return normal\_m(a, b)

else:

a11 = a[:int(len(a) / 2), :int(len(a) / 2)]

a12 = a[:int(len(a) / 2), int(len(a) / 2):]

a21 = a[int(len(a) / 2):, :int(len(a) / 2)]

a22 = a[int(len(a) / 2):, int(len(a) / 2):]

b11 = b[:int(len(b) / 2), :int(len(b) / 2)]

b12 = b[:int(len(b) / 2), int(len(b) / 2):]

b21 = b[int(len(b) / 2):, :int(len(b) / 2)]

b22 = b[int(len(b) / 2):, int(len(b) / 2):]

p1 = strassen(a11, b12 - b22)

p2 = strassen(a11 + a12, b22)

p3 = strassen(a21 + a22, b11)

p4 = strassen(a22, b21 - b11)

p5 = strassen(a11 + a22, b11 + b22)

p6 = strassen(a12 - a22, b21 + b22)

p7 = strassen(a11 - a21, b11 + b12)

result = np.zeros((n, n))

result[:int(len(result) / 2), :int(len(result) / 2)] = p5 + p4 - p2 + p6

result[:int(len(result) / 2), int(len(result) / 2):] = p1 + p2

result[int(len(result) / 2):, :int(len(result) / 2)] = p3 + p4

result[int(len(result) / 2):, int(len(result) / 2):] = p5 + p1 - p3 - p7

return result

sizes = [2 \*\* i for i in range(2, 8)]

normal\_m\_times = []

divide\_and\_conquer\_times = []

strassen\_times = []

for size in sizes:

a = np.random.uniform(low=-100, high=100, size=(size, size))

b = np.random.uniform(low=-100, high=100, size=(size, size))

start\_time = 0

start\_time = time()

normal\_m(a, b)

normal\_m\_times.append(time() - start\_time)

start\_time = 0

start\_time = time()

divide\_and\_conquer(a, b)

divide\_and\_conquer\_times.append(time() - start\_time)

start\_time = 0

start\_time = time()

strassen(a, b)

strassen\_times.append(time() - start\_time)

df = pd.DataFrame({

'Normal m': normal\_m\_times,

'Divide and conquer': divide\_and\_conquer\_times,

'Strassen': strassen\_times},

index=sizes)

print(df.to\_string())

1. Результаты программы

Изображение выглядит как текст

Автоматически созданное описание

1. Вывод

В ходе выполнения лабораторной работы были изучены и реализованы алгоритмы умножения двух матриц. Было установлено, что алгоритм Штрассена работает быстрее, чем алгоритм полного перебора и метод «Разделяй и властвуй». Программы работают правильно, задача выполнена в полном объёме.