Лабораторная работа № 13

«Метод Штрассена»

Цель работы

1. Реализовать метод Штрассена.
2. Реализовать рекурсию через многопоточность.
3. Сравнить точность результата со стандартным алгоритмом умножения.
4. Построить на одном графике зависимость времени t (сек) умножения двух матриц размера N x N стандартным алгоритмом, методом Штрассена и методом Штрассена с многопоточностью от размера матрицы N.

```
import copy
import time
from multiprocessing.pool import ThreadPool
import matplotlib.pyplot as plt
import numpy as np



def fill_matrix(n, a, b):
    m = np.random.random((n, n)) * (b - a) + a
    return m


def split(matrix):
    row, col = matrix.shape
    row2, col2 = row // 2, col // 2
    return matrix[:row2, :col2], matrix[:row2, col2:], matrix[row2:, :col2], matrix


def strass(a, b, n_min):
    if len(a) <= n_min:
        return multiply(a, b)

    a11, a12, a21, a22 = split(a)
    b11, b12, b21, b22 = split(b)

    p1 = strass(a11 + a22, b11 + b22, n_min)
    p2 = strass(a21 + a22, b11, n_min)
    p3 = strass(a11, b12 - b22, n_min)
    p4 = strass(a22, b21 - b11, n_min)
    p5 = strass(a11 + a12, b22, n_min)
    p6 = strass(a11 - a21, b11 + b12, n_min)
    p7 = strass(a12 - a22, b21 + b22, n_min)

    c11 = p1 + p4 - p5 + p7
```

```
    c12 = p3 + p5
    c21 = p2 + p4
    c22 = p3 + p1 - p2 - p6

    a21 = np.vstack((np.hstack((c11, c12)), np.hstack((c21, c22))))

    return a21


def strass_thread(a, b, n_min):
    if len(a) <= n_min:
        return multiply(a, b)

    a11, a12, a21, a22 = split(a)
    b11, b12, b21, b22 = split(b)

    pool = ThreadPool(processes=7)

    p1 = pool.apply_async(strass, (a11 + a22, b11 + b22, n_min)).get()
    p2 = pool.apply_async(strass, (a21 + a22, b11, n_min)).get()
    p3 = pool.apply_async(strass, (a11, b12 - b22, n_min)).get()
    p4 = pool.apply_async(strass, (a22, b21 - b11, n_min)).get()
    p5 = pool.apply_async(strass, (a11 + a12, b22, n_min)).get()
    p6 = pool.apply_async(strass, (a11 - a21, b11 + b12, n_min)).get()
    p7 = pool.apply_async(strass, (a12 - a22, b21 + b22, n_min)).get()

    c11 = p1 + p4 - p5 + p7
    c12 = p3 + p5
    c21 = p2 + p4
    c22 = p3 + p1 - p2 - p6

    a21 = np.vstack((np.hstack((c11, c12)), np.hstack((c21, c22))))

    return a21


def multiply(a, b):
    c = np.matrix(np.zeros([a.shape[0], a.shape[1]]))
    for i in range(c.shape[0]):
        for j in range(c.shape[0]):
            for k in range(c.shape[0]):
                c[i, j] += a[i, k] * b[k, j]
    return c


def compare():
    i = 6
    power = []
    time_strass = []
    time_mul = []
    time_th = []
    while np.power(2, i) < np.power(2, 10):
        power.append(i)

        a = fill_matrix(np.power(2, i), 1, 10)
```

```
        b = fill_matrix(np.power(2, i), 1, 10)

        strass_time_now = time.time()
        strass_matrix = strass(copy.deepcopy(a), copy.deepcopy(b), 8)
        strass_time_after = time.time()

        print(f"strass time for {np.power(2, i)}: {strass_time_after - strass_time_

        mult_time_now = time.time()
        mult_matrix = multiply(copy.deepcopy(a), copy.deepcopy(b))
        mult_time_after = time.time()

        print(f"mult time for {np.power(2, i)}:   {mult_time_after - mult_time_now}

        th_time_now = time.time()
        th_matrix = strass_thread(a, b, 8)
        th_time_after = time.time()

        print(f"threading time for {np.power(2, i)}:   {th_time_after - th_time_now

        time_strass.append(strass_time_after - strass_time_now)
        time_mul.append(mult_time_after - mult_time_now)
        time_th.append(th_time_after - th_time_now)

        print()
        print(f"Соотношение практическое: {time_strass[-1] / time_mul[-1]}")
        print(f"Соотношение формульное:   {7 / 8}")
        print()
        i += 1

    plt.plot(power, time_strass, 'red')
    plt.plot(power, time_mul, 'blue')
    plt.plot(power, time_th, 'green')
    plt.show()

if __name__ == '__main__':
    compare()
```

```
strass time for 64: 0.48758578300476074
mult time for 64:   0.607797384262085
threading time for 64:   0.3824787139892578

Соотношение практическое: 0.8022176396772902
Соотношение формульное:   0.875

strass time for 128: 2.765552043914795
mult time for 128:   4.403645992279053
threading time for 128:   2.7392358779907227

Соотношение практическое: 0.6280141611663742
Соотношение формульное:   0.875

strass time for 256: 18.57403826713562
mult time for 256:   19.9533793926239
threading time for 256:   10.872419834136963

Соотношение практическое: 0.9308718038009052
Соотношение формульное:   0.875

strass time for 512: 78.50583362579346
mult time for 512:   164.5295684337616
threading time for 512:   76.61525321006775

Соотношение практическое: 0.47715334315363095
Соотношение формульное:   0.875
```