**Київський національний університет імені Тараса Шевченка**

**факультет комп’ютерних наук та кібернетики**

**кафедра інформаційних систем**

**Лабораторна робота № 5**

**“Реалізація алгоритму Штрассена”**

**Виконав студент 2 курсу**

**групи К-28**

**Гоженков Єгор Геннадійович**

**2018**

1. Зміст завдання

Реалізувати алгоритм Штрассена для множення матриць. На практиці алгоритм починає застосовуватися для матриць такого розміру, коли з'являється виграш порівняно з класичним способом на основі означення, який використовується для матриць меншого розміру. Спробуйте експериментально визначити цю "точку перетину" для свого комп'ютера.

1. Опис алгоритму вирішення та основні модулі програми

Реалізацію алгоритму Штрассена подано на мові С++ без використання класів та додаткових можливостей бібліотек для роботи з матрицями.

Використано наступні допоміжні функції:

* set(mat A, corners a, datatype k)
* randk(mat A, corners a, double l, double h)
* print(mat A, corners a, const char \*name)
* add(mat A, mat B, mat C, corners a, corners b, corners c)
* sub(mat A, mat B, mat C, corners a, corners b, corners c)
* find\_corner(corners a, int i, int j, corners \*b)

Усі вищезгадані процедури не повертають значення, оскільки працюють з вказівниками і змінюють одразу структури, передані в якості формальних параметрів.

Процедура set приймає на вхід матрицю А з координатами кутів, що зберігаються у структурі corners = {int ra, rb, ca, cb;}, та значення відповідного типу, яким буде заповнено матрицю ( в нашій програмі використовується для обнулення матриць ).

Процедура randk також приймає на вхід матрицю А з кутами а, але заповнює її випадковими числами з діапазону [l...h). У нашій програмі основна функція для ініціалізації початкових матриць.

Функція print здійснює форматований вивід на вікно консолі матриці А з кутами а та формальною назвою name (для зручності користувача).

Функції add та sub відповідно здійснюють додавання та віднімання матриць А та В з кутами а та b відповідно, записуючи результат у матрицю С з кутами с.

Процедура find\_corner використовується для коректного визначення координат кутів b з матриці з кутами а за допомогою індексів визначення чверті i та j, де і,j = 0,0 — верхня ліва чверть, 0,1 — верхня права чверть, 1,0 та 1,1 — нижня ліва та права чверті відповідно.

Основна процедура mul(mat A, mat B, mat C, corners a, corners b, corners c) власне і є реалізацією алгоритму Штрассена, що множить матриці А та В і записує результат у матрицю С

3. Складність

Складність алгоритму: О(n^(log7)), де логарифм двійковий. Складність пояснюється рекурсивним обчисленням семи коефіцієнтів Pi, 0 < i < 8.

4. Експеримент з обчислення «точки перетину»

Виграш у ефективності порівняно з алгоритмом множення матриць за O(n^3) досягається на матрицях розміру не менше ніж 64х64. При такому розмірі виграш майже непомітний, а надалі зростає, оскільки двійковий логарифм числа 7 менший за 3.

Продемонструємо зміну у відносній ефективності експериментальними даними:

n = 16:

Тут звичайний алгоритм працює настільки швидко, що навіть не вдається зафіксувати час його роботи, в той час як алгоритм Штрассена за рахунок багатьох проміжних обчислень виконується досить довго.

n = 32:

Результат той самий, алгоритм Штрассена виконується ще довше, тоді як звичайний алгоритм досі дуже швидкий.

Виграш звичайного алгоритму тут: 0.7 секунди.

n = 64:

З’являється невеличкий виграш алгоритму Штрассена порівняно зі звичайним алгоритмом: 0.197 секунди.

Тут - 0.313 секунди:

Перевіримо тепер для n = 128:

Виграш вже зростає: близько 2х секунд.

Отже, “точка перетину” для мого комп’ютера: n = 64, кількість елементів >4000.

Код для обрахунку часу:

// обчислення часу виконання алгоритму Штрассена

clock\_t start = clock();

set(C, ci, 0);

mul(A, B, C, ai, bi, ci);

clock\_t end = clock();

print(C, ci, "C");

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("Strassen's matrix multiplication algotithm time: %f seconds\n", seconds);

// обчислення часу виконання звичайного алгоритму

start = clock();

simple\_mul(A, B, C);

end = clock();

print(C, ci, "C");

seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

Використана література:

Стаття “Алгоритм Штрассена”, Вікіпедія: https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC\_%D0%A8%D1%82%D1%80%D0%B0%D1%81%D1%81%D0%B5%D0%BD%D0%B0