Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра інформаційних систем

**Алгоритми та складність**

**Завдання №5**

“алгоритм Штрассена для множення матриць ”

Виконав:

студент групи К-29

Печкуров Віталій Михайлович

**Київ-2018**

**Умова завдання:**

Реалізуйте 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.

**Вирішення завдання:**

Найбільша витратна за часом операція - множення. Тобто щоб прискорити процес, потрібно скоротити число множень. Штрассен придумав алгоритм, який дозволяє уникнути одного множення. Чудово, що формули не вимагають комутативності додавання, тобто вони можуть бути використані і для матриць, що дозволяє застосовувати алгоритм рекурсивно.

Алгоритм працює з квадратними матрицями розміру 2n. Зауважимо, що для множення матриць такого розміру потрібно 23n множень. Якщо ж матриці не квадратні, то можна розширити їх до потрібного розміру, заповнивши відсутні рядки і стовпці нулями. Припустимо, є дві матриці A і B розміру 2n, які потрібно перемножити: C = AB. Розіб'ємо кожну матрицю на чотири підматриці:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A2;1 | A2;2 | B2;1 | B2;2 |  | C2;1 | C2;2 |  |
| A1;1 | A1;2 | B1;1 | B1;2 | = | C1;1 | C1;2 |  |

Тоді матриці Ai; j і Bi; j мають розмір 2n-1. А елементи матриці можна обчислити таким чином:

C1;1= A1;1B1;1+ A1;2B2;1

C1;2= A1;1B1;2+ A1;2B2;2

C2;1= A2;1B1;1+ A2;2B2;1

C2;2= A2;1B1;2+ A2;2B2;2

Але в цьому випадку ми не зменшили кількість множень, тому що кожне рівняння вимагає двох множень матриць розміру 2n-1. А на кожному кроці рекурсії множень вийде вісім.

Штрассен ввів сім нових елементів, які обчислюються за допомогою одного множення, а матриця C виходить з них додаванням або відніманням. Визначимо ці сім елементів:

M1= (A1;1+ A2;2)(B1;1+ B2;2)

M2= (A2;1+ A2;2)B1;1

M3= A1;1(B1;2B2;2)

M4= A2;2(B2;1B1;1)

M5= (A1;1+ A1;2)B2;2

M6= (A2;1A1;1)(B1;1+ B1;2)

M7= (A1;2A2;2)(B2;1+ B2;2)

Елементи матриць С вираховуються за формулами:

C1;1 = M1 + M4 M5 + M7

C1;2 = M3 + M5

C2;1 = M2 + M4

C2;2 = M1 M2 + M3 + M6

Тепер на кожному етапі рекурсії потрібно виконувати тільки сім множень замість восьми. Передбачається, що ітераційний процес триває 2n раз, до тих пір, поки матриці Ci; j НЕ виродяться в числа. Зауважимо, що для маленьких матриць алгоритм втрачає свою ефективність. Також цей алгоритм не має великої стійкості, тобто для речових матриць, в разі, якщо потрібна дуже висока точність, то не варто множити по Штрассену. Цей істотний мінус легко пояснити великою кількість додавань і віднімань. Але для цілочисельних матриць, звичайно, проблеми не виникає.

Рекурсивний алгоритм Штрассена примножує дві матриці за час (n2: 81), тоді як просте множення справляється з цим завданням за (n3). Для матриць великого розміру (починаючи приблизно з 2000) ця різниця істотна.

**Модулі програми**

* void show\_result(int n, int C[][N])

Функція виводить результат який вийшов після множення

* void Matrix\_Multiplication(int A[][N], int B[][N], int C[][N])

Функція, де реалізоване множення матриць за нормальним алгоритмом

* void Matrix\_Sum(int n, int A[][N], int B[][N], int C[][N])

Функція, де реалізована сума матриць

* void Matrix\_Sub(int n, int A[][N], int B[][N], int C[][N])

Функція, де реалізоване віднімання матриць

* void Strassen(int n, int A[][N], int B[][N], int C[][N])

Функція, де реалізован алгоритм множення матриць за алгоритмом Штрассена