Київський національний університет імені Тараса Шевченка
Факультет комп’ютерних наук та кібернетики

Алгоритми та складність

Завдання № 5

Звіт

Виконав:

студент групи К-29
Грищенко Юрій Анатолійович

Київ-2019

Умова задачі

Реалізуйте алгоритм Штрассена для множення матриць.

Опис алгоритму

Звичайний алгоритм для множення квадратних матриць розміром nxn, за визначенням

$$c_{i,k} = \sum a_{i,j} b_{j,k}$$

має складність O(n^3) (оскільки на виході маємо n^2 елментів, для обчислення кожного з них виконуємо цикл для j від 1 до n).

Алгоритм Штрассена дозволяє виконати множення за $\Theta(n^{\log_2 7}) = O(n^{2.81})$ операцій.

Нехай нам потрібно знайти матрицю C=AB, і розмір матриць A та B є степінню двійки.

В такому разі можна розбити A, B та C на рівні між собою за розміром «під-матриці»:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

Тоді

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{1,2} &= \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2} \\
\mathbf{C}_{2,1} &= \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{2,2} &= \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}
\end{aligned}$$

Таким чином ми ще не зменшили кількість операцій. Як і при звичайному методі, все одно ми виконаємо 8 множень.

Але можемо розглянути наступні елементи:

$$\begin{aligned}
\mathbf{P}_1 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2}) \\
\mathbf{P}_2 &:= (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1} \\
\mathbf{P}_3 &:= \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2}) \\
\mathbf{P}_4 &:= \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1}) \\
\mathbf{P}_5 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2} \\
\mathbf{P}_6 &:= (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2}) \\
\mathbf{P}_7 &:= (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})
\end{aligned}$$

з яких виразимо C_{ij}.

$$\begin{aligned} \mathbf{C}_{1,1} &= \mathbf{P}_1 + \mathbf{P}_4 - \mathbf{P}_5 + \mathbf{P}_7 \\ \mathbf{C}_{1,2} &= \mathbf{P}_3 + \mathbf{P}_5 \\ \mathbf{C}_{2,1} &= \mathbf{P}_2 + \mathbf{P}_4 \\ \mathbf{C}_{2,2} &= \mathbf{P}_1 - \mathbf{P}_2 + \mathbf{P}_3 + \mathbf{P}_6 \end{aligned}$$

Таким чином замість 8 множень ми виконаємо 7, за рахунок чого і знижується складність алгоритму. Але при цьому ми виконали досить велику кількість додавань та віднімань, тому цей алгроитм є повільнішим за стандартний при маленьких матрицях. На моєму ноутбуці алгоритм Штрассена є швидшим за звичайний при матрицях розміром 256 та вище.

Додаткові зауваження:

1. Під час рекурсії, коли ми доходимо до матриць маленького розміру (як 128 чи нижче), варто переходити на стандартний алгоритм. Таким чином збільшимо ефективність алгоритму. (в моїй імплементації цього не зроблено).
2. Щоб можна було працювати з матрицями довільного розміру, а не тільки 2^n, можна додавати нулі знизу та зліва від матриці на кожному кроці так, щоб розмір був завжди парним. (в моїй імплементації цього також не зроблено)

Інтерфейс користувача:

Програма виконується в режимі демонстрації, тобто випадково генеруються дві матриці, виводяться на екран; виводиться результат стандартного множення і множення за Штрассеном.

```
Matrix 1:
-5 -4 3 0
0 -3 -5 2
2 5 -1 0
4 -5 -5 0
Matrix 2:
2 -5 -1 -5
-1 2 1 5
4 0 -4 2
-1 2 5 3
Standard multiplication:
6 17 -11 11
-19 -2 27 -19
```

```
-5 0 7 13
-7 -30 11 -55
Strassen multiplication:
6 17 -11 11
-19 -2 27 -19
-5 0 7 13
-7 -30 11 -55
```

Список використаних джерел:

https://en.wikipedia.org/wiki/Strassen_algorithm
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