Лекция 4

Параллельные алгоритмы для вычисления определителя и арифметических операций

(Конспект: Е. Петренко)

4.1 Вычисление определителя

Пусть A — матрица размерами $n \times n$. Требуется вычислить определитель этой матрицы. Алгоритм будет использовать $\mathrm{poly}(\log n)$ процессоров.

Алгоритм будет одновременно вычислять определитель и искать обратную матрицу. Заметим, что обратная матрица к матрице B выглядит следующим образом:

$$B^{-1} = \frac{1}{\det B} \left(\begin{array}{c|c} B^{11} & \cdots \\ \hline \vdots & \ddots \end{array} \right).$$

Пусть $B^{(i)}$ — правый нижний $i \times i$ угол матрицы B. Тогда

$$(B^{-1})_{11} = \frac{\det B^{(n-1)}}{\det B^{(n)}},$$

а в общем случае

$$\left(B^{(i)^{-1}}\right)_{11} = \frac{\det B^{(i-1)}}{\det B^{(i)}}.$$

Положим $\det Z = 1$, где матрица Z — размерности 0×0. Имеем

$$\det B = \det B^{(n)} = \frac{\det B^{(n-1)}}{\left((B^{(n)})^{-1}\right)_{11}} = \ldots = \frac{1}{\prod\limits_{i=1}^{n} \left(B^{(i)^{-1}}\right)_{11}}.$$

При помощи этого тождества будем искать определитель символьной матрицы $E-xA$ (и из него извлечем $\det A$). Поскольку $\deg\det(E-xA) \leq n$, все вычисления можно проводить по $\bmod\ x^{n+1}$:

$$(E - xA^{(k)})^{-1} = \sum_{i=0}^{k-1} (xA)^i \pmod{x^{n+1}}$$

Возводить в степень и складывать матрицы можно при помощи двоичного дерева. Сложение многочленов производим поэлементно, умножение многочленов не используется: $(xA)^i = x^i A^i$.

Так мы вычислим $\prod\limits_{i=0}^{n} (B^{(i)^{-1}})_{11}$. Для получения $\det(E - xA)$ остается обратить полученный многочлен $p(x)$. Пусть

$$p(x) = \mathrm{const} \cdot (1 - xq(x)).$$

Тогда имеет место равенство

$$(1 - xq(x))^{-1} = \sum_{i=0}^{n} (xq(x))^i \pmod{x^{n+1}} \tag{4.1}$$

Для проверки этого утверждения достаточно раскрыть сумму. (Если же $p(x)$ не представим в таком виде, то $\exists k :\ p(x) = x^k \cdot \mathrm{const}(1 - xq(x))$; тогда при помощи 4.1 мы можем обратить все, кроме x^k.)

Итак, мы нашли $\det(E - xA)$ (либо нечто, что при домножении на x^k дает $x^k \cdot \det(E - xA)$). Коэффициент при x^n этого многочлена и есть $\det A$:

$$\lim_{n\to+\infty} \frac{\det(E - xA)}{(-x)^n} = \lim_{n\to+\infty} \det\left(\frac{E}{(-x)^n} + A\right) = \det A;$$

(аналогично, если вычислили «с точностью до x^k»:

$$\lim_{x\to+\infty} \frac{\det(E - xA)x^k}{(-x)^n x^k} = \lim_{x\to+\infty} \det\left(\frac{E}{(-x)^n x^k} + A\right) = \det A).$$

Мы сделали это за логарифмическое время, если время измерять в арифметических операциях. Осталось научиться параллелизовать арифметические операции. (Заметим, кстати, что коэффициенты наших многочленов были порядка $O(b) \cdot 2^{O(n)}$, где b — количество битов в исходных коэффициентах.)

4.2 Вычисление суммы и произведения целых чисел

Лемма 4.1. *Пусть $\diamond$ — ассоциативная операция, тогда одновременное вычисление всех $(a_1 \diamond a_2 \cdots \diamond a_i)$ для $i = 1, 2, \ldots, n$ можно произвести за логарифмическое время на $O(n/\log n)$ процессорах.*

Доказательство.

ПРОБЕЛ В КОНСПЕКТЕ.

□

Disclaimer: *все написанное ниже мне проверить не удалось, поскольку не удалось понять конспектирующего. Текст приводится «as is». Если появится альтернативный конспект, я готов поместить сюда его.*

– – Э.А.

4.2.1 Вычисление суммы

$$a_i + b_i + c_i \to d_i, c_{i+1}$$

Линейно, хотим сделать за log.

$$p_i = a_i \vee b_i // g_i = a_i \wedge b_i //$$

Операция сложения с переносом для двоичных чисел. g_i — перенос будет, p_i — перенос перейдет. По 4.2.1 получим

$$c_i = g_i \vee (p_i \wedge c_{i-1})$$

Определим операцию:

$$(x, y) \diamond (x', y') = (x' \vee (y' \wedge x), y' \wedge y)$$

Проверим ассоциативность.

$$(c_i, -) = (c_{i-1}, -) \diamond (g_i, p_i) // \Rightarrow //(\circ, \circ) \diamond (g_1, p_1) \diamond \cdots \diamond (g_i, p_i)$$

$\diamond$ — ассоциативная операция. Следовательно, можно вычислить за log. Для вычисления суммы всего числа можно выполнить этот процесс в каждом разряде. Построили алгоритм сложения за log шагов.

4.2.2 Вычисление произведения

Умножение — это сложения и сдвиги. Алгоритм умножения «в столбик». Для каждого разряда вычисляем отдельно. Нужно вычислить сумму n чисел из $2n$ битов. Каждый бит $c_{ij} = a_i \wedge b_i$ или нуль.

Просто сложить все числа. Сложность будет $\log^2$. Есть другое решение: рассмотрим троичное дерево

$$\oplus(a, b, c) = (e, f)$$

Высота дерева $\log_{\frac{3}{2}}$

$$\begin{aligned} x_i, y_i, z_i : \qquad & (x_i, y_i, z_i) \to (u_i, v_i) \\ & v_i = x_i + y_i + z_i \pmod 2 \\ & u_i = x_i + y_i + z_i \div 2 \end{aligned}$$

$a_i + b_i + c_i = \overline{e_i f_i}$ — просто побитовое сложение. Результат суммы трех двоичных чисел влезет в два двоичных числа. Когда остаются только 2 числа, применим обычное сложение.