Лекция 8

Проверка простоты числа. Рисование планарного графа. Параллельный алгоритм для задачи о максимальном независимом множестве.

(Конспект: А. Кожевников)

8.1 Проверка простоты числа

Для начала вспомним несколько определений. Для начала символ Лежандра:

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & , \text{ уравнение } x^2 \equiv a \pmod{p} \text{ имеет корни} \\ -1 & , \text{ в противном случае} \end{cases}, \qquad (8.1)$$

где p — простое, $a \neq 0$.

Упражнение 8.1. Вспомнить формулу вычисления 8.1.

Теперь символ Якоби:

$$\left(\frac{a}{N}\right) = \prod_i \left(\frac{a}{p_i}\right),$$

если $N = p_1 \cdots p_k$ — разложение N на простые множители. Некоторые свойства:

$$\begin{aligned}\left(\frac{a}{N}\right) &= (-1)^{\frac{N-1}{2}\frac{a-1}{2}}\left(\frac{N}{a}\right),\\ \left(\frac{1}{p}\right) &= 1.\end{aligned}$$

Упражнение 8.2. Вспомнить, как вычислять $\left(\frac{2}{p}\right) = (-1)^?$.

Формулировка алгоритма.

Алгоритм 8.1. Нужно проверить, простое ли число N.

$$M \leftarrow random[2..N-1] \tag{8.2}$$
$$\text{if } (M, n) \neq 1 \ \text{ then } \ \text{answer } \ \text{“composite”} \tag{8.3}$$
$$\text{else } \ \text{if } \left(\frac{M}{N}\right) \not\equiv M^{\frac{N-1}{2}} \pmod N \tag{8.4}$$
$$\text{then answer “composite”} \tag{8.5}$$
$$\text{else answer “prime” (тут алгоритм может ошибиться)} \tag{8.6}$$

□

Корректность шага (8.4) доказывает следующая лемма:

Лемма 8.1. *Если для всех M, таких что $(M, N) = 1$, выполняется $\left(\frac{M}{N}\right) \equiv M^{\frac{N-1}{2}} \pmod N$, то N — простое.*

Доказательство. Будем доказывать от противного:

1. Пусть N не содержит квадратов: $N = p_1 \cdots p_k$, $p_i \neq p_j \in \mathbb{P}$. Фиксируем r такое, что $\left(\frac{r}{p_1}\right) = -1$ (такое есть: пересчитаем все квадраты mod p_i ...). По китайской теореме об остатках можно выбрать такое M, что

$$\begin{aligned}M &\equiv r \pmod{p_1},\\ M &\equiv 1 \pmod{p_i} \quad \text{при } i \neq 1.\end{aligned}$$

С одной стороны,

$$\left(\frac{M}{N}\right) = \left(\frac{M}{p_1}\right) \cdot \prod_{i \neq 1}\left(\frac{M}{p_i}\right) = -1 \text{ (в том числе и mod } p_2).$$

С другой стороны,

$$M^{\frac{N-1}{2}} \equiv 1 \pmod{p_2}.$$

Противоречие.

2. Пусть N содержит квадраты: $N = p^2 n$, $p \in P$. Пусть r — первообразный корень по модулю p^2. По предположению,

$$r^{N-1} \equiv (r^{(N-1)/2})^2 \equiv \left(\frac{r}{N}\right)^2 \equiv 1 \pmod{N}$$

Т.е., одновременно $N - 1 \vdots p(p-1)$ и $N \vdots p$, т.е., два последовательных числа делятся на p. Противоречие.

$\square$

Лемма 8.2. *Если $N \notin \mathbb{P}$, то для более чем половины всех M, взаимно простых с N, $\left(\frac{M}{N}\right) \neq M^{\frac{N-1}{2}} \pmod{N}$.*

Доказательство. По лемме 8.1 существует такое число a, что $\left(\frac{a}{N}\right) \not\equiv a^{\frac{N-1}{2}} \pmod{N}$. Пусть для $b_1,\ b_2, \dots,\ b_k$ выполнено равенство $\left(\frac{M}{N}\right) \equiv M^{\frac{N-1}{2}} \pmod{N}$.

Рассмотрим $ab_1,\ ab_2, \dots,\ ab_k \pmod{N}$. Они все различны, так как если $ab_i \equiv ab_j \pmod{N}$, то $b_i \equiv b_j \pmod{N}$ (ведь $(a, N) = 1$). Они также отличны от b_i:

$$\left(\frac{ab_i}{N}\right) = \left(\frac{a}{N}\right)\left(\frac{b_i}{N}\right) = \left(\frac{a}{N}\right) \cdot b_i^{\frac{N-1}{2}} \neq (ab_i)^{\frac{N-1}{2}}.$$

$\square$

Тем самым, вероятность ошибки нашего алгоритма не превосходит 1/2.

8.2 Рисование планарного графа

Мы будем рассматривать двусвязные графы, то есть такие, что в них не существует двух вершин, удаление которых ведет к появлению двух компонент связности.

(Заметим, что если граф не таков, то его можно разбить на компоненты, нарисовать их по отдельность и соединить рисунки; чтобы отправить точку соединения на границу области, занимаемой графом, рисуем граф на сфере, после чего раскрываем сферу с дыркой рядом с вершиной.)

Определение 8.1. Окрестность компоненты S графа G это $\Gamma(S) = \{v \in V_G | \exists e \in E : e = (v, s), s \in S\}$.

Алгоритм 8.2.

1. Взять какой-то цикл и нарисовать его.

2. Если существует компонента, согласованная лишь с одной клеткой, то взять путь в компоненте и вставить путь в эту клетку (удалив путь из компоненты и разбив компоненту на несколько, если она развалилась).

3. Если все компоненты согласованы с несколькими клетками, то взять любую компоненту и вставить путь в клетку (удалив путь ...).

□

Задача 8.1. Придумать незначительное изменение алгоритма, работающее $O(n^2)$ для любого графа, а не только для планарного.

Корректность алгоритма утверждается в следующей лемме.

Лемма 8.3. *Пусть в какой-то моменент что-то уже нарисовано и алгоритм находится в шаге 3. Компонента C согласована с клетками K_1 и K_2. Если C можно вложить в K_1 (и успешно дорисовать граф до конца), то ее можно вложить и в K_2 (и успешно дорисовать).*

Доказательство. Поменяем все компоненты, согласованные с K_1 и K_2, местами. Перебором случаев проверим, что конфликтов возникнуть не должно (разобъем границу K_1 и K_2 на участки, принадлежащие только K_1, только K_2, либо им вместе; разберем случаи, когда конфликтующий путь начинается на одном из типов участков, а заканчивается на другом или том же самом). □

Замечание 8.1. Существует алгоритм, позволяющий нарисовать планарный граф за линейное время отрезками прямых.

8.3 Параллельный алгоритм для задачи о максимальном независимом множестве

Замечание 8.2. Пусть $f(n)$ процессоров могут за время $t(n)$ решить некоторую задачу. Тогда $g(n) < f(n)$ процессоров могут решить ее за время $t(n) \cdot \frac{f(n)}{g(n)}$. Для $g(n) > f(n)$ это неверно, поэтому представляют интерес параллельные алгоритмы, использующие как можно больше процессоров (при той же самой суммарной работе $t(n)f(n)$.

Определение 8.2. *Независимое множество* — это множество вершин графа, между которыми нет ребер.

Следующий параллельный алгоритм находит максимальное (*по включению*) независимое множество за время $O(\log^2 |\text{длина входа}|)$.

Алгоритм 8.3.

Пусть дан граф $G = (V, E)$, строим S следующим образом:

1. Для всех $v \in V$ таких, что $\deg(v) = 0$, $S := S \cup v$.
2. Все другие вершины метим с вероятностью $\frac{1}{2\deg(v)}$.
3. Для всех ребер $e \in E$, если оба конца помечены, то убрать пометку с вершины меньшей степени.
4. $S := S \cup \{\text{помеченные вершины}\}$.
5. $G := G \setminus (\{\text{помеченные вершины}\} \cup \{\text{окрестности помеченных вершин}\})$.
6. Повторять, пока граф непуст.

□

Замечание 8.3. Алгоритм корректный, каждый его шаг занимает время не более $O(\log |E|)$ и требует не более $|E|$ процессоров.

Определение 8.3. v — *хорошая*, если не менее $\frac{deg(v)}{3}$ ее соседей имеет степень не более чем $\deg(v)$.

Лемма 8.4. *$P\{$хотя бы одна соседка хорошей вершины v помечена на шаге 2$\} \geq 1 - e^{-\frac{1}{6}}$.*

Доказательство. $P\{\text{не так}\} = \prod_{u \in F(v), \deg(u) \leq \deg(v)} P\{u \text{ — не помечены}\} \leq (1 - \frac{1}{2\deg(v)})^{\deg(v)/3} \leq e^{-1/6}$. □

Лемма 8.5. *Вероятность снятия пометки не превосходит $\frac{1}{2}$.*

Доказательство. Эта вероятность для вершины v не превосходит

$$\text{кол-во соседок } v \text{ не меньшей степени} \cdot P\{\text{отметить такую соседку}\}$$
$$\leq \deg(v) \cdot \frac{1}{2\deg(v)} = \frac{1}{2}.$$

□

Следствие 8.1. *Вероятность того, что хотя бы одна соседка вершины v включена в S на шаге 4 — не менее* $\frac{1-e^{1/6}}{2}$.

Определение 8.4. Ребро e хорошее, если хотя бы один из концов хороший.

Определение 8.5. $e(S,T) = \{$количество ребер из S в $T\}$.

Лемма 8.6. *Хороших ребер — не менее* $\frac{|E|}{2}$.

Доказательство. Развернем ребра так, что бы они были направлены из меньшего ребра в большее; пусть d_i и d_o — входная и выходная степени, соответственно. Итак, v — плохое ребро, значит $d_o(v) - d_i(v) \geq \frac{\deg(v)}{3}$ (по определению хорошей вершины). Пусть V_G — множество хороших вершин, V_B — множество плохих вершин, $e(V_1, V_2)$ — множество всех ребер из вершин множества V_1 в вершины множества V_2.

$$\begin{aligned}
& 2 \cdot e(V_B, V_B) + e(V_B, V_G) + e(V_G, V_B) = \sum_{v \text{ — плохая}} \deg(v) && \leq \\
\leq\ & 3 \cdot \sum_{v \text{ — хорошая}} (d_o(v) - d_i(v)) = 3 \cdot \sum_{v \textit{ is good}} (d_o(v) - d_i(v)) && = \\
=\ & 3((e(V_B, V_G) + e(V_G, V_G)) - (e(V_G, V_B) + e(V_G, V_G))) && = \\
=\ & 3(e(V_B, V_G) - e(V_G, V_B)) \leq 3(e(V_B, V_G) + e(V_G, V_B)),
\end{aligned}$$

т.е.

$$e(V_B, V_B) \leq e(V_B, V_G) + e(V_G, V_B).$$

□

Отсюда следует основная теорема (из которой непосредственно следует оценка на время работы алгоритма, ибо по ней количество итераций не превосходит $O(\log |E|)$):

Теорема 8.1. *На каждой итерации исчезает в среднем* $\frac{1-e^{1/G}}{4}|E|$ *ребер.*