Решето Эратосфена, время работы (б/д)

Лесников Юрий, ceagest

1 Решето Эратосфена

Для составления таблицы простых чисел, не превосходящих данного целого N, существует способ, имеющий название «решето Эратосфена».

1. Выписываем числа $1, 2, \ldots, N$.
2. Первое, большее 1, число этого ряда есть 2. Оно простое.
3. Вычёркиваем из ряда $1, 2, \ldots, N$ все числа, кратные 2 и не равные 2.
4. Теперь первое не вычеркнутое число есть 3. Оно не делится на 2 и является простым.
5. Вычёркиваем из ряда $1, 2, \ldots, N$ все числа, кратные 3 и не равные 3.
6. и т.д.

Algorithm 1 Решето Эратосфена

```
function Primes(n)
bool[n] primes ← {true, true, ..., true}
primes[0] ← false
primes[1] ← false
for i ∈ {2 ... n − 1} do
    if primes[i] then
        for j ← i²; j ≤ n; j ← j + i do
            primes[j] ← false
        end for
    end if
end for
return primes
```

1.1 Корректность

Когда указанным способом будут вычеркнуты все числа, кратные простым, которые меньше простого p, то все не вычеркнутые, меньшие p^2, будут простыми. Действительно, всякое составное a, меньшее p^2, было вычеркнуто как кратное своему наименьшему простому делителю d такому, что: $d \leq \sqrt{a} < p$.

Замечание 1.1. *При вычёркивании кратных простому p, это вычёркивание следует начинать с p^2.*

Замечание 1.2. *Составление таблицы простых чисел, не превосходящих N, будет закончено, когда вычеркнуты все составные, кратные простым, не превосходящим $\sqrt{N}$.*

1.2 Асимптотика

Теорема 1.1. *Сложность алгоритма "решето Эратосфена" составляет $\mathcal{O}(n \log \log n)$*

2 Линейное решето Эратосфена

2.1 Идея оптимизации

Основная проблема решета Эратосфена состоит в том, что некоторые числа мы будем помечать как составные несколько раз — столько, сколько у них различных простых делителей. Чтобы достичь линейного времени работы, нам нужно придумать способ, как рассматривать все составные числа ровно один раз.

Обозначим за $d(k)$ минимальный простой делитель числа $k \in \mathbb{N}$ и заметим следующее: у составного числа k есть единственное представление $k = d(k) \cdot r$, где $r \in \mathbb{N}$, и при этом у числа r нет простых делителей меньше $d(k)$.

Идея оптимизации состоит в том, чтобы перебирать этот r, и для каждого перебирать только нужные множители — а именно, все от 2 до $d(r)$ включительно.

2.2 Алгоритм

Немного обобщим задачу — теперь мы хотим посчитать для каждого числа k на отрезке $[2, n]$, его минимальный простой делитель d_k, а не только определить его простоту.

Изначально массив d заполним нулями, что означает, что все числа предполагаются простыми. В ходе работы алгоритма этот массив будет постепенно заполняться. Помимо этого, будем поддерживать список p всех найденных на текущий момент простых чисел.

Теперь будем перебирать число k от 2 до n. Если это число простое, то есть $d_k = 0$, то присвоим $d_k = k$ и добавим k в список p.

Дальше, вне зависимости от простоты k, начнём процесс расстановки значений в массиве d – переберем найденные простые числа p_i, не превосходящие d_k, и сделаем присвоение $d_{k \cdot p_i} = p_i$.

Реализация линейного решета Эратосфена на Java

```java
public class Main {
    public static void main(String[] args) {
        final int n = 1000000;
        int[] d = new int[n + 1];
        List<Integer> p = new ArrayList<>();
        for (int k = 2; k <= n; k++) {
            if (d[k] == 0) {
                d[k] = k;
                p.add(k);
            }
            for (int x : p) {
                if (x > d[k] || x * k > n) {
                    break;
                }
                d[k * x] = x;
            }
        }
    }
}
```

2.3 Асимптотика и корректность

- Согласно ОТА, каждое составное натуральное число имеет единственное представление вида $d(k) \cdot r$.
- Алгоритм же для каждого числа смотрит, на какие простые числа его можно домножить, чтобы получить составное число. При этом эти простые числа являются наименьшими простыми числами в разложениях полученных чисел, так что каждое составное число будет тронуто только один раз в силу единственности разложения.
- Время работы такого решета — $\mathcal{O}(n)$.