Решето Эратосфена

1 Алгоритм

Дано натуральное число n. Необходимо найти все простые числа на отрезке $[1; n]$. Идея алгоритма в следующем: запишем ряд чисел $1..n$ и будем вычеркивать из него все составные числа. Единицу можно сразу вычеркнуть, т.к. это не простое число. Далее вычеркнем все числа делящиеся на 2, кроме самого числа 2. Потом все числа, делящиеся на 3, кроме самого числа 3 и т.д. В конечном итоге в ряде останутся только простые числа, которые нам и нужны. Предложенный выше алгоритм потребляет $O(n)$ памяти и имеет асимптотику $O(n \log \log n)$.

1.1 Время работы

Для каждого простого числа $p \leq n$ нужно совершить $\frac{n}{p}$ вычеркиваний. Т.е. надо оценить величину:

$$\sum_{\substack{p \leq n, \\ p \text{ is prime}}} \frac{n}{p} = n \cdot \sum_{\substack{p \leq n, \\ p \text{ is prime}}} \frac{1}{p}.$$

Из теоремы о распределении простых чисел следует, что количество простых чисел на отрезке $[1; n]$ примерно $\frac{n}{\ln n}$, и что k-ое простое число примерно равно $k \ln k$. Тогда:

$$sum \sum_{\substack{p \leq n, \\ p \text{ is prime}}} \frac{1}{p} \approx \frac{1}{2} + \sum_{k=2}^{\frac{n}{\ln n}} \frac{1}{k \ln k}.$$

Оценим сумму с помощью интеграла от той же функции по k на отрезке $[2; \frac{n}{\ln n}$ (по формуле прямоугольников):

$$\sum_{k=2}^{\frac{n}{\ln n}} \frac{1}{k \ln k} \approx \int_{2}^{\frac{n}{\ln n}} \frac{1}{k \ln k} dk.$$

Вычислим интеграл:

$$\int_{2}^{\frac{n}{\ln n}} \frac{1}{k \ln k} = \ln \ln \frac{n}{\ln n} - \ln \ln 2 = \ln(\ln n - \ln \ln n) - \ln \ln 2.$$

Вернемся к первоначальной сумме и уберем члены меньшего порядка и константы:

$$\sum_{\substack{p \leq n, \\ p \text{ is prime}}} \frac{n}{p} \approx \frac{n}{2} + n \ln(\ln n - \ln \ln n) - n \ln \ln 2 \approx n \ln \ln n.$$

2 Реализация

Алгоритм имеет довольно простую реализацию. Представленный ниже код можно оптимизировать как по времени, так и по памяти. Кроме того, есть линейный алгоритм решета, но в рамках курса он не рассматривается.

```
def eratosthenes_sieve(n):
    prime = [True] * (n + 1)
    prime[0] = prime[1] = False
    for i in range(2, n  + 1):
        if prime[i] and i * i <= n:
            for j in range(i * i, n + 1, i):
                prime[j] = False
    return [i for i, j in enumerate(prime) if j]
```