Moscow ACM ICPC Workshop 2015, Number Theory Lecture Notes

Mikhail Tikhomirov

November 20, 2015

1 Notation and preliminaries

An integer $p > 1$ is *prime* if its only divisors are 1 and p.

Fundamental theorem of arithmetic. Each positive integer n can be uniquely represented as $n = p_1^{\alpha_1} \dots p_k^{\alpha_k}$, where p_1, ldots, p_k are different primes.

The *Moebius function* $\mu(n)$ is defined as follows:

- if n is a product of k different primes (without squares), then $\mu(n) = (-1)^k$.
- else, $\mu(n) = 0$.

2 Eratosthene's sieve

The most well-known algorithm for finding prime numbers not exceeding n is the Eratosthene's sieve. Here is a simple implementation:

```
for all k from 2 to n:
    set isPrime[i] = true
for all k from 2 to n:
    if isPrime[k]:
        add k to the list of primes
        for all j = 2k, 3k, ...:
            set isPrime[j] = false
```

After running this pseudo-code, we will obtain a valid list of primes. The running time of the algorithm is roughly $\displaystyle\sum_{p \text{ is a prime } \leqslant n} \frac{n}{p} \sim n \log\log n$.

The sieve can be enhanced as follows:

```
for all k from 2 to n:
    set minimalDiv[i] = i
for all k from 2 to n:
    if minimalDiv[k] == k:
        add k to the list of primes
```

```
    for all primes p <= minimalDiv[k]:
        if p * k <= n:
            minimalDiv[p * k] = p
```

How is this an enhancement? First of all, the running time of the algorithm becomes $O(n)$, since for each k we will change `minimalDiv[k]` at most once. To see that, let $k = p_1^{\alpha_1} \dots p_m^{\alpha_m}$, and $p_1 < \dots < p_m$. Assuming that `minimalDiv` was computed correctly for all numbers less than k, we conclude that the only way to overwrite `minimalDiv[k]` is from $k' = k/p_1$ with $p = p_1$.

Moreover, the array `minimalDiv` provides us with information to compute many useful functions on all numbers from 1 to n, such as the number of divisors or Euler's totient φ function, since they can be found easily using factorization of the number.

3 Fast sums: first examples

3.1 Points under hyperbola

Count the number of integer pairs x, y such that $x, y > 0$ and $xy \leqslant n$.

Solution The number can be represented as the sum

$$\sum_{k=1}^{n} \left\lfloor \frac{n}{k} \right\rfloor$$

Fact 1. There are $O(\sqrt{n})$ different values of $\left\lfloor \frac{n}{k} \right\rfloor$ for integer k.

Proof. If $k \geqslant \sqrt{n}$, then $\left\lfloor \frac{n}{k} \right\rfloor \leqslant \sqrt{n}$. □

Fact 2. If $k \geqslant 1$, the number of x's satisfying $\left\lfloor \frac{n}{x} \right\rfloor = k$ is $\left\lfloor \frac{n}{k} \right\rfloor - \left\lfloor \frac{n}{k+1} \right\rfloor$.

Proof. The equation is equivalent to $k \leqslant \frac{n}{x} < k + 1$. □

Thus we can break the sum in two, for small and large values of k:

$$\sum_{k=1}^{\sim\sqrt{n}} \left\lfloor \frac{n}{k} \right\rfloor + \sum_{x=1}^{\sim\sqrt{n}} x \left(\left\lfloor \frac{n}{x} \right\rfloor - \left\lfloor \frac{n}{x+1} \right\rfloor \right)$$

We should take care to count each summand exactly once, especially for values of x and k around $\sqrt{n}$.

To sum up, we can compute this sum in $O(\sqrt{n})$ time.

3.2 Squarefree numbers

A number is *squarefree* if it's not divisible by a square of any number greater than 1. Find $sq(n)$ — the number of squarefree numbers not exceeding n.

Solution. We will use the inclusion-exclusion principle. We will start with n numbers, then subtract numbers divisible by $2^2 = 4$ and $3^2 = 9$, then add back numbers divisible by $6^2 = 36$ and so on. Since $\mu(n)$ are exactly the coefficients for inclusion-exclusion principle, we obtain the formula:

$$sq(n) = \sum_{k=1}^{\lfloor \sqrt{n} \rfloor} \mu(k) \left\lfloor \frac{n}{k^2} \right\rfloor$$

All $\mu(k)$ up to $\sqrt{n}$ can be directly obtained from the linear Eratosthene's sieve. Thus, this sum can be computed in $O(\sqrt{n})$.

4 Harder summation examples

4.1 Sum of $\varphi(k)$

Let us find $\Phi(n) = \sum\limits_{k=1}^{n} \varphi(k)$. We can find it easily in linear time using the Eratosthene's sieve, but we would like to do better.

Fact. $\sum\limits_{d|k} \varphi(d) = k$

Proof. Let us break all the numbers from 1 to k into groups with respect to $GCD(k, x)$. There are exactly $\varphi(k/d)$ numbers with $GCD(k, x) = d$, hence the formula.

□

Fact. $\sum\limits_{d=1}^{n} \Phi(\lfloor n/d \rfloor) = \dfrac{n(n+1)}{2}$

Proof. This is the previous statement after summation over k from 1 to n.

□

It follows that $\Phi(n) = \frac{n(n+1)}{2} - \sum\limits_{d=2}^{n} \Phi(\lfloor n/d \rfloor)$. Thus, if we know values of $\Phi(\lfloor n/d \rfloor)$ for all d, then we can find $\Phi(n)$. Note that the summation here can be performed in $O(\sqrt{n})$ similar to the above examples.

To improve the summation even more, let's find first K values of Φ explicitly using the sieve, and values of $\Phi(n/d)$ for d up to $\sim n/K$ using the above method. The complexity becomes

$$O(K + \sum_{j=1}^{n/K} \sqrt{n/j}) \sim O(k + \int_{1}^{n/K} \sqrt{n/x} dx) \sim O(K + n/\sqrt{K})$$

Choosing $K \sim n^{2/3}$, we obtain a method with complexity $O(n^{2/3})$.

4.2 Sum of $\mu(k)$

Let's find $M(n) = \sum\limits_{k=1}^{n} \mu(k)$, so called Mertens' function. Let $g(x) \equiv 1$ for all x. Then

$$M(n) = \sum_{k=1}^{n} \mu(k) g(\lfloor n/x \rfloor)$$

By the second Moebius' inversion formula,

$$1 = g(n) = \sum_{k=1}^{n} M(\lfloor n/k \rfloor)$$

Thus, we have the expression $M(n) = 1 - \sum_{k=2}^{n} M(\lfloor n/k \rfloor)$. We can use the method for finding $\Phi(n)$ here without much modification to find $M(n)$ in $O(n^{2/3})$.

4.3 Revisiting $\Phi(n)$

The Mertens' function proves useful in finding quite general sums involving $\mu(n)$, which arise naturally if we use inclusion-exclusion principle. As an illustration, consider $cp(n)$ — the number of pairs $1 \leqslant a \leqslant b \leqslant n$ such that $GCD(a, b) = 1$ (we already know that $cp(n) = \Phi(n)$, but here we will show another way of computing this value).

Let us use the inclusion-exclusion principle: take all pairs, subtract all pairs with common divisors 2 and 3, add back pairs with common divisor 6, and so on. We obtain the formula:

$$cp(n) = \sum_{d=1}^{n} \mu(d) \frac{\lfloor n/d \rfloor (\lfloor n/d \rfloor + 1)}{2}$$

Perform the $\sqrt{n}$-breaking of the sum:

$$cp(n) = \sum_{d=1}^{\sim\sqrt{n}} \mu(d) \frac{\lfloor n/d \rfloor (\lfloor n/d \rfloor + 1)}{2} + \sum_{k=1}^{\sqrt{n}} \frac{k(k+1)}{2} \left(M\left(\left\lfloor \frac{n}{k} \right\rfloor \right) - M\left(\left\lfloor \frac{n}{k+1} \right\rfloor \right) \right)$$

Note that the method of computing $M(n)$ also produces all values of $M(\lfloor n/d \rfloor)$, thus our problem can be solved directly using results of computing $M(n)$ as shown above. The complexity is still $O(n^{2/3})$.

5 Counting the primes

We would like to find $\pi(n)$ — the number of primes not exceeding n. Once again, we want to do better than the linear sieve approach.

Denote p_j the j-th prime number. Denote $dp_{n,j}$ the number of k such that $1 \leqslant k \leqslant n$, and all prime divisors of k are at least p_j (note that 1 is counted in all $dp_{n,j}$, since the set of its prime divisors is empty). $dp_{n,j}$ satisfy a simple recurrence:

- $dp_{n,1} = n$ (since $p_1 = 2$)
- $dp_{n,j} = dp_{n,j+1} + dp_{\lfloor n/p_j \rfloor, j}$, hence $dp_{n,j+1} = dp_{n,j} - dp_{\lfloor n/p_j \rfloor, j}$

Let p_k be the smallest prime greater than $\sqrt{n}$. Then $\pi(n) = dp_{n,k} + k - 1$ (by definition, the first summand accounts for all the primes not less than k).

If we evaluate the recurrence $dp_{n,k}$ straightforwardly, all the reachable states will be of the form $dp_{\lfloor n/i \rfloor, j}$. We can also note that if p_j and p_k are both greater than $\sqrt{n}$, then $dp_{n,j} + j = dp_{n,k} + k$. Thus, for each $\lfloor n/i \rfloor$ it makes sense to keep only $\sim \pi\sqrt{n/i}$ values of $dp_{\lfloor n/i \rfloor, j}$.

Instead of evaluating all DP states straightforwardly, we perform a two-step process:

- Choose K.
- Run recursive evaluation of $dp_{n,k}$. If we want to compute a state with $n < K$, memorize the query "count the numbers not exceeding n with all prime divisors at least k".
- Answer all the queries off-line: compute the sieve for numbers up to K, then sort all numbers by the smallest prime divisor. Now all queries can be answered using RSQ structure. Store all the answers globally.
- Run recurisive evaluation of $dp_{n,k}$ yet again. If we want to compute a state with $n < K$, then we must have preprocessed a query for this state, so take it from the global set of answers.

The performance of this approach relies heavily on Q — the number of queries we have to preprocess.

Statement. $Q = O(\frac{n}{\sqrt{K}\log n})$.

Proof. Each state we have to preprocess is obtained by following a $dp_{\lfloor n/p_j \rfloor, j}$ transition from some greater state. It follows that Q doesn't exceed the total number of states for $n > K$.

$$Q \leqslant \sum_{j=1}^{n/K} \pi(\sqrt{n/j}) \sim \sum_{j=1}^{n/K} \sqrt{n/j}/\log n \sim \frac{1}{\log n}\int_1^{n/K} \sqrt{n/x}dx \sim \frac{n}{\sqrt{K}\log n}$$

□

The preprocessing of Q queries can be done in $O((K+Q)\log(K+Q))$, and it is the heaviest part of the computation. Choosing optimal $K \sim \left(\frac{n}{\log n}\right)^{2/3}$, we obtain the complexity $O(n^{2/3}(\log n)^{1/3})$.