16. Поиск факториала по простому модулю

Часто в задачах бывает необходимо искать $n! \bmod p$. Обычно модуль — это большое число, а n не очень большое, поэтому можно просто заранее предпосчитать все факториалы. Однако иногда бывает обратная ситуация: $p < n$. С первого взгляда кажется, что это бессмысленная задача: в $n!$ в таком случае входит число p, поэтому $n! \bmod p = 0$. Но часто нас не устраивает такой ответ, потому что нам часто нужно делить факториалы друг на друга. В этом случае мы можем получить выражения типа $\frac{0}{0}$, которое на самом деле равно чему-то ненулевому. Встает задача: посчитать $n! \bmod p$ без вхождений p в $n!$ и отдельно посчитать степень вхождения p в факториал, то есть представить $n!$ в виде $a \cdot p^k$, где a не делится на p.

Пускай $\left\lfloor \frac{n}{p} \right\rfloor = k$ и $n \bmod p = b$. Тогда

$$n! = (1 \cdot 2 \cdot \ldots \cdot (p-1)) \cdot p \cdot ((p+1) \cdot (p+2) \cdot \ldots \cdot (2p-1)) \cdot (2p) \cdot \ldots \cdot (kp) \cdot ((kp+1) \cdot (kp+2) \cdot \ldots \cdot (kp+b))$$

Если брать это по модулю p, то получится

$$(p-1)! \cdot p \cdot (p-1)! \cdot (2p) \cdot \ldots \cdot (p-1)! \cdot (kp) \cdot b! = ((p-1)!)^k \cdot b! \cdot (k! \cdot p^k)$$

При этом p^k мы игнорируем, потому что это вхождения p.

По теореме Вильсона (15.1.1) $(p-1)! \bmod p = -1$. Так что нам необходимо посчитать $(-1)^k \cdot b! \cdot k!$. При этом $b < p$ (это остаток от деления на p), так что $b!$ можно посчитать за $O(p)$, после чего рекурсивно запуститься для подсчета $k!$. Каждый раз мы делим число n на p, так что будет всего $\log_p n = \frac{\log n}{\log p}$ итераций. Асимптотика — $O(p \log_p n)$. С другой стороны, все факториалы до p можно предпосчитать заранее, и тогда асимптотика будет $O(\log_p n)$ на запрос и $O(p)$ на предпосчет.

Реализация представлена ниже:

```
1 int mod_factorial(long long n, int p) {
2     int factorial = 1;
3     while (n > 0) {
```

```
4           long long k = n / p;
5           int b = n % p;
6           if (k % 2 == 1) {
7               factorial = 1LL * factorial * (p - 1) % p;
8           }
9           for (int i = 1; i <= b; i++) {
10              factorial = 1LL * factorial * i % p;
11          }
12          n = k;
13      }
14      return factorial;
15  }
```

С другой стороны, если нам надо посчитать степень вхождения p в факториал, это делается еще проще. Давайте для этого посчитаем количество чисел, которые делятся на p, на p^2, на p^3 и т.д. И тогда если число делится ровно на p^k, то мы учтем его как раз k раз. А если заметить, что количество чисел, делящихся на p^i, — это $\left\lfloor \frac{n}{p^i} \right\rfloor$, то получается такой незатейливый алгоритм:

```
1  int factorial_power(int n, int p) {
2      int power = 0;
3      while (n > 0) {
4          n /= p;
5          power += n;
6      }
7      return power;
8  }
```

Асимптотика равна $O(\log_p n)$.

17. Поиск факториала по простому модулю за $O(\sqrt{\min(p,n)}\log^2 n)$

В прошлом разделе мы научились искать $n!\mod p$ без вхождений p в факториал за $O(p\log_p n)$ или за $O(p+\log_p n)$, если заранее предпосчитать все факториалы. Однако чаще всего модуль — это число порядка 10^9, поэтому эти алгоритмы нам не подходят. Давайте научимся искать факториал быстрее.

Давайте сначала научимся искать $n!\mod p$ за $O(\sqrt{n}\log^2 n)$, если $n < p$.

Обозначим $k = \lfloor\sqrt{n}\rfloor$. Тогда мы хотим построить алгоритм за $O(k\log^2 k)$. Пусть $n = k^2 + b$, где $b \leqslant 2\cdot k$. В таком случае нам достаточно посчитать $\left(k^2\right)!$, а затем за $O(k)$ домножить его на b недостающих чисел.

Построим многочлен $P(x) := (kx)\cdot(kx-1)\cdot\ldots\cdot(kx-k+1)$. Тогда $\left(k^2\right)! = P(1)\cdot P(2)\cdot\ldots\cdot P(k)$. Введем еще один многочлен $Q(x) := P(2x)\cdot P(2x-1)$. Получается, что $\left(k^2\right)! = Q(1)\cdot Q(2)\cdot\ldots\cdot Q\left(\frac{k}{2}\right)$, если k четно, а если k нечетно, то еще нужно домножить на $P(k)$, но его мы легко можем сделать за $O(k)$.

Теперь давайте рекурсивно запустимся от многочлена Q. Глубина рекурсии будет $\log k$. Однако проблема в том, что изначально многочлен P имел степень k, однако многочлен Q имеет степень уже $2k$, и степень будет возрастать. Но если учесть, что многочлен Q нам нужно посчитать только в точках $1, 2, \ldots, \left\lfloor\frac{k}{2}\right\rfloor$, то можно взять вместо Q многочлен $Q \bmod (x-1)\cdot(x-2)\cdot\ldots\cdot(x-\left\lfloor\frac{k}{2}\right\rfloor)$. И у такого многочлена уже степень будет $< \left\lfloor\frac{k}{2}\right\rfloor$.

Взятие двух многочленов по модулю можно реализовать за $O(k\log^2 k)$ через деление. Тогда асимптотика алгоритма получается $O\left(k\log^2 k + \frac{k}{2}\log^2\left(\frac{k}{2}\right) + \frac{k}{4}\log^2\left(\frac{k}{4}\right) + \ldots\right) = O\left(\left(k+\frac{k}{2}+\frac{k}{4}+\ldots\right)\log^2 k\right) = O(k\log^2 k)$. Что и требовалось.

Чтобы получить алгоритм, который работает при $n \geqslant p$, нужно просто применить алгоритм из предыдущего раздела, однако вычислять $b!$ с помощью нового метода. $b < p$, поэтому это вычисление будет работать не дольше $\sqrt{p}\log^2 p$. Всего итераций будет $\log_p n$, так что асимптотика получается $O(\sqrt{p}\log^2 p\cdot\log_p n) = O\left(\sqrt{p}\log^2 p\frac{\log n}{\log p}\right) = O\left(\sqrt{p}\log p\log n\right)$. Если объединить случаи $p < n$ и $p \geqslant n$, получается время работы $O\left(\sqrt{\min(p,n)}\log\min(p,n)\log n\right)$.

17.1 Задачи для практики

- https://www.spoj.com/problems/FACTMODP/