Обозначения

$\sum_{i=1}^{n} a_i = a_1 + a_2 + \ldots + a_n$

$\prod_{i=1}^{n} a_i = a_1 \cdot a_2 \cdot \ldots \cdot a_n$

$f(n) = \mathcal{O}(g(n))$ — существуют такие числа a и b, что для всех n: $f(n) \le a + bg(n)$.

$f(n) = \Omega(g(n))$ — то же самое, что и $g(n) = \mathcal{O}(f(n))$.

$f(n) = \Theta(g(n))$ — одновременно $f(n) = \mathcal{O}(n)$ и $f(n) = \Omega(n)$.

$a \equiv b \pmod{m}$ — a и b имеют одинаковые остатки при делении на m.

Задачи

Задачи разбиты на блоки, которые отделены друг от друга линиями вида «— — —». На решение задач из каждого блока отводится некоторое время, после которого проводится их разбор. Рекомендуется сдавать задачи именно из текущего блока, если вы, конечно, уже не решили их все.

Задача 1. Предложите алгоритм вычисления НОД двух натуральных чисел за $\mathcal{O}(n^2)$, где n — ограничение на длину чисел в какой-то фиксированной системе счисления.

Разбор. Будем действовать следующим образом:

1. Если оба числа делятся на 2, то $\gcd(a, b) = 2 \cdot \gcd(\frac{a}{2}, \frac{b}{2})$.
2. Если ровно 1 число делится на 2, будем считать, что это a. Тогда $\gcd(a, b) = \gcd(\frac{a}{2}, b)$.
3. В противном случае считаем, что $a \ge b$, и тогда $\gcd(a, b) = \gcd(b, a - b))$.

Заметим, что после действий 1 и 2 хотя бы одно из чисел уменьшается в 2 раза. Также после действия 3 одно из чисел становится четным, следовательно, далее совершается действие 2. За каждые два последовательных действия хотя бы одно из чисел уменьшается в 2 раза, следовательно, мы совершим $\mathcal{O}(n)$ действий, каждое из которых работает за $\mathcal{O}(n)$, из чего и получается требуемое время работы $\mathcal{O}(n^2)$.

Задача 2. Даны n целых чисел, по модулю не превосходящих C. Рассмотрим следующий алгоритм вычисления их НОД.

```
answer = 0
for i = 0 ... n - 1:
    answer = gcd(answer, a[i])
```

Здесь функция `gcd` реализует алгоритм Евклида, использующий операции деления с остатком. Дайте асимптотическую оценку сложности работы данного алгоритма.

Разбор.

Лемма 1. Пусть $a \ge b$. Тогда $a\%b < \frac{a}{2}$.

Доказательство. Пусть $b > \frac{a}{2}$. Тогда очевидно, что $a\%b = a - b < \frac{a}{2}$. Иначе, если $b \le \frac{a}{2}$, то $a\%b < b \le \frac{a}{2}$.

Лемма 2. За каждые три последовательных шага алгоритма Евклида каждое из чисел в нем уменьшается хотя бы в 2 раза.

Доказательство. Случай 1. $a \ge b$. Тогда на следующем шаге будем находить $gcd(b, a\%b)$, а потом $gcd(a\%b, b\%(a\%b))$, притом очевидно, что $b \ge a\%b$. Из леммы 1 следует, что $a\%b \le \frac{a}{2}$, а также $b\%(a\%b) \le \frac{b}{2}$. То есть оба числа уменьшились хотя бы в 2 раза после двух шагов. *Случай 2.* $a < b$. Тогда после одного шага перейдём к случаю 1, и за следующие оба числа уменьшатся хотя бы в 2 раза.

Решение задачи. Рассмотрим i-й шаг цикла. Во время него алгоритм Евклида совершал какие-то действия. Мы можем разбить эти действия на последовательные тройки (действия типа 1). Также, возможно, для каждого шага останутся одно или два действия, которые не вошли в тройки. Заметим, что суммарно по всем шагам цикла последовательных троек $\mathcal{O}(\log C)$, поскольку после выполнения тройки ответ уменьшается хотя бы в 2 раза. Следовательно, суммарно будет совершено не более $\mathcal{O}(n + \log C)$ шагов алгоритма Евклида, каждый из которых работает за $\mathcal{O}(1)$.

Задача 3. Пусть a, b, c — целые числа. Рассмотрим уравнение $ax + by = c$ относительно целых x, y.

a. Покажите, что, если c не делится на $gcd(a, b)$, решений нет.

b. Покажите, что при $c = gcd(a, b)$, решение есть.

c. Покажите, что решение существует тогда и только тогда, когда c делится на $gcd(a, b)$.

d. Покажите, что, если существует хотя бы одно решение, существует бесконечно много решений. Опишите их все.

Разбор.

a. Так как $\gcd(a,b)|a$ и $\gcd(a,b)|b$, то $\gcd(a,b)|ax+by$. Значит, если решение существует, то $\gcd(a,b)|c$.

b. Решение можно явно предъявить, используя расширенный алгоритм Евклида.

c. Если c не делится на $\gcd(a,b)$ то по пункту **a** решений нет. Иначе $c = d \cdot \gcd(a,b)$. Пусть (x', y') — решение уравнения $ax+by = \gcd(a,b)$. Домножим x' и y' на d и получим решение исходного уравнения.

d. Пусть существует решение $ax_0 + by_0 = c$. Положим $x_t = x_0 + t \cdot b/\gcd(a,b)$, $y_t = y_0 - t \cdot a/\gcd(a,b)$ для некоторого целого t. Эта пара значений также является решением уравнения. Таким образом, решений бесконечно много. Осталось показать, что мы описали все решения. Заметим, что, если есть 2 решения x_1, y_1 и x_2, y_2, то $a \cdot (x_1 - x_2) + b \cdot (y_1 - y_2) = 0$ (вычли из одного равенства другое). Тогда $a \cdot (x_1 - x_2) = b \cdot (y_2 - y_1)$. Тогда существует целое число t такое, что $(b/gcd(a,b)) \cdot t = x_1 - x_2$ и $(a/gcd(a,b)) \cdot t = -(y_1 - y_2)$, следовательно, наш алгоритм, имея одно решение, опишет и другое.

Определение. Числа a и b называются взаимно обратными по модулю m, если $a \cdot b \equiv 1 \pmod m$.

Задача 4. Дано число a. Предложите алгоритм, который за $\mathcal{O}(\log m)$ вычислит обратный ему по **не обязательно простому** модулю m или определит, что такого не существует.

Разбор. Мы хотим найти такое число x, что $ax \equiv 1 \pmod m$. Иными словами, $ax + my = 1$ для некоторого целого y. Из предыдущей задачи мы знаем, что решение существует тогда и только тогда, когда $\gcd(a,m) = 1$, и в таком случае мы его можем найти.

— — —

Задача 5. Докажите, что $\frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{n} = \Theta(\log n)$.

Разбор. $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \ldots + \frac{1}{n} = \left(\frac{1}{1}\right) + \left(\frac{1}{2} + \frac{1}{3}\right) + \left(\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7}\right) + \ldots + \left(\ldots + \frac{1}{n}\right) \le$
$\le 1 \cdot 1 + 2 \cdot \frac{1}{2} + 4 \cdot \frac{1}{4} + \ldots \le \lceil \log_2(n+1) \rceil$

$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \ldots + \frac{1}{n} = \left(\frac{1}{1}\right) + \left(\frac{1}{2} + \frac{1}{3}\right) + \left(\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7}\right) + \ldots + \left(\ldots + \frac{1}{n}\right) \ge$
$\ge 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 4 \cdot \frac{1}{8} + \ldots \ge \lfloor \log_2(n+1) \rfloor / 2$

Задача 6. Пусть $\tau(n)$ — количество натуральных делителей n. Докажите, что $\sum\limits_{i=1}^{n} \tau(i) = \mathcal{O}(n \log n)$.

Разбор. Заметим, что 1 — делитель у каждого первого числа, 2 — делитель у каждого второго числа и так далее. Тогда, используя предыдущую задачу, получаем:

$$\sum_{i=1}^{n} \tau(n) = \left\lfloor \frac{n}{1} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + \ldots + \left\lfloor \frac{n}{n} \right\rfloor = \mathcal{O}(n \log n)$$

Определение. Функция Эйлера $\varphi(n)$ — количество натуральных чисел меньших n, взаимно простых с n.

Задача 7. **a.** Выведите формулу для $\varphi(p^k)$ (p — простое число), которая будет зависеть от p и k.

b. Предложите алгоритм вычисления $\varphi(n)$ для произвольного n за $\mathcal{O}(\sqrt{n})$.

Разбор.

a. Все числа, меньше или равные p^k, взаимно просты с p^k, кроме тех, что делятся на p.
Поэтому $\varphi(p^k) = p^k - \frac{p^k}{p} = p^k - p^{k-1}$.

b. Функция Эйлера мультипликативная, то есть для любых взаимно простых a, b: $\varphi(ab) = \varphi(a)\varphi(b)$. Следовательно, достаточно факторизовать n, после чего задача сводится к пункту **a**.

Задача 8. (*Китайская теорема об остатках*) Дано n взаимно простых чисел $a_1, a_2, \ldots, a_n$ ($a_i \le C$). Дано n неотрицательных чисел $r_1, r_2, \ldots, r_n$. Докажите, что существует единственное число x ($0 \le x < \prod\limits_{i=1}^{n} a_i$) такое, что $x \equiv r_i \pmod{a_i}$ для всех $i = 1 \ldots n$.

Разбор. Смотрите на wiki.algocode.ru.

Задача 9. В предположении, что все арифметические операции с числами любой длины выполняются за $\mathcal{O}(1)$, научитесь находить число x из предыдущей задачи за $\mathcal{O}(n \log C)$.

Разбор. Смотрите на wiki.algocode.ru.

Задача 10. (*Теорема Эйлера*) Докажите, что если k взаимно просто с n, то $k^{\varphi(n)} \equiv 1 \pmod n$.

Примечание. Частный случай этой теоремы при простом n называется *малой теоремой Ферма*. Этот случай примечателен тем, что $\varphi(p) = p - 1$ для простого p, отчего теорема принимает совсем простой вид.

Разбор. Смотрите на wiki.algocode.ru.

— — —

Задача 11. Предложите способ ускорить решето Эратосфена, чтобы оно работало за линейное время.

Подсказка. Для каждого числа попробуйте найти минимальный простой делитель.

Разбор. Смотрите на wiki.algocode.ru.

Задача 12. За $\mathcal{O}(n)$ для всех чисел от 1 до n найдите:

a. В какой степени минимальный простой делитель входит в его разложение.

b. Количество его простых делителей.

c. Количество его делителей.

d. Сумму его делителей.

e. Функцию Эйлера от него.

Разбор.

a. При подсчёте решета Эратосфена добавляем вторую динамику — степень минимального простого делителя.

b. При подсчёте решета Эратосфена добавляем вторую динамику — кол-во простых делителей.

c. Будем использовать пункт **a**. Пусть у нас число x, мин. простой делитель — p и он входит в степени q. Тогда количество делителей x равно произведению количества делителей p^q и количества делителей $x/(p^q)$ (функция количества делителей, как и функция Эйлера, мультипликативна). Таким образом, мы можем насчитывать нужную динамику.

d. Аналогично пункту **c**.

e. Аналогично пункту **c**.

Задача 13. Научитесь вычислять $a \cdot b$ для натуральных a и b, используя только сложение, деление на 2 (в том числе с остатком), а также проверку на равенство единице за $\mathcal{O}(\log a)$ операций сложения.

Разбор. Пусть a и b — перемножаемые числа.

1. Если $b = 1$, то $a \cdot b = a$.
2. Если b четно, то $a \cdot b = (a + a) \cdot (b/2)$
3. Если b нечетно, то $a \cdot b = a + (a + a) \cdot (\lfloor b/2 \rfloor)$

Запустим рекурсивный алгоритм, использующий вышеописанные наблюдения. За каждый шаг b уменьшается в 2 раза. То есть мы получим ответ за $O\big(\log b\big)$ действий.

Задача 14. Для всех $k \le n \le X$ научитесь находить C_n^k по простому модулю $p > X$ за $\mathcal{O}(1)$, с предподсчётом за $\mathcal{O}(n)$.

Разбор. Смотрите на wiki.algocode.ru.

Задача 15. (*Дискретное логарифмирование*) $a^x \equiv b \pmod m$, a и m взаимнопросты. Найти решение или определить, что его не существует, за время $\mathcal{O}(\sqrt{m} \log m)$.

Подсказка. Представьте x в виде $ky - r$ для $k = \lfloor \sqrt{m} \rfloor$.

Разбор. Смотрите на wiki.algocode.ru.

— — —

Задача 16. Найти количество натуральных чисел, меньших n, взаимно простых с числом m, за $\mathcal{O}(\sqrt{m})$

Разбор. Заметим, что от числа m нам интересны только его простые делители $p_1, p_2, \ldots, p_k$. Будем считать x — количество натуральных чисел, меньше или равных n, которые **не** взаимно просты с m, то есть делятся на какое-то из p_i. Тогда ответ на задачу равен $n - x$.

Пусть A_i — множество натуральных чисел, меньше или равных n и делящихся на p_i. Тогда $x = A_1 \cup A_2 \cup \ldots \cup A_k$. Для подсчета мощности этого объединения множеств, мы можем воспользоваться методом включений-исключений: просто переберем все подмножества этой системы множеств, и включим в ответ мощности пересечения нечетного количества множеств со знаком «+», а нечетного — со знаком «−». Осталось научиться эффективно вычислять мощность таких пересечений. Понятно, что натуральных чисел, меньше или равных n, и делящихся одновременно на $p_{i_1}, p_{i_2}, \ldots, p_{i_t}$ ровно $\left\lfloor \frac{n}{p_{i_1} \cdot p_{i_2} \cdot \ldots \cdot p_{i_t}} \right\rfloor$.

Время работы данного алгоритма складывается из времени на факторизацию числа m и перебора всех подмножеств простых делителей m, но подмножеств простых делителей не более, чем делителей числа.

Задача 17. Найти сумму НОД по всем подотрезкам массива натуральных чисел, не больших C, за $\mathcal{O}(n \log C)$.

Разбор. *Лемма.* Если при добавлении числа в множество НОД всех чисел в нём изменяется, то НОД уменьшается хотя бы в 2 раза. *Доказательство.* Новый НОД — делитель старого.

Решение задачи. Будем идти слева-направо, фиксируя правую границу отрезка r. Тогда для заданного r, если будем двигать $l = r \dots 1$, получим $\mathcal{O}(\log C)$ различных значений НОД. Будем поддерживать список отрезков левых границ с равным НОД — элементов в нём будет $\mathcal{O}(\log C)$ (это следует из леммы). Имея этот список, легко посчитать сумму НОД для отрезков с таким правым концом.

Осталось научиться переходить от правой границы r к $r+1$. Пусть $(r+1)$-е число равно x. Тогда это делается так:

1. Для самого правого отрезка левых границ с НОД, равным g делаем: $g = \gcd(g, x)$. Перебираем справа-налево остальные отрезки левых границ. Пусть в текущем рассматриваемом отрезке НОД равен g, а в том, что справа от него, g'. Тогда делаем: $g = \gcd(g, g')$. Из задачи 2 следует, что все обновления НОД выполнятся суммарно за $\mathcal{O}(\log C)$.

2. В список добавляем отрезок левых границ $[r+1; r+1]$ со значением НОД x.

3. Если какие-то соседние отрезки левых границ в списке стали иметь равный НОД, нужно их объединить в один. Это обеспечит нам то, что в любой момент в списке $\mathcal{O}(\log C)$ элементов. Удобнее всего объединять отрезки, создав новый список и, проходя по старому, добавлять в новый список уже сжатые отрезки.

Задача 18. Найти сумму НОД по всем непустым подмножествам массива из n натуральных чисел, не больших C, за $\mathcal{O}(n + C \log C)$. Ответ найдите по модулю $10^9 + 7$.

Разбор. Найти сумму *gcd* по всем непустым подмножествам массива из n натуральных чисел, не больших C, за $O(n + C \log C)$. Ответ найдите по модулю $10^9 + 7$.

Решение. Заведём $cnt[x]$ — количество чисел, равных x. Посчитаем $d[x]$ — количество чисел, делящихся на x. Он вычисляется за $O(C \log C)$ так:

$$d[x] = \sum_{i=1}^{\lfloor C/x \rfloor} cnt[i \cdot x].$$

Далее, посчитаем $sets[x]$ — количество множеств с НОД, равным x. Он вычисляется в обратном порядке так:

$$sets[x] = 2^{d[x]} - 1 - \sum_{i=2}^{\lfloor C/x \rfloor} sets[i \cdot x].$$

Ответ — это просто $\sum\limits_{i=1}^{C} i \cdot sets[i]$.

Задача 19. Дан массив из n натуральных чисел, не больших C. Выпишем *gcd* по всем непустым подмножествам этого массива. Найти медиану выписанных чисел за $\mathcal{O}(n \cdot C \log C)$.

Разбор. Аналогично предыдущей задаче, но нужно воспользоваться длинной арифметикой: количества подмножеств содержат $O(n)$ цифр.

Задача 20. Назовём натуральное число кубастым, если его можно представить в виде $a^3 \cdot b$ для каких-то натуральных $a > 1, b \geq 1$. Найти количество кубастых чисел, не больших n. (n до 10^{18}, стандартный компьютер со стандартными ограничениями)

Разбор. Будем перебирать a ($a \leq n^{\frac{1}{3}}$), свободные от квадратов (то есть такие, которые не делятся ни на какой квадрат натурального числа, большего единицы). Далее достаточно применить метод включений-исключений, аналогично задаче 18.