

Задача А. Интересные числа

Подзадача 1. Для всех значений $r \leq 9$ требуемое количество чисел совпадает с r , поскольку каждое число первого десятка, очевидно, делится на единственную цифру в своей записи. Поэтому числа из оставшегося промежутка $[10; 20]$ легко найти непосредственной проверкой.

Подзадача 2. Будем перебирать все целые положительные числа n от 1 до r , каждый раз проверяя, делится ли n на свою последнюю цифру. Заметим, что последняя (справа) цифра в десятичной записи заданного числа n совпадает с остатком при делении n на 10, то есть равна $d = n \% 10$. Поскольку в этой подзадаче $r \leq 10^6$, непосредственная проверка делимости на d всех чисел из промежутка $[1; r]$ проходит по времени тесты второй группы.

Подзадача 3. Последняя цифра, на которую делятся требуемые числа, отлична от нуля, и значит, равна некоторой цифре от 1 до 9. Рассмотрим каждую цифру от 1 до 9 отдельно.

Заметим, что все числа, оканчивающиеся на 1 или на 2, делятся на 1 и на 2, то есть удовлетворяют условию задачи. Другими словами, интересными будут все числа вида $1, 11, 21, \dots$, а также числа вида $2, 12, 22, \dots$. Сколько таких чисел попадает в промежуток $[1; r]$? Их количество равно $(r-1)/10 + 1$ — для чисел первого вида, и $(r-2)/10 + 1$ — для чисел второго вида.

Теперь рассмотрим числа, оканчивающиеся на 3 (или на 6), кратные 3 (или 6) — они образуют арифметическую прогрессию с шагом 30: $3, 33, 63, \dots$ (или $6, 36, 66, \dots$). Поэтому их количество равно $(r-3)/30 + 1$ (или $(r-6)/30 + 1$). Аналогично рассматриваются остальные случаи возможных значений для последних цифр. В каждом из них определяем длину арифметической прогрессии, соответствующей значению последней цифры.

Приведённое решение не требует проверки всех чисел от 1 до r и имеет сложность $O(1)$.

Задача В. Кофейные автоматы

Подзадача 1. Составим для одной кофе-машины математическую модель. В данном случае, это уравнения: $a \cdot x - b \cdot y = 1$ или $b \cdot y - a \cdot x = 1$. Если одно из уравнений имеет решение, то Ильнур может расплатиться. Будем перебирать все неотрицательные целые коэффициенты x и y .

Подзадача 2. Будем перебирать все неотрицательные целые коэффициенты x и y , проверяя, что выполняется одно из равенств: $a \cdot x - b \cdot y = c$ или $b \cdot y - a \cdot x = c$.

Подзадача 3. Пусть x и y теперь будут целыми числами (допускаем, что они могут быть отрицательными). Тогда следует проверить равенство $a \cdot x + b \cdot y = c$. Выразим $x = \frac{1}{a}(c - b \cdot y)$. Значит, $c - b \cdot y$ должно делиться на a , то есть $(c - b \cdot y) \equiv 0 \pmod{a} \implies b \cdot y \equiv c \pmod{a}$. Поиск решения в равенстве по модулю a означает перебор y от 0 до $a-1$; если нет решения, то Ильнур не может купить кофе.

Подзадача 4. Если $a = b$, то уравнение $a \cdot x + b \cdot y = c$ будет иметь решение тогда и только тогда, когда $c \pmod{a} \equiv 0$. Так как a и b взаимно простые, то $b \cdot y \pmod{a}$ пробегает все классы вычетов по \pmod{a} , и значит, в этом случае найдётся решение. Другими словами, в тестах этой подзадачи Ильнур всегда сможет купить себе кофе.

Подзадача 5. Уравнение $a \cdot x + b \cdot y = c$ называется диофантовым, и оно имеет решение, если НОД(a, b) является делителем c . Для решения данной подзадачи найдем НОД(a, b) перебором делителей a за $O(\sqrt{a})$.

Подзадача 6. Для полного решения задачи нужно искать НОД с помощью алгоритма Евклида (<https://brestprog.by/topics/gcd/>).

Задача С. Приборная панель

Подзадачи 1-2. С помощью перебора всех возможных состояний кнопок найдём нужную последовательность нажатий, которая соответствует требуемому состоянию освещённости здания. Всего

таких состояний 2^{10} , поэтому такое решение проходит по времени все тесты подзадачи 1. Для решения подзадачи 2 можно отсортировать по возрастанию массив номеров освещаемых комнат и провести некоторые оптимизации предыдущего решения.

Подзадача 3. Прежде всего, отметим, что результат нажатия нескольких кнопок не зависит от порядка их нажатия. Далее, *повторное* нажатие одной и той же кнопки приборной панели не меняет освещённость здания, поэтому требуемая минимальная последовательность не должна содержать совпадающих номеров нажатых кнопок.

Определим битовый массив b из n чисел 0 и 1, соответствующий текущему состоянию освещённости всех n комнат здания; i -й элемент этого массива равен 1, если в комнате i лампочка включена, и равен 0, – если выключена.

Рассмотрим отсортированный по возрастанию массив номеров комнат, в которых необходимо включить лампочки. Пусть j – номер *первой* такой комнаты (в примере это комната $j = 3$). Для включения лампочки в этой комнате используем кнопку 3, при этом все комнаты с номерами, кратными 3, также окажутся освещёнными – это комнаты 3, 6, 9 и так далее. С помощью команды $b_j \rightarrow 1 - b_j$ заменим в массиве b значения соответствующих битов $b[j], b[2j], \dots$ на противоположные. Вновь определим номер первой после j не освещённой комнаты 6, повторно нажмём на кнопку 6, и так далее, до тех пор, пока все комнаты не станут освещёнными.

Задача D. Комбинаторная система

Подзадачи 1–2. Будем искать комбинаторную запись числа $n = C_{x_3}^3 + C_{x_2}^2 + C_{x_1}^1$, перебирая значения x_3, x_2, x_1 с ограничением $x_3 \geq x_2 \geq x_1 \geq 0$. Так как $n \leq 10^5$, то значения величин x_i , очевидно, не больше 1000. Таким образом, подзадачи 1 и 2 легко решаются несложным перебором с помощью не более двух циклов.

Подзадачи 3–4. Будем строить комбинаторную запись числа n , начиная с поиска самой первой (слева) цифры x_m . Для этого воспользуемся «жадным» правилом – подберём *наибольшее* значение x_m , для которого выполняется неравенство $C_{x_m}^m \leq n$. Теперь нужно найти комбинаторную запись числа $n_1 = n - C_{x_m}^m$, и следующую цифру x_{m-1} выбираем так, чтобы $C_{x_{m-1}}^{m-1} \leq n_1$. Затем снова вычисляем разность $n_2 = n_1 - C_{x_{m-1}}^{m-1}$, и так далее, пока не придем к значению разности, равной нулю.

Почему жадный алгоритм работает правильно и почему нужно выбирать x_m как можно бóльшим? Действительно, определим x_m и x_{m-1} «жадным» способом и предположим, что оказалось $x_m \leq x_{m-1}$. Тогда

$$C_{x_{m+1}}^m = C_{x_m}^m + C_{x_{m-1}}^{m-1} \leq C_{x_m}^m + C_{x_{m-1}}^{m-1} \leq n \iff C_{x_{m+1}}^m \leq n,$$

то есть x_m можно было взять больше, противоречие. После нескольких шагов мы приходим к случаю $m = 1$, который разбирается очевидным образом, поскольку $C_{x_1}^1 = x_1$. Из этого рассуждения следует, что жадный алгоритм обосновано приводит к записи числа n в комбинаторной системе счисления.

В зависимости от технической реализации процедуры подсчёта комбинаторных коэффициентов $C_{x_k}^k$, удаётся построить решение соответствующих подзадач 3–4.

Подзадача 3. Будем вычислять значения C_s^k , используя формулу

$$C_s^k = \frac{s(s-1) \cdot \dots \cdot (s-k+1)}{1 \cdot 2 \cdot \dots \cdot k}.$$

Поскольку значение x_m удовлетворяет неравенству $C_{x_m}^m \leq n$, будем подбирать значение x_m , начиная с 0, с шагом 1, пока не нарушится условие $C_{x_m}^m \leq n$. Далее, вычисляем разность $n_1 = n - C_{x_m}^m$, и сводим задачу к меньшему значению $m - 1$, и так далее.

Такая реализация жадного алгоритма проходит все тесты подзадачи 3.

Подзадача 4. Для вычисления значений комбинаторных коэффициентов C_s^k можно использовать рекуррентную формулу

$$C_s^k = C_{s-1}^k + C_{s-1}^{k-1},$$

сохраняя на каждом шаге значения C_s^k при всех k , $0 \leq k \leq s$. При вычислении коэффициентов C_s^k при фиксированном значении s можно также воспользоваться формулой

$$C_s^k = \frac{s - k + 1}{k} \cdot C_s^{k-1}$$

при начальном условии $C_s^0 = 1$. При каждом шаге итерации числитель уменьшается на 1 (начальное значение равно n), а знаменатель соответственно увеличивается на 1 (начальное значение -1). Для вычисления значения C_s^k этот метод требует $O(1)$ памяти и $O(k)$ времени.

Каждая такая реализация проходит все тесты подзадачи 4.

Задача Е. Вызов к доске

Подзадача 1. При указанных в подзадаче ограничениях решение можно получить моделированием процесса.

Подзадача 2. Пусть ученики пронумерованы целыми числами от 0 до $n-1$. В этом случае можно считать, что номер s принимает значения от 0 до $n-1$. Если Ильнур начинает отсчёт от ученика с номером s , то счёт остановится на $(s + a_s) \bmod n$. Таким образом, мы ускоряем моделирование.

Подзадача 3. Переформулируем задачу в терминах графов: ученики — вершины, а переход от ученика s к $(s + a_s) \bmod n$ будет ребром из s в $(s + a_s) \bmod n$. Такой граф является *функциональным*.

Известно, что каждая компонента связности функционального графа состоит из единственного цикла (возможно, петли) и присоединённых к нему деревьев, все дуги которых ориентированы к циклическим вершинам. (Вершина считается циклической, если из нее существует путь в саму себя.)

Пусть мы начинаем из вершины s и сделаем t шагов до циклической вершины y , длина цикла этой вершины равна r . Тогда ответом для вершины s будет $(k - t) \bmod r$ -я вершина цикла от вершины y .

Подзадача 4. Поскольку запросов много, разложим их по вершинам и будем их обрабатывать совместно. Затем для каждой вершины запустим поиск цикла, при этом пройдем не более n вершин для каждой из них. Асимптотика такого решения — $O(q + n^2)$.

Подзадача 5. Разложим запросы по вершинам. Для вершин, лежащих в цикле, ответ очевиден: зафиксируем порядок вершин в цикле $(v_0, v_1, \dots, v_{len_{cycle}-1})$, и тогда ответ для запроса в вершине v_z равен $v_{(z+k) \bmod len_{cycle}}$.

Теперь решим задачу для запросов, не стартующих в циклических вершинах. Удалим рёбра, лежащие в цикле, и поменяем направление рёбер, не входящих в цикл. Запустим обход dfs от циклической вершины v_z .

Пусть есть ребро из v_z в v_t (v_t — не циклическая вершина). Тогда ответом для всех запросов в вершине v_t будет $v_{(z+k-1) \bmod len_{cycle}}$.

Если k меньше расстояния от вершины s до циклической вершины v_z , то будем выписывать последовательность вершин от v_z и находить нужную вершину в этой последовательности.