

# Бесконечная зима в Карелии

## Подзадачи 1–2.

В этих подзадачах действовало ограничение  $k = 2$ . Попробуем для каждого запроса  $x$  вычислить наименьшее количество монет  $a_2$ , которые можно использовать, чтобы набрать данную сумму.

Заметим, что это количество зависит только от  $x \bmod a_1$ . А именно мы знаем, что если  $x \equiv k \cdot a_2 \pmod{a_1}$ , то можно использовать в сумме  $k$  монет номиналом  $a_2$ , а всю оставшуюся сумму добить монетами номиналом  $a_1$ . Рассматривать значения  $k \geq a_1$  бессмысленно, поэтому мы можем просто перебрать  $k$  от 0 до  $a_1 - 1$  и для каждого остатка при делении на  $a_1$  вычислить минимальное количество монет номиналом  $a_2$ , которое надо набрать в сумму, пусть это количество равно  $\min_2[r]$  для  $x \equiv r \pmod{a_1}$ . Эта часть решения работает за время  $O(d)$ .

С помощью этого массива легко можно проверить, возможно ли решение в принципе, а также найти решение, которые использует минимальное количество монет номиналом  $a_2$ . Разберемся с тем, как выглядят другие решения. Очевидно, что для получения других решений нам понадобится уменьшать количество монет номиналом  $a_1$  и увеличивать количество монет номиналом  $a_2$ . Пусть  $g$  — наибольший общий делитель  $a_1$  и  $a_2$ , тогда мы можем уменьшать количество монет первого номинала на  $\frac{a_1}{g}$ , а количество монет второго номинала увеличивать на  $\frac{a_2}{g}$ . Таким образом для подсчета количества решений достаточно вычислить, сколько раз мы можем уменьшать количество монет номинала  $a_1$ .

## Подзадачи 3–4.

Воспользуемся методом динамического программирования. Пусть  $\text{dp}[l][x]$  — количество способов набрать  $x$  рублей, используя только монеты номиналом  $a_1, \dots, a_l$ . Очевидно, что для  $l = 0$  есть только одно ненулевое состояние динамики —  $\text{dp}[0][0] = 1$ . Разберем как выглядит переход, если мы хотим вычислить  $l$ -й слой динамики:

- Количество способов набрать сумму  $x$ , которые не используют монету  $a_l$  равно  $\text{dp}[l-1][x]$ ;
- Количество способов набрать сумму  $x$ , которые используют монету  $a_l$  равно  $\text{dp}[l][x - a_l]$ , так как в любом из этих способов можно вычесть одну монету номиналом  $a_l$ .

Таким образом  $\text{dp}[l][x] = \text{dp}[l-1][x] + \text{dp}[l][x - a_l]$  (если  $x < a_l$ , то второе слагаемое отсутствует). Эту динамику легко вычислить за время  $O(k \cdot \max\{x_i\})$ , а затем за  $O(1)$  ответить на каждый из запросов.

## Подзадача 5.

Решение этой подзадачи не ведет к полному решению, но использует популярный в других задачах подход. Дело в том, что динамику можно вычислять с помощью возведения матрицы в степень. Способов это сделать много, но мы приведем самый простой, если вы не знакомы с этой техникой.

$$\text{dp}[x] = \sum_{\emptyset \neq B \subseteq \{a_1, \dots, a_k\}} (-1)^{|B|-1} \text{dp} \left[ x - \sum_{y \in B} y \right]$$

Тут  $\text{dp}[x]$  — непосредственно ответ на задачу, количество способов набрать сумму  $x$  с помощью монет номиналом  $a_1, \dots, a_k$  ( $\text{dp}[x] = 0$  для  $x < 0$ ). Объясним почему эта формула верна:

- Каждый набор учтен не менее одного раза в сумме  $\text{dp}[x - a_1] + \dots + \text{dp}[x - a_k]$ , мы просто перебираем монету, которая есть в наборе и вычитаем ее.
- Но если набор содержит сразу две различные монеты, то он учтен в этой сумме дважды, поэтому все такие наборы мы опять таки вычитаем —  $\sum_{i < j} \text{dp}[x - a_i - a_j]$ .
- Но теперь наборы, которые содержат три различные монеты, вообще не учтены в сумме, поэтому их надо прибавить, и так далее.

То, что произошло выше, является классическим объяснением формулы включений-исключений. Теперь вернемся к рекуррентному выражению. Если сгруппировать одинаковые слагаемые, то мы заметим, что формула может иметь вид:

$$\text{dp} = \sum_{j=1}^{dk} \lambda_j \cdot \text{dp}[x - j]$$

Где  $\lambda_1, \dots, \lambda_{dk}$  — какие-то коэффициенты. В этой подзадаче можно было наивно перебрать все подмножества делителей и наивно вычислить эти коэффициенты за время  $O(2^k)$ . Но теперь после всех этих действий мы, наконец, можем определить переход динамики с помощью матриц:

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ \lambda_{dk} & \lambda_{dk-1} & \lambda_{dk-2} & \dots & \lambda_1 \end{pmatrix} \cdot \begin{pmatrix} \text{dp}[n - dk + 1] \\ \text{dp}[n - dk + 2] \\ \vdots \\ \text{dp}[n - 1] \\ \text{dp}[n] \end{pmatrix} = \begin{pmatrix} \text{dp}[n - dk + 2] \\ \text{dp}[n - dk + 3] \\ \vdots \\ \text{dp}[n] \\ \text{dp}[n + 1] \end{pmatrix}$$

Так как произведение матриц ассоциативно, то можно и выразить нужное состояние динамики с помощью операции возведения матрицы в степень (символом  $*$  обозначены числа, которые нас не интересуют):

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ \lambda_{dk} & \lambda_{dk-1} & \lambda_{dk-2} & \dots & \lambda_1 \end{pmatrix}^N \cdot \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} * \\ * \\ \vdots \\ * \\ \text{dp}[N] \end{pmatrix}$$

В итоге получили решение с асимптотикой  $O(q \cdot (dk)^3 \cdot \log x)$ , это же решение можно оптимизировать до времени  $O((dk)^3 \log C + q \cdot (dk)^2 \cdot \log x)$ .

#### Подзадачи 6–9

Подзадачи 6–7 были нужны, чтобы стимулировать участников на написание полного решения. В этих задачах ясно, что  $k \leq 10$  и есть мотивация разбивать монетки на группы. При ограничениях этой задачи ясно, что  $k \leq 72$  (в этом можно убедиться, если непосредственно вычислить количество делителей у всех чисел от 1 до 15 000).

Рассмотрим произвольный набор монет с суммой  $x$ . Некоторые монеты одинакового номинала можно сложить в «пачки» стоимостью  $d$  (если рассматривается  $i$ -й номинал, то в пачке будет содержаться  $\frac{d}{a_i}$  монет). Таким образом, каждый набор монет однозначно разбивается на две части:

- Монеты, которые образуют пачки. Суммарная стоимость таких монет обязательно делится на  $d$ ;
- Монеты, которые не попали в пачки. То есть монета  $a_i$  взята не более чем  $\frac{d}{a_i} - 1$  раз.

Рассмотрим подробнее монеты, которые не попали в пачки. Пусть  $\text{dp}[y]$  — количество способов сумму  $y$  так, чтобы никакие использованные монеты не могли образовать пачку. Это количество легко вычислить с помощью динамического программирования. Пусть  $\text{dp}[l][y]$  — количество способов набрать сумму  $y$  используя монеты  $a_1, \dots, a_l$  так, чтобы никакой набор монет одинакового номинала не образовывал пачку.

Переход имеет следующий вид:

$$\text{dp}[l][y] = \text{dp}[l - 1][y] + \text{dp}[l - 1][y - a_l] + \dots + \text{dp}[l - 1][y - d + a_l]$$

Но вычислять динамику именно по такой формуле долго, поэтому можно сократить вычисления используя следующий трюк:

Понятно, что такую динамику имеет смысл считать только для  $0 \leq y < k \cdot d$ . Таким образом требуемое значение можно вычислить за время  $O(k^2 \cdot d)$ .

Теперь для ответа на запрос  $x$  мы можем перебрать суммарную стоимость монет, которые не образуют пачку  $x_0$  ( $x - x_0$  делится на  $d$  и  $x_0 < k \cdot d$ , т.е. есть  $\leq k$  вариантов значения  $x_0$ ). Положим  $n = \frac{x - x_0}{d}$  — это количество пачек, которые надо набрать, чтобы получить сумму  $x$ .

По сути надо вычислить количество массивов  $\{y_1 + \dots + y_k = n\}$ , состоящих из целых неотрицательных чисел. Это так, потому что можно считать, что  $y_i$  — это количество пачек из монет номиналом  $a_i$ . Найти это количество легко с помощью метода шаров и перегородок, оно равно:

$$\frac{(n+1) \cdot (n+2) \cdot \dots \cdot (n+k-1)}{(k-1)!}$$

Числитель можно вычислить наивно за время  $O(k)$ , проблемой остается только то, как поделить на  $(k-1)!$  по модулю  $10^9 + 7$ . Так как  $10^9 + 7$  — простое число, то обязательно найдется такое число  $f$ , что  $f \cdot 72! \equiv 1 \pmod{10^9 + 7}$ , такое число  $f$  можно найти перебором и вписать это значение как предпосчитанное в код программы.

Теперь просто заметим, что:

$$(k-1)! \cdot (f \cdot k \cdot (k+1) \cdot \dots \cdot 72) \equiv 1 \pmod{10^9 + 7}$$

Поэтому просто вычислим  $g = f \cdot k \cdot (k+1) \cdot (k+2) \cdot \dots \cdot 72$  и вместо деления на  $(k-1)!$  будем домножать выражение на  $g$ . Таким образом ответ на задачу равен:

$$\sum_{x_0} \text{dp}[k][x_0] \cdot g \cdot (n+1)(n+2) \dots (n+k-1), \quad n = \frac{x - x_0}{d}$$

Полученное решение имеет время работы  $O((d+q) \cdot k^2)$  и укладывается в ограничения задачи.