Степенные числа

Автор задачи и разработчик: Иван Пальченков

Идея этой задачи заключалась в том, чтобы заметить, что k-степенное число — это число, в записи которого в системе счисления с основанием k не содержится никаких цифр, кроме 0 и 1.

При $k_i = 2$ любое число при переводе состоит только из цифр 0 и 1, т.е. для решения **первой подзадачи** достаточно было вывести само число n. Что интересно, некоторые полные решения, реализованные недостаточно эффективно (с большой константой над асимптотикой времени работы), не проходили эту подзадачу по времени, поэтому, если вы сталкивались с такой проблемой, можно было отдельно поставить `if` на запросы с $k_i = 2$.

Для решения **второй подзадачи** можно заметить, что при $n \leqslant k$ ответом является n при $n = 1$, или k в противном случае. Действительно, число k всегда является k-степенным, потому что записывается как 10_k, и в принципе просто равно k^1, что подходит под определение.

В **третьей подзадаче** почти достаточно было написать последовательный перевод всех чисел больше либо равных n в систему счисления с основанием k, после чего для каждого из них проверить, является ли оно k-степенным. Для перевода числа в другую систему счисления можно воспользоваться стандартным алгоритмом:

1. вычислим последнюю цифру числа x как $x \bmod k$;
2. оставшиеся цифры образуют число $\left\lfloor \frac{x}{k} \right\rfloor$ — повторим шаг 1 для него, если оно больше 0;
3. в конце остается развернуть массив выписанных цифр.

Если заметить, что ответ всегда не превышает $n \cdot k$, потому что между n и $n \cdot k$ всегда есть число, равное k^a для некоторого целого a, то будет видно, что при $k \leqslant 50$ такое решение укладывается в ограничения по времени, а при $k > 50$ ответ будет состоять не более чем из четырех цифр в k-ичной системе счисления, и поэтому не более шестнадцати вариантов ответа можно просто перебрать отдельно.

Чтобы решить **четвертую и пятую подзадачи**, можно было предпосчитать все хорошие числа в интервале от 1 до 10^6 за $\mathcal{O}(10^6 \cdot \log_k n)$ описанным выше образом. Для нахождения первого большего либо равного хорошего числа для каждого запроса можно воспользоваться бинарным поиском по полученному массиву — в C++ это можно сделать, например, встроенной функцией `lower_bound`. Четвертая подзадача также не требовала перевода числа между системами счисления: достаточно было считать число как строку.

Шестая подзадача решалась перебором всех битовых масок длины $\log_k n$, то есть всех подмножеств тех степеней k, которые войдут в число как слагаемые. Действительно, если каждая цифра числа в k-ичной системе счисления равна 0 или 1, то у нас есть выбор для каждой степени k от 0 до $\log_k n$: взять ее в число или не взять.

Поскольку в этой подзадаче $k_i \geqslant 20$, то $\log_k n \leqslant 13$, то есть различных масок не более 2^{13}, что проходит в ограничения по времени с $q \leqslant 500$. Стоило обратить внимание на то, как из битовой маски получить десятичное число: проще всего предподсчитать степени k и просто сложить те, которые вошли в итоговый ответ.

Рекурсивный перебор в этой подзадаче работает эффективнее перебора двоичных масок и перевода в десятичную систему счисления, потому что можно считать получаемое в итоге перебора число по ходу перебора, не тратя на это лишние $\log_k n$ времени.

Наконец, **полное решение** — это стандартный алгоритм генерации следующего лексикографически комбинаторного объекта. Переведем число n в систему счисления с основанием k. Если в записи не встретилось цифр больше 1, само число n будет ответом. Иначе,

- найдем самый старший разряд, в котором цифра больше 1 — пусть это будет `pos`;
- чтобы итоговое число стало не меньше, какой-то из более старших разрядов надо увеличить, то есть сделать равным 1;

- а тогда во все разряды младше увеличенного можно поставить цифру 0, чтобы оно было минимальным из больших n;
- чтобы выбрать, какой разряд увеличивать до единицы, найдем первый нулевой разряд старше `pos` и поставим в него 1 (если такого разряда не нашлось, добавим новый).

Асимптотика времени работы ответа на запрос — $\mathcal{O}(\log_k n)$.