Разбор задачи «Автобусы»

Обозначим за $lcm(a, b, c)$ наименьшее общее кратное чисел a, b, c.

Легко видеть, что все три автобуса одновременно отправляются от остановки в моменты времени 0, $lcm(a, b, c)$, $2 \cdot lcm(a, b, c)$, $3 \cdot lcm(a, b, c)$, ...

Тогда задача сводится к тому, чтобы посчитать сколько чисел на отрезке $[(d-1) \cdot t; d \cdot t - 1]$ кратны $lcm(a, b, c)$.

Обозначим за $pr(x)$ количество неотрицательных целых чисел, не превосходящих x, которые кратны $lcm(a, b, c)$, тогда $pr(x) = \lfloor \frac{x}{lcm(a,b,c)} \rfloor + 1$ при $x \geqslant 0$ и $pr(x) = 0$ при $x \leqslant -1$.

Количество интересующих нас чисел на произвольном отрезке $[l; r]$ можно выразить следующим образом: $pr(r) - pr(l-1)$, а, следовательно, ответ на задачу равен $pr(d \cdot t - 1) - pr((d-1) \cdot t - 1)$.

Асимптотика: $O(\log a + \log b + \log c)$ или $O(a + b + c)$ в зависимости от алгоритма для вычисления $lcm(a, b, c)$.

Разбор задачи «День рождения»

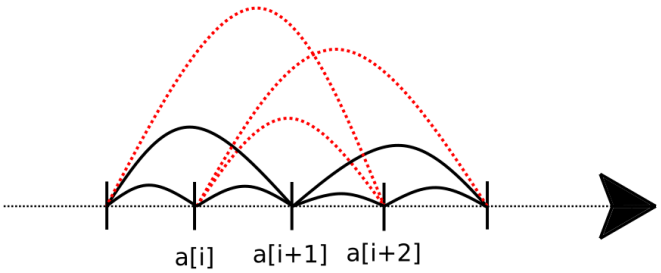
Упорядочим a_1, a_2, a_3, $\cdots$, a_n так, что $a_i \leqslant a_{i+1}$.

После этого выпишем в ответ сначала все элементы с нечётными индексами в порядке возрастания, а потом все элементы с чётными индексами в порядке убывания.

Например, если $n = 5$, то выпишем $a_1, a_3, a_5 \mid a_4, a_2$, а если $n = 6$, то $a_1, a_3, a_5 \mid a_6, a_4, a_2$.

Заметим, что данная конструкция приводит к ответу, не превосходящему $\max\limits_{1 \leqslant i \leqslant n-2} (a_{i+2} - a_i)$.

Покажем, что для любой разницы $a_{i+2} - a_i$ ответ не может быть меньше. Для этого изобразим возможные переходы как граф. Вершинами будут числа a_i, а решению задачи будет соответствовать гамильтонов (то есть проходящий по всем вершинам) цикл.



Попробуем получить ответ меньший $a_{i+2} - a_i$. Красным отмечены запрещённые переходы, черным разрешенные. Несложно видеть, что $a_i + 1$ является точкой сочленения графа и гамильтонова цикла не существует.

Асимптотика: $O(n \log n)$ для сортировки массива и $O(n)$ для последующего построения ответа.

Разбор задачи «Произведение строк»

Заметим, что умножение строк ассоциативно, то есть $(a \cdot b) \cdot c = a \cdot (b \cdot c)$. Таким образом, вместо порядка умножения "$(a \cdot b) \cdot c$", заданного в условии, будем использовать "$a \cdot (b \cdot c)$"

То есть сначала вычислим ответ для s_n, затем для $s_{n-1} \cdot s_n$, затем $s_{n-2} \cdot (s_{n-1} \cdot s_n)$ и так далее.

Поскольку хранить произведение строк целиком не предоставляется возможным (итоговая строка слишком длинная, чтобы уместиться в памяти программы), будем хранить только некоторую базовую информацию о ней, которая позволит нам посчитать ответ:

1. Длина наибольшей подстроки, состоящей из одного символа (ответ на задачу),
2. Состоит ли строка целиком из одного символа,
3. Левый и правый символы строки,
4. Длина префикса, состоящего только из первого символа строки, и длина суффикса, состоящего только из последнего символа строки.

Легко видеть, что если мы знаем эти значения для $s_i \cdot \ldots \cdot s_n$, то мы их можем вычислить для $s_{i-1} \cdot s_i \cdot \ldots \cdot s_n$ за $O(|s_{i-1}|)$.

Асимптотика: $O(\sum_{i=1}^{n} |s_i|)$.

Разбор задачи «Выбор гурмана»

Эта задача имеет несколько возможных решений.

Одно из них такое — завести структуру данных СНМ (совокупность независимых множеств) для всех $n + m$ блюд и объединить все блюда, которые должны быть оценены одним и тем же числом согласно таблице (объединяем блюда i и $n + j$ если a_{ij} равно "=").

Затем создадим граф и соотнесем каждому множеству в СНМ свою вершину. Переберем все i и j и добавим направленное ребро между вершинами, которые представляют множества, которым принадлежат i и j. Это ребро должно иметь направление от "большей" вершины к "меньшей".

В случае, если граф имеет цикл или петлю, легко увидеть, что ответа не существует. Иначе присваиваем каждой вершине такой номер, чтобы у всех её соседей (куда ведёт направленное ребро от данной вершины) были меньшие номера. Это можно сделать с использованием метода динамического программирования по ацикличному графу.

Асимптотика: $O(nm)$.

Разбор задачи «Ася и котята»

В этой задаче у нас есть описание процесса объединения СНМ (системы непересекающихся множеств) из $n - 1$ шагов, который объединяет n одноэлементных множеств в одно n-элементное множество. Нам надо упорядочить исходные одноэлементные множества таким образом, чтобы каждая пара множеств, которые мы объединяем на текущем шаге, была непосредственными соседями (то есть между ними не было никаких других множеств).

Эту задачу можно решить за $O(n \log n)$ или за $O(n\alpha(n))$ (где $\alpha(n)$ — обратная функция Аккермана) с использованием стандартной структуры данных "система непересекающихся множеств", если хранить корректный порядок для элементов в множестве в виде односвязного списка.

Самое простое решение, где СНМ использует ранговую эвристику по размеру множества:

- храним соответствие от каждого элемента к номеру его текущего множества, и обратное соответствие от номера множества к списку его элементов.
- когда нам нужно объединить два множества A и B, мы делаем меньшее множество частью большего и обновляем соответствия, присваивая новый идентификатор множества для элементов за $O(\min(|A|, |B|))$ и объединяя списки (это можно сделать за $O(1)$ или за $O(\min(|A|, |B|))$).

Это даёт сложность по времени $O(n \log n)$: каждый элемент меняет своё множество не более чем $logn$ раз, потому что каждое объединие множеств ведёт к увеличению размера меньшего множества как минимум в два раза.

Чтобы получить сложность $O(n\alpha(n))$, следует использовать СНМ со сжатием путей и ранговой эвристикой, и объединение списков должно быть реализовано за $O(1)$, но этого не требовалось для полного решения задачи.

Асимптотика: $O(n \log n)$ или $O(n\alpha(n))$, где $\alpha(n)$ — обратная функция Аккермана.

Разбор задачи «Самая опасная акула»

Решим задачу с использованием метода динамического программирования.

Пусть dp_i минимальная стоимость, за которую можно уронить ровно i первых доминошек. Если i доминошка упала влево, тогда $dp_i = \min(dp_j + c_i)$ по таким j, что при падении i доминошки упадут все доминошки с $j + 1$ до i. Иначе, какая-то другая доминошка упала вправо и уронила доминошку i. Тогда $dp_i = min(dp_{j-1} + c_j)$ по таким j, что при падении вправо j доминошка уронит i и не уронит $i + 1$. Такое решение работает за $O(m^2)$.

Нам нужно ускорить это решение. Для этого для каждого i нам надо найти все такие j, что при падении i доминошки влево, она уронит доминошку j. Такие j образуют непрерывный отрезок, поэтому мы можем не хранить явно все множество, а только его левую границу, правая граница будет

равна i. Эти левые границы можно посчитать при помощи стека за линейное время. Аналогично для каждого j посчитаем такое максимальное i, что при падении вправо j доминошка роняет все доминошки до i но не роняет $i+1$ доминошку.

Другая часть решения, которую мы должны оптимизировать — это получение минимального элемента на подотрзеке dp. Это легко сделать при помощи дерева отрезков, однако тогда наше решение будет работать за $O(m \log m)$, что слишком медленно. Чтобы сделать это быстрее, мы можем снова использовать стек. Будем хранить стек растущих значений dp_i (или $dp_i + c_i$, в зависимости от случая). Поскольку отрезки, на которых мы берем минимум, являются вложенными или непересекающимися, мы всегда можем убрать из стека все значения после оптимального для каждого запроса. Используя метод амортизационного анализа, можно увидеть, что такое решение работает за линейное время.

Асимптотика: $O(m)$.