Разбор задач краевой олимпиады 2020

Автор разбора: Степан Остапенко (telegram @flaax).

Разбор задачи «Гидра»

Автор задачи: Сергей Поспелов

Средний балл: 57.94 (мл.)

Для начала следует разобрать несколько крайних случаев: 1) $n < k$ – ответ 0; 2) $n \geq k = m$ – ответ -1. В остальных случаях предположим, что мы хотим сделать x ударов. Тогда условие того, что мы можем это сделать: $n - (k - m) \cdot (x - 1) \geq k$ (после $x - 1$ ударов мы срубили $(x - 1) \cdot (k - m)$ голов, а для x-го нам необходимо наличие хотя бы k голов). Путем несложных преобразований получаем $x \leq \frac{n-m}{k-m}$, а значит ответ равен $\lfloor \frac{n-m}{k-m} \rfloor$, так как $x \in \mathbb{Z}$.

Сложность решения $\mathcal{O}(1)$.

Разбор задачи «Мультиварка»

Автор задачи: Степан Остапенко

Средний балл: 69.41 (мл.)

В задаче требуется просто расставить блюда в правильном порядке, а затем повторить процесс, описанный в условии. Пусть ans_i – время, за которое Сережа приготовит i первых блюд. Тогда $ans_0 = 0; ans_{i+1} = max(ans_i, a_i) + b_i + k$ (ждем, когда будут готовы i первых блюд, и когда можно будет начать готовить $(i+1)$-е, готовим его, чистим мультиварку). Итоговый ответ – ans_n.

Сложность решения: $\mathcal{O}(n)$.

Разбор задачи «Званый ужин»

Автор задачи: Григорий Базилевич

Средний балл: 41.18 (мл.)

Заметим, что $x \leq \min a_i$, поскольку, в противном случае, существует коробка, в которой не хватит пончиков для того, чтобы каждый получил хотя бы один. Тогда для решения достаточно просто перебрать все возможные $x \leq \min a_i$ и посчитать, сколько пончиков получит Аня (по формуле $a_1 \bmod x + a_2 \bmod x + \cdots + a_n \bmod x$).

Решение работает за $\mathcal{O}(n \cdot \min a_i)$.

Разбор задачи «Уроки Литературы»

Автор задачи: Артем Захаренко

Средний балл: 14.12 (мл.), 63.85 (ст.)

Для того, чтобы произведение всех чисел на отрезке равнялось 1, все числа на отрезке должны быть равны 1. Для того, чтобы произведение чисел на отрезке равнялось 0, хотя бы одно из чисел на отрезке должно быть равно 0. Отсюда нетрудно догадаться, что нам необходимо и достаточно заполнить единицами все отрезки, на которых произведение равно 1, а потом проверить, что на всех остальных отрезках произведение действительно равно 0.

Для полного решения задачи осталось написать достаточно эффективную реализацию вышеперечисленных действий. Это можно сделать с помощью метода сканирующей прямой ($\mathcal{O}(n)$ или $\mathcal{O}(n \log n)$), дерева отрезков или std::set ($\mathcal{O}(n \log n)$).

Разбор задачи «Настя, Юля и длинные числа»

Автор задачи: Степан Остапенко

Средний балл: 5 (мл.), 29.65 (ст.)

Заметим, что $s = a \cdot (1 + 10^{len(a)} + 10^{2len(a)} + \cdots + 10^{(n-1) \cdot len(a)})$. Тогда посчитаем остаток от деления числа a на 998244353 (обозначим его за x) и остаток от деления вышеуказанной суммы на 998244353 (обозначим его за y). В таком случае ответ будет равен $(x \cdot y) \bmod 998244353$. Число x можно посчитать просто перебрав последовательно все цифры в числе a. Число y можно посчитать двумя способами:

1) Вышеуказанная сумма представляет собой геометрическую прогрессию с первым членом 1 и знаменателем $q = 10^{len(a)}$. В таком случае y можно посчитать по формуле $\frac{q^n-1}{q-1}$, выполняя все операции[1] по модулю 998244353 с помощью двоичного возведения в степень[2].

2) Пусть мы хотим узнать число y для некоторого n, обозначим его как $y(n)$. Если n четно, $y(n) = y(\frac{n}{2}) \cdot (10^{\frac{n}{2} \cdot len(a)} + 1)$. Если n нечетно, $y(n) = y(n-1) \cdot 10^{len(a)} + y(1)$. Эти соображения позволяют быстро считать $y(n)$ рекурсивным образом.

Сложность поиска числа y в первом случае составляет $\mathcal{O}(\log n)$, так как мы просто несколько раз пользуемся двоичным возведением в степень. Слож-

[1] `https://e-maxx.ru/algo/reverse_element`

[2] `https://e-maxx.ru/algo/binary_pow`

ность поиска числа y во втором случае составляет $\mathcal{O}(\log^2 n)$, так как мы считаем $\mathcal{O}(\log n)$ значений функции ans и для подсчета каждого значения один раз пользуемся двоичным возведением в степень. Итоговая сложность составляет $\mathcal{O}(\sum len(a) + t \log n)$ или $\mathcal{O}(\sum len(a) + t \log^2 n)$.

Разбор задачи «Приключения на высоте»

Автор задачи: Сергей Поспелов
Средний балл: 17.65 (мл.), 25.88 (ст.)

Выпишем в 2 отдельных массива все переходы вправо и все переходы вверх. Тогда заметим, что если у нас есть путь из некоторого количества подряд идущих переходов, то все переходы вправо, использующиеся в этом пути, в своем массиве тоже идут подряд, так же как и все переходы вверх. Помимо этого, получается, что если мы хотим сконструировать какой-то путь, мы должны выбрать некоторое количество переходов вправо, идущих подряд, и добавить к ним недостающее количество переходов вверх, тоже идущих подряд.

Ключевая идея заключается в том, что выбирать переходы вправо и вверх можно независимо друг от друга. Поэтому для решения задачи достаточно перебрать сколько переходов вправо мы возьмем в ответ (допустим, это число p, $0 \le p \le k$), выбрать p последовательных переходов вправо так, чтобы их суммарная длина была максимальна, и выбрать $k-p$ последовательных переходов вверх так, чтобы их суммарная длина была максимальна, просуммировать полученные длины и выбрать максимум по длинам среди всех $0 \le p \le k$.

Сложность решения $\mathcal{O}((n+m)k)$.

Разбор задачи «Подготовка к Лин. Алгебре»

Автор задачи: Максим Няшин
Средний балл: 26.73 (ст.)

Счет Максима при определенном n будет равен $n!$, поскольку каждый из n листов побывает на своем месте ровно $(n-1)!$ раз (зафиксируем положение какого-то листа, тогда остальные можно переставить $(n-1)!$ способами). То есть в задаче требуется уметь быстро находить наименьшее n такое, что $n! \vdots M$, для какого-то произвольного M.

Разложим M на простые множители. Для того, чтобы $n!$ делилось на M, нужно, чтобы степень вхождения каждого простого числа в $n!$ была не мень-

ше, чем степень его вхождения в M, поэтому можно найти искомое n двоичным поиском, однако, такое решение не набирает полный балл.

Для более быстрого решения следует, во-первых, решить задачу для каждого простого множителя отдельно, а потом просто выбрать максимум из всех ответов, а во-вторых, быстрее решать саму задачу поиска подходящего n. Пусть сейчас мы решаем задачу для множителя p, который входит в M в степени α. Тогда будем пробовать в качестве n числа вида $p, 2p, 3p, \ldots$ и на каждом шаге поддерживать в какой степени сейчас число p входит в $n!$. Если это число не меньше α, мы нашли ответ, иначе продолжаем дальше. Очевидно почему подходящее нам n всегда кратно p (пусть это не так, тогда уменьшим n до ближайшего числа, кратного p – ничего не сломается, а ответ станет лучше). Еще стоит упомянуть, что для полного решения требуется раскладывать число M на простые множители с помощью решета Эратосфена[3].

Осталось понять, за сколько это все работает. Сложность решета Эратосфена в простейшей реализации составляет $\mathcal{O}(C \log\log C)$, где $C \approx 10^7$, в нашем случае. Разложение числа на простые множители решетом работает за $\mathcal{O}(\log M)$, так как каждый раз раскладываемое число уменьшается хотя бы в 2 раза. Поиск подходящего n тоже работает за $\mathcal{O}(\log M)$, поскольку на каждом шаге степень вхождения числа p в $n!$ увеличивается хотя бы на 1, при этом $\alpha \leq \log M$ (аналогичные рассуждения как и в разложении). Итоговая сложность решения составляет $\mathcal{O}(C \log\log C + S \cdot \log M)$.

Разбор задачи «xOriSthEBesT Strike Back»

Автор задачи: Степан Остапенко
Средний балл: 4.35 (ст.)

Для начала разберем решение первой подзадачи. Так как каждое число в таблице не превышает 200, максимальный возможный счет не превышает 255 (каждое число в таблице укладывается в 8 бит памяти, значит и ответ тоже). Пользуясь этим, решим задачу методом динамического программирования[4]: $dp_{i,j,k} = 1$, если мы можем дойти из начальной клетки до клетки в i-й строке и j-м столбце, набрав при этом счет равный k; или $dp_{i,j,k} = 0$ в противном случае. С помощью этой динамики можно найти максимальный счет, который можно набрать на пути в правую нижнюю клетку. Сохраняя массив предков, можно восстановить сам путь.

[3] https://wiki.algocode.ru/index.php?title=Решето_Эратосфена

[4] `https://ejudge.lksh.ru/archive/2013/07/Bprime/texts/05_dynprog.pdf`

Во второй подзадаче можно просто перебрать все возможные пути (их $m + 1$) и выбрать оптимальный.

Решения третьей, четвертой и пятой подзадачи представляют собой оптимизации решения для первой подзадачи с учетом того, что нам необязательно находить оптимальный ответ. Ключевая идея в решении задачи заключается в том, что мы можем сжать числа в таблице, оставив от каждого из них только самую важную информацию, и затем запустить на таблице со сжатыми числами динамику из первой подзадачи. Так что теперь нам нужно научиться искать важную информацию в каждом числе и научиться сжимать числа.

Заметим, что на k-й бит в набранном счете влияют только значения k-х битов в числах таблицы. Это позволяет нам проверить, существует ли ответ, в котором k-й бит равен 1. Выпишем в отдельную таблицу k-е биты у всех чисел и запустим на этой таблице динамику из первой подзадачи. Тогда $dp_{n,m,1}$ – ответ на наш вопрос. Перебрав все возможные k ($0 \le k \le 30$), мы можем найти самый старший бит в оптимальном ответе, обозначим его за b (этого достаточно для решения третьей подзадачи, так как любые числа, у которых совпадает старший бит, отличаются не более чем в 2 раза, поскольку $1 + 2 + 4 + 8 + \cdots + 2^{p-1} = 2^p - 1$).

Теперь, когда мы знаем старший бит в оптимальном ответе, мы можем удалить у чисел все биты, которые старше, чем b, поскольку они все равно ни на что не влияют. Дальше у каждого числа оставляем несколько (допустим, w) самых значимых битов и удаляем все остальное. Числа, образованные этими w битами и есть те сжатые числа, на которых можно запустить динамику из первой подзадачи. Причем, чем больше w, тем ближе к оптимальному будет ответ, который мы найдем. Можно доказать, что для того, чтобы выполнялось условие $pa \cdot 2^q \ge ja \cdot (2^q - 1)$ (pa – ответ участника, ja – ответ жюри), достаточно взять $w = q + 1$ бит. То есть для третьей подзадачи можно взять $w = 2$, для четвертой $w = 5$ и для пятой $w = 11$.

Итоговая сложность решения составляет $\mathcal{O}(nmC)$ для первой подзадачи ($C \approx 256$), $\mathcal{O}(m^2)$ для второй подзадачи и $\mathcal{O}(nm \log C \cdot 2^w)$ или $\mathcal{O}(nm(\log C + 2^w))$ для третьей-пятой подзадачи.