Руководитель олимпиады: Шульгина Галина Михайловна

Председатель жюри: Базилевич Григорий
Состав жюри: Кориненко Матвей
Коробко Семен
Пупков Кирилл
Девятериков Иван

Спонсоры олимпиады: XSOLLA
PARMA Technologies Group
Knoema

Отдельная благодарность тем, кто протестировал задачи: Поспелов Сергей
Субботин Игорь
Перескокова Марина

Задача A. Задание на дом

Автор и разработчик: Семён Коробко

Давайте переберём все x от -1000 до 1000 для каждого тестового случая. Если $ax^5 + bx^4 + cx^3 + dx^2 + ex + f = 0$, то выведем данный x.

Задача B. Маленькая задачечка

Автор и разработчик: Семён Коробко

Заметим, что минимальное число ans, которому будет равно количество золота для каждого мешочка, равно $ans = \text{НОК}(a_i)$ для всех i, где НОК — наименьшее общее кратное для множества чисел.

Тогда ответ будет равен

$$\sum_{i=1}^{n} \frac{ans}{a_i}$$

Задача C. Ваня и формочки

Авторы: Иван Девятериков, Матвей Кориненко
Разработчик: Матвей Кориненко

Переформулируем условие. От нас требуется для каждого числа от 1 до n найти количество делителей среди чисел s_i. Это можно сделать просто пробегаясь по массиву $[0, n]$ с шагом s_i и прибавляя к значению элемента этого массива единицу.

Но, так как значения s_i могут многократно повторяться, то необходимо как-то оптимизировать решение, так как в чистом виде оно может иметь сложность $O(n \times m)$, что будет неоптимальным.

Для ускорения решения задачи можно использовать $std :: map$ в C++ или $dict()$ в Python 3, где ключом будет являться размер формы, а значением по этому ключу — количество таких формочек. Тогда решение будет иметь сложность $O(m \times log(n))$.

Задача D. Фокус с зеркалом

Автор и разработчик: Кирилл Пупков

Назовем множество позиций кубиков правильным, если существует точка, такая, что относительно нее это множество симметрично (то есть для него ответ это 0). Ясно, что если размер наибольшего правильного множества равен M, то ответ на задачу $N - M$. Действительно, для правильного множества кубиков добавлять не придется, а для всех остальных придется добавить ровно по одному симметричному.

Наибольшее правильное множество можно найти за $O(2^n \cdot n)$ перебором или за $O(n^3)$ методом двух указателей. Но квадратичный алгоритм проще: достаточно просто найти точку, такую, что она является серединой наибольшего количества отрезков. Тогда она будет центром искомого множества. Это можно легко сделать, используя std::map, std::unordered_map или массив. В зависимости от выбранной структуры решение набирает от 50 до 100 баллов. При этом нужно не забывать учитывать, что центр не может находиться внутри кубика.

Задача E. Генеральная репетиция

Автор и разработчик: Григорий Базилевич

Для решения первой и второй подзадач можно перебрать все возможные подмножества человек средней группы и проверить их средние погрешности по танцу. Асимптотика: $O(2^k \times n \times k)$.

Для дальнейшего решения нам необходимо доказать пару лемм. Сначала заметим, что для любого $\forall i \; \forall j \quad a_{ij} \geqslant b_j$.

Лемма 1 Погрешность удара k человек можно заменить средним арифметическим погрешностей ударов каждого человека.

$$\left|\frac{a_1 + a_2 + ... + a_k}{k} - b\right| = \left|\frac{(b - b + a_1) + ... + (b - b + a_k)}{k} - b\right| =$$
$$= \left|\frac{b \times k + (a_1 - b) + ... + (a_k - b))}{k} - b\right| = \left|b + \frac{(a_1 - b) + ... + (a_k - b)}{k} - b\right| =$$
$$= \left|\frac{(a_1 - b) + ... + (a_k - b)}{k}\right| = \frac{|(a_1 - b) + ... + (a_k - b)|}{k} = \frac{(a_1 - b) + ... + (a_k - b)}{k}$$

Лемма 1 доказана.

Лемма 2 Средняя погрешность танца k человек равна среднему арифметическому средних погрешностей танца каждого человека.

$$\frac{\left|\frac{a_{11} + ... + a_{k1}}{k} - b_1\right| + ... + \left|\frac{a_{1n} + ... + a_{kn}}{k} - b_n\right|}{n} =$$
$$= \frac{\left(\frac{(a_{11} - b_1) + ... + (a_{k1} - b_1)}{k}\right) + ... + \left(\frac{(a_{1n} - b_n) + ... + (a_{kn} - b_n)}{k}\right)}{n} =$$
$$= \frac{((a_{11} - b_1) + ... + (a_{k1} - b_1)) + ... + ((a_{1n} - b_n) + ... + (a_{kn} - b_n))}{kn} =$$
$$= \frac{((a_{11} - b_1) + ... + (a_{1n} - b_n)) + ... + ((a_{k1} - b_1) + ... + (a_{kn} - b_n))}{kn} =$$
$$= \frac{\left(\frac{(a_{11} - b_1) + ... + (a_{1n} - b_n)}{n}\right) + ... + \left(\frac{(a_{k1} - b_1) + ... + (a_{kn} - b_n)}{n}\right)}{k}$$

Лемма 2 доказана.

Для решения третьей подзадачи необходимо было заметить, что если поставить в танец человека, у которого погрешность по танцу будет $\geqslant 0$, то мы уже не сможем получить среднюю погрешность танца среди всех человек $\leqslant d = 0$. Значит необходимо вывести только тех, у кого танец исполнен идеально.

Для решения четвертой и пятой подзадач воспользуемся жадными идеями — будем добавлять людей, пока средняя погрешность танца $\leqslant d$. Для этого в четвертой подзадаче будем на i-м шаге выбирать человека для добавления в танец за линейное время, а в пятой подзадаче отсортируем массив и пройдем по нему. Итоговая асимптотика: $O(k^2)$ или $O(k \log k)$ в зависимости от реализации.

Задача F. Ваня и быстрая игра

Автор: Иван Девятериков
Разработчик: Семён Коробко

Во-первых заметим, что $a_i >$ n мы точно никогда не используем. Действительно, так как при решении для какого-то i мы используем числа, которые меньше данного i, то мы будем использовать только числа $a_i \leqslant i \leqslant n$.

Выкинем все $a_i > n$. Пусть m теперь — это количество оставшихся чисел.

Давайте создадим массив $left$ длины n+1, где предпосчитаем ближайшее слева a_i для каждого i.

Решение №1 Давайте сразу искать ответ для текущего i.

Давайте, пока $teck > 0$ и $left[i] > 0$ делать следующую вещь.

Заметим, что из текущего i мы будем вычитать $left[i] - min(teck, \frac{i}{left[i]})$ ходов. Если $teck < \frac{i}{left[i]}$, то присвоим $i = i - teck \times left[i]$ — это i и будет ответом. А иначе, присвоим из $teck = teck - \frac{i}{left[i]}$ и i = i mod left[i].

Заметим, что мы сделаем не больше log k операций, потому что каждое «иначе» уменьшает i хотя бы в два раза, таким образом не более чем через log k ходов left[i] будет равен 0.

Данный алгоритм работает за $\mathcal{O}(n \log k)$ по времени и $\mathcal{O}(n)$ по памяти.

Решение №2 Представим переходы $X \to X_{\texttt{новый}}$ в виде ориентированного графа. Заметим, что граф у нас ациклический (причем из каждой вершины выходит ровно одно ребро).

Значит наша задача сводится к тому, чтобы для каждой вершины найти потомка на расстоянии k. Давайте инвертируем все ребра и теперь нам нужно найти предка на расстоянии k.

Для этого можно воспользоваться техникой «двоичные спуски/подъёмы». Тогда решение будет работать за $\mathcal{O}(n \log n)$. Подробнее про эту технику можно почитать здесь: https://neerc.ifmo.ru/wiki/index.php?title=Метод_двоичного_подъёма

Но так как k фиксировано, мы можем для каждой вершины посчитать предка на расстоянии k, просто запустив DFS и поддерживая текущий путь до вершины. Попадая в вершину, мы сохраним ее предка на нужном расстоянии. Такое решение работает за $\mathcal{O}(n)$.

Замечание Хоть асимптотически решение с помощью DFS и быстрее, но на практике первое решение работает либо также, либо даже лучше.

Задача G. Забывчивый Матвей

Авторы: Матвей Кориненко, Семен Коробко, Кирилл Пупков
Разработчик: Семён Коробко

Заметим, что всего возможных вариантов пароля может быть M^N (так как длина пароля — N символов, на месте каждого из которых может быть один из M символов множества A).

Но поскольку число комбинаций может не быть кратным числу K, а ответ все равно необходимо взять по модулю MOD, то это число мы можем домножить на K^{MOD-2} и итоговое также взять по модулю MOD. Это работает как следствие из малой теоремы Ферма ($a^{p-1} \equiv 1 \mod p$):

$$\frac{M^N}{K} = M^N \times \frac{1}{K} = M^N \times K^{-1} \equiv M^N \times K^{MOD-2} \equiv ans \mod MOD$$

Для полного решения этой задачи необходимо было воспользоваться быстрым возведением в степень, в процессе которого необходимо брать число по модулю MOD.