

Задача А. Цвет глаз

Идея: Николай Дубчук

Разработка: Николай Дубчук

Жюри предлагает решать задачу так. Сначала напишем функцию `isSame`, которая принимает на вход две строки s_1 и s_2 и проверяет, верно ли, что строка s_1 , поданная на вход, может обозначать цвет, записанный в строку s_2 . На C++ такую функцию можно реализовать так:

```
bool isSame(string s_1, string s_2) {  
    return s_1 == "." || s_1 == s_2;  
}
```

Теперь напишем функцию, которая по цветам глаз родителей определит, какой цвет глаз должен быть у ребёнка. Оказывается, определить этот цвет можно однозначно согласно правилам, написанным в условии. Реализация на C++:

```
string getChildColor(string s_1, string s_2) {  
    if (s_1 == "brown" || s_2 == "brown")  
        return "brown";  
    if (s_1 == "green" || s_2 == "green")  
        return "green";  
    return "blue";  
}
```

Используя эти функции, уже легко решить задачу. Считаем все три цвета глаз. Тремя вложенными циклами переберём цвета глаз: сначала первого родителя, потом второго и, наконец, ребёнка. Перебирать будем по лексикографическому возрастанию, то есть в порядке `[blue, brown, green]`. Внутри самого внутреннего цикла проверим тремя вызовами функции `isSame`, что цвета глаз соответствуют считанным строкам. Наконец, проверим, что цвет глаз ребёнка совпал с цветом, возвращаемым функцией `getChildColor`, и лишь если все эти проверки прошли, выведем три цвета глаз.

Задача В. Большая мировая политика

Идея: Роман Первутинский

Разработка: Роман Первутинский

Рассмотрим страны, к которым есть территориальные претензии у страны i . Они либо целиком лежат внутри минимального прямоугольника страны i , либо пересекают его границу, либо не пересекаются с ним. Нам нужно посчитать суммарное количество первых и вторых стран.

Заметим, что периметр минимального прямоугольника страны i не больше, чем учетверённое количество клеток, принадлежащих стране i , поскольку по клеткам страны i можно пройти от левой до правой границы и от верхней до нижней границы минимального прямоугольника. Поэтому сумма периметров всех минимальных прямоугольников — $O(nm)$.

Поймём, как учесть страны, целиком лежащие внутри минимального прямоугольника. Выберем для каждой страны клетку-представителя и посчитаем количество представителей на всех префиксных прямоугольниках. Таким образом, мы можем найти количество стран, представители которых лежат внутри минимального прямоугольника.

Осталось учесть страны, которые пересекают границу минимального прямоугольника, но представители которых лежат вне него. Для этого мы можем пройти по клеткам вдоль границы и посчитать все такие цвета. Получаем решение за $O(nm)$.

Задача С. Головоломка

Идея: Роман Гусарев

Разработка: Екатерина Ведерникова

Разрешённые операции позволяют независимо переставлять элементы внутри каждого столбца, при этом количество нулей и единиц в столбце не меняется.

Значит, в каждом столбце мы можем выбрать, какие строки будут иметь значение 1, а какие 0, но общее число единиц фиксировано.

Если в столбце j находится k единиц, то максимум строк, совпадающих в этом столбце, равен $\max(k, n - k)$ — можно сделать либо все эти строки с единицей, либо с нулём.

Чтобы строки были полностью одинаковыми, они должны совпадать во всех столбцах одновременно. Поэтому общее количество одинаковых строк не может превышать минимальное значение среди всех столбцов.

Достаточно посчитать количество единиц в каждом столбце и взять минимум из значений $\max(k, n - k)$. Это и будет ответом.

Алгоритм работает за $O(nm)$ и укладывается в ограничения.

Задача D. Новые шарики

Идея: Иван Сафонов

Разработка: Егор Юлин

Для решения данной задачи необходимо построить n деревьев, в которых не будет пересечений по ребрам.

Покажем, что если какие-то два дерева будут иметь хотя бы одно общее ребро, то будет существовать дерево, которое можно получить из обоих деревьев за $n - 1$ операций или меньше.

Покажем последовательность шагов для первого дерева:

1. Во время перестройки дерева будем выбирать такие (c, d) , что в текущем дереве нет такого ребра, а во втором дереве такое ребро есть. Тогда добавим ребро (c, d) в текущее дерево. У нас в графе появился цикл, заметим, что на таком цикле будет хотя бы одно ребро, которого нет во втором дереве, тогда в качестве (a, b) возьмём это ребро;
2. Будем так повторять, пока у нас не станет n общих рёбер между деревьями;

После этого у нас различие не более чем в $n - 1$ рёбер между вторым деревом и деревом из запроса, а тогда мы можем такими же операциями преобразовать второе дерево в только что полученное, а значит мы нашли дерево, которое можно получить из двух различных, что не позволяет нам однозначно восстановить, из какого дерева оно было получено.

Заметим, полный граф на $2n$ вершинах будет содержать $\frac{2n \cdot (2n - 1)}{2}$, а дерево на $2n$ вершинах содержит $2n - 1$ рёбер, тогда максимальное количество деревьев, которое мы можем получить — $\frac{2n \cdot (2n - 1)}{2 \cdot (2n - 1)} = n$. То есть мы должны распределить все рёбра полного графа между деревьями.

Будем решать немного другую задачу, пусть мы хотим разбить полный граф на непересекающиеся гамильтоновы пути, это можно сделать следующим образом:

1. Построим путь $1, 2, 2n, 3, 2n - 1, \dots, n + 1$;
2. Все остальные пути будут получены из первого, сдвигом всех вершин по модулю $2n$ на значения от 1 до $n - 1$. Например, при сдвиге на 2 получим путь $2, 3, 1, 4, 2n, \dots, n + 2$.

Не сложно заметить, что такие пути не будут иметь пересечения по ребрам, а в объединении будут давать полный граф. И раз мы смогли разбить граф на пути, то мы смогли тем же самым образом разбить граф на деревья, которые будут являться бамбуком.

После постройки таких деревьев для ответа на запрос достаточно находить дерево, которое имеет хотя бы n общих рёбер с деревом из запроса.

Задача E. Опустошение массива

Идея: Павел Скобелин

Разработка: Павел Скобелин

Воспользуемся жадным алгоритмом для решения задачи, который будет заключаться в следующем: на очередном ходу выберем самый левый индекс i , такой что $a_i + a_{i+1} = x$, и удалим эту пару элементов. Утверждается, что Ксюша может пройти уровень если и только если она может пройти уровень такой жадной стратегией.

Для начала разберем более простую часть — реализация жадной стратегии. Это можно сделать, обрабатывая массив слева направо: будем поддерживать стек еще не удаленных элементов. Если очередное число массива в сумме с последним элементом стека дает x , то удалим последнее число

из стека, иначе — добавим очередное число в стек. Несложно заметить, что этот алгоритм реализует жадный алгоритм из решения.

Осталось показать, что этот жадный алгоритм корректен. Очевидно, что если в конце стек оказался пуст, то массив опустошить можно — действительно, ведь мы достигли пустого массива удалением соседних (в моменте) элементов. Покажем следствие в другую сторону. Пусть в решении был выбран индекс j , в то время как жадный алгоритм выбрал индекс i такой, что $i < j$. Разберем 2 случая:

1. $j = i + 1$. Тогда $a_i + a_{i+1} = x$, и $a_{i+1} + a_{i+2} = x$. Из этого следует, что $a_i = a_{i+2}$ — значит, что массив будет одинаковым при удалении i -го и j -го элемента, значит порядок операций можно поменять.
2. $j > i + 1$. В таком случае несложно показать, что можно поменять порядок этих операций, и решение не ухудшится.

Таким образом, для решения было достаточно реализовать описанный выше жадный алгоритм.

Задача F. Шоколадки Фибоначчи

Идея: Ильдар Гайнуллин

Разработка: Ильдар Гайнуллин

Рассмотрим состояния шоколадок независимо друг от друга. Каждая шоколадка в любой момент времени описывается парой множеств

$$(L, R),$$

где L — множество возможных состояний, в которые может перейти Алиса, а R — множество состояний, в которые может перейти Боб.

Любое состояние игры аналогично описывается парой таких множеств. Таким образом, каждая позиция игры имеет вид

$$G = \{L \mid R\}.$$

Утверждается, что для каждой шоколадки, существует рациональное число, эквивалентное игре с этой шоколадкой в следующем смысле: если сумма рациональных чисел, соответствующих всем шоколадкам, строго больше нуля, то выигрывает Алиса; в противном случае выигрывает Боб.

Рассмотрим правило суммирования двух таких игр. Пусть

$$G_1 = \{L_1 \mid R_1\}, \quad G_2 = \{L_2 \mid R_2\}.$$

Тогда их сумма определяется как

$$G_1 + G_2 = \{L_1 + G_2, G_1 + L_2 \mid R_1 + G_2, G_1 + R_2\}.$$

Для сравнения двух игр x и y достаточно проверить, верно ли неравенство $x \leq y$. Утверждается, что $x \leq y$, если не существует такого хода Алисы $x^L \in L_x$, что $y \leq x^L$, и не существует такого хода Боба $y^R \in R_y$, что $y^R \leq x$.

Заметим, что при анализе игры от нас фактически требуется знать лишь наибольший элемент множества L и наименьший элемент множества R . Более того, чтобы найти числовое значение некоторой определённой игры x , значение которой является целым, достаточно уметь проверять сравнения вида $x \leq n$ и $x \geq n$, где

$$n = 1 + 1 + 1 + \dots + 1$$

— сумма n единичных игр.

Если значение игры не является целым, можно доказать, что оно всегда является рациональным числом со знаменателем, равным степени двойки. Поэтому значение можно последовательно удваивать (прибавляя игру к самой себе), пока оно не станет равным некоторому натуральному числу n .

Сравнения и вычисление суммы можно реализовать с использованием мемоизации независимо для каждой шоколадки.

Операция удаления всех частей одной шоколадки может быть описана как переход из текущего состояния s данной игры для одной шоколадки в состояние 0 (то есть $0 \in L$, если шоколадка принадлежит Алисе, и $0 \in R$ — в противном случае).

На самом деле, данная игра может быть описана как красно-синий Хакенбуш — игра Хакенбуш, в которой оба игрока могут на своём ходу удалить любое ребро из дерева, оставляя только компоненту связности, содержащую корень, — но с тем отличием, что рёбра покрашены в синий и красный цвета, причём Алиса может удалять только синие рёбра, а Боб — только красные. Для каждой шоколадки из корня выходит ровно одно ребро того цвета, кому принадлежит шоколадка.

После вычисления чисел, соответствующих каждой шоколадке (что можно предварительно посчитать локально), можно заметить, что число каждой шоколадки в данных ограничениях имеет вид

$$\pm \frac{a}{2^b},$$

где a, b — небольшие натуральные числа (не больше 50).

Чтобы посчитать количество подмножеств с положительной суммой среди чисел такого вида, можно воспользоваться поразрядной динамикой по двоичным разрядам.

Задача G. Магический ритуал

Идея: Николай Ведерников

Разработка: Николай Ведерников

Для решения задачи нужно минимизировать стоимость перестановки массива в неубывающий порядок.

Обмен элементов на позициях с разностью 2 имеет нулевую стоимость. Значит, элементы на чётных позициях можно переставлять между собой бесплатно, и то же самое верно для нечётных позиций. Любой обмен, меняющий чётность позиции элемента, стоит хотя бы 1.

Отсортируем массив. Посмотрим, какие значения должны стоять на чётных и нечётных позициях в отсортированном массиве.

Разобьём элементы исходного массива на два мультимножества: элементы с чётных позиций и элементы с нечётных позиций. Затем пройдём по отсортированному массиву и будем забирать нужные элементы из соответствующего множества. Если элемент нашёлся, удаляем его.

После этого в множествах останутся элементы, стоящие не на своей чётности. Каждый платный обмен исправляет две такие ошибки, поэтому минимальное число платных операций равно половине количества оставшихся элементов.

Это и есть ответ для данного теста.

Сортировка и работа с мультимножествами дают сложность $O(n \log n)$ на тест, что укладывается в ограничения.

Задача H. Плотина бобров

Идея: Павел Скобелин

Разработка: Павел Скобелин

Будем воспринимать лабиринт как дерево. Из того, что он уложен на таблице, будем использовать только то, что у каждой вершины не более чем 4 соседа.

Для решения докажем вспомогательную лемму. Рассмотрим любое ребро $u - v$. Тогда запустим бобров в вершины u и v . Из соображений общности, пусть бобер из вершины u добрался до выхода не позже, чем бобер из вершины v . Разобьём граф на два множества: U — поддереву вершины u , включая её саму, и V — все остальные вершины. В таком случае утверждается 2 вещи:

1. В множестве вершин U есть хотя бы один выход.
2. Если запустить бобра в любую вершину множества U , то ближайший выход для него будет находиться в множестве U .

Первое утверждение понятно: ведь если все выходы в текущем дереве находятся в множестве V , то бобер, запущенный в вершину v , добрался бы до него строго раньше, ведь кратчайший путь из u в любую вершину множества V идет через вершину v , значит, имеет строго большую длину.

Докажем второе утверждение. Пусть кратчайший путь из вершины v до выхода имеет длину x . Значит, кратчайший путь из вершины v до выхода в множестве V имеет длину $y \geq x$. Значит, кратчайший путь из вершины v до выхода, который лежит в множестве U , имеет длину $z \geq y + 1 = x + 1$.

Таким образом, строго ближайший выход из вершины u лежит в множестве U . Тогда, пусть оптимальный выход для какой-то вершины $w \in U$ находится в множестве V . В таком случае он проходит через вершину u . Значит, этот путь можно было строго уменьшить, пойдя из вершины u в вершину из множества U . Получается, что мы уменьшили оптимальный ответ, то есть получили противоречие — значит, такого быть не могло, и оптимальный выход для w лежит в множестве U . Второй пункт леммы доказан.

Давайте придумаем решение для исходной задачи. Получается, что если выбрать какое-то ребро $u - v$, после запроса, используя первый пункт леммы, мы уменьшаем рассматриваемое множество до одного из множеств U или V . Заметим, что если мы уменьшили рассматриваемое множество до U — то в нем, используя второй пункт леммы, можно решать задачу рекурсивно, то есть выбрать ребро $u' - v'$, и проделать для них аналогичные рассуждения, ведь в этом множестве будет хотя бы один выход по первому пункту леммы.

Размер рассматриваемого множества каждый раз уменьшается. Когда он станет 1, то в единственной вершине этого множества будет находиться выход. Осталось понять, как нужно каждый раз выбирать ребро $u - v$.

Понятно, что хочется находить такое ребро $u - v$, что размер минимального из множеств U и V будет максимален. Давайте найдем центроид дерева. Для него будет выполняться условие, что все его дети размера не более $s/2$, где s — текущий размер дерева. Также можно понять, что будет хотя бы один ребенок, размер которого хотя бы $s/4$, так как у каждой вершины дерева не более четырех соседей. Значит, если рассмотреть ребро от наибольшего ребенка до центроида в качестве ребра $u - v$, то размер минимального поддерева будет от $s/4$ до $s/2$, то есть хотя бы $s/4$. Значит, выбрав такое ребро, мы будем откидывать хотя бы $1/4$ от размера дерева.

Понятно, что с помощью таких действий мы сделаем не более $\log_{4/3}(nm) \leq \log_{4/3}(nm) = 48$ действий, пока не получим множество размера 1. На деле такое решение легко укладывается в ограничение по запросам, ведь это довольно грубая оценка.

Задача I. Предсказывание позиции

Идея: Григорий Хлытин

Разработка: Григорий Хлытин

Обозначим исходный отсортированный по возрастанию массив ключей как $X = x_1, x_2, \dots, x_n$. Обозначим максимально допустимую ошибку предсказания позиции как \mathcal{E} .

Формальная задача состоит в том, чтобы разбить массив ключей на минимальное число частей так, чтобы для каждой части $[l, r]$ для набора $\{(x_i, y_i)\} = \{(x_l, 0) \dots (x_r, r - l)\}$ можно было провести некоторую прямую $y = k \cdot x + b$ которая удовлетворяла бы неравенствам ошибки для всех $i \in [l, r]$:

$$\begin{cases} k \cdot x_i + b \leq y_i + \mathcal{E} \\ k \cdot x_i + b \geq y_i - \mathcal{E} \end{cases}$$

Преобразуем систему следующим образом:

$$\begin{cases} b \leq (-x_i) \cdot k + (y_i + \mathcal{E}) \\ b \geq (-x_i) \cdot k + (y_i - \mathcal{E}) \end{cases}$$

Рассмотрим каждое неравенство системы по отдельности, временно убрав из рассмотрения \mathcal{E} :

$$b \leq (-x_i) \cdot k + (y_i) \tag{T1}$$

$$b \geq (-x_i) \cdot k + (y_i) \tag{T2}$$

Неравенство (T1) в координатах (k, b) описывает некоторое множество точек Q , которые ограничены сверху *нижней огибающей* прямых, а неравенство (T2) описывает множество точек P , которые ограничены снизу *верхней огибающей* прямых, заданных уравнениями $b = (-x_i) \cdot k + (y_i)$, с угловыми коэффициентами $-x_i$ и свободными коэффициентами y_i для всех $i \in [l, r]$.

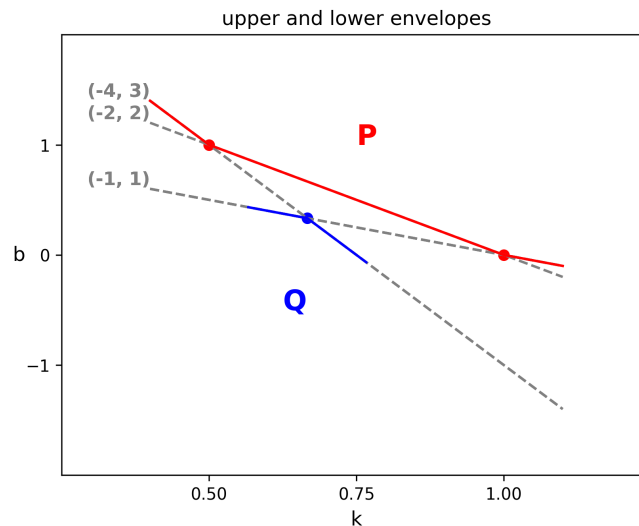


Рис. 1: Верхняя и нижняя огибающие для набора $\{(x_i, y_i)\} = (1, 1), (2, 2), (4, 3)$

Заметим, что система имеет решение тогда и только тогда, когда минимальное расстояние вдоль оси ординат между верхней и нижней огибающими не превосходит $2 \cdot \mathcal{E}$, так как неравенства в системе получаются параллельным переносом вдоль оси ординат неравенств $(T1)$ и $(T2)$ — нижней и верхней огибающих на \mathcal{E} вверх и \mathcal{E} вниз соответственно.

Таким образом, для решения нашей задачи, необходимо поддерживать текущее множество прямых, заданных парой коэффициентов $(-x_i, y_i)$, а также поддерживать верхнюю и нижнюю огибающие и минимальное расстояние вдоль оси ординат между ними, добавляя очередную i -ю прямую до тех пор, пока минимальное расстояние между огибающими не превосходит $2 \cdot \mathcal{E}$.

Задача J. Странная сумма

Идея: Ильдар Гайнуллин

Разработка: Ильдар Гайнуллин

Рассмотрим величину $W_{l,r}$ — сумму $w_{i,j}$ по всем $l \leq i \leq j \leq r$.

Аналогично рассмотрим $c_k = \max W_{l,l+k-1}$.

Разобьём x на слагаемые t_1, t_2, \dots, t_m от 1 до n , чтобы максимизировать $\sum_{i=1}^m c_{t_i}$. Иными словами, решим задачу о рюкзаке, где требуется набрать суммарный вес, равный x , максимизируя стоимость, при этом вес k -го предмета равен k , а стоимость — c_k .

Утверждается, что ответом на задачу является эта максимальная стоимость.

Действительно, для любого ответа a_1, a_2, \dots, a_n мы для всех значений y можем рассмотреть все максимальные по включению отрезки из чисел, не меньших y , и заметить, что $\sum_y \sum_k c_k$ не меньше значения $f(A)$ из задачи.

С другой стороны, можно, начиная с массива из нулей, для каждого взятого в рюкзак предмета веса k прибавить 1 на отрезке, на котором достигается максимум, увеличив ответ на величину не меньшую c_k .

Таким образом, мы доказали, что ответ не меньше и не больше результата задачи о рюкзаке, тем самым гарантируя их равенство.

Осталось решить задачу о рюкзаке при $x = 10^9$. Утверждается, что пока $x > n^2$, выгодно взять предмет с наибольшим $\frac{c_k}{k}$. Так как если предметов не равных k суммарно больше n (что следует, если их сумма $> n^2$), то если мы возьмем любые n из них в любом порядке, то среди них найдется подотрезок с суммой весов, делящейся на k (так как по принципу Дирихле две префиксные суммы будут иметь равный остаток по модулю k), а значит все предметы на этом отрезке можно заменить на какое-то количество предметов веса k , не уменьшив стоимость. С помощью деления с остатком можно посчитать, сколько предметов k мы таким образом найдем.

А для оставшего $x \leq n^2$ решим задачу о рюкзаке за $O(n^2x)$.

Задача К. Две косые дороги

Идея: Михаил Иванов

Разработка: Михаил Иванов

Если надо перейти только одну из двух улиц, то это можно сделать в отдалении от перекрёстка — то есть четыре из этих шести чисел будут $\ell_H, \ell_W, \ell_H, \ell_W$. Осталось найти два расстояния между противоположными частями. Две из частей являются острыми (или прямыми) углами α , а две — тупыми (или прямыми) углами $180^\circ - \alpha$. Пересечением улиц является параллелограмм $ABCD$ с углом $\angle DAB = \alpha$ при основании, и у этого параллелограмма известны высоты: расстояние между AB и CD равно ℓ_H , а между BC и DA — ℓ_W . Частями города же являются углы, вертикальные к углам параллелограмма.

Начнём с пары углов, вертикальных к $\angle DAB$ и к $\angle BCD$. Понятное дело, для них нельзя найти ничего лучше, чем AC . Как найти AC ? Можно явно построить этот параллелограмм, но мы лучше поступим так: опустим высоту CC' на продолжение отрезка AB за точку B . Тогда $AC = \sqrt{(AB + BC')^2 + C'C^2}$.

- $C'C^2 = \ell_H^2$ — тут всё просто;
- BC' можно найти, зная, что в треугольнике $\triangle BC'C$ нам известен прямой угол $\angle BC'C$, угол $\angle CBC' = \alpha$, а также катет $CC' = \ell_H$. Тогда другой катет равен $BC' = CC' \operatorname{ctg} \angle CBC' = \ell_H \operatorname{ctg} \alpha$;
- AB можно узнать, наоборот, опустив высоту BB' из B на AD . Тогда в треугольнике $\triangle BB'A$ мы знаем прямой угол $\angle BB'A$, угол $\angle B'AB = \alpha$, катет $BB' = \ell_W$, тогда гипотенуза равна $AB = \frac{BB'}{\sin \angle B'AB} = \frac{\ell_W}{\sin \alpha}$.

Отсюда

$$\begin{aligned} AC &= \sqrt{\left(\frac{\ell_W}{\sin \alpha} + \ell_H \operatorname{ctg} \alpha\right)^2 + \ell_H^2} = \\ &= \sqrt{\frac{\ell_W^2}{\sin^2 \alpha} + \frac{2\ell_H \ell_W \cos \alpha}{\sin^2 \alpha} + \frac{\ell_H^2 \cos^2 \alpha}{\sin^2 \alpha} + \ell_H^2} = \\ &= \frac{\sqrt{\ell_H^2 + 2\ell_H \ell_W \cos \alpha + \ell_W^2}}{\sin \alpha}. \end{aligned}$$

Аналогично можно найти $BD = \frac{\sqrt{\ell_H^2 - 2\ell_H \ell_W \cos \alpha + \ell_W^2}}{\sin \alpha}$. Однако тут есть тонкость: расстояние между углами, вертикальными к $\angle ABC$ и к $\angle CDA$, может быть равно BD , но иногда оно меньше — именно, если угол $\angle ABD$ тупой или если угол $\angle CBD$ тупой. Научимся работать в одном из этих случаев, в другом всё аналогично. Если, скажем, угол $\angle ABD$ тупой, то быстрее перемещаться не по отрезку BD , а по отрезку DD' , где D' — проекция D на AB . Так что ответ будет просто ℓ_H . Но как определить, тупой ли угол $\angle ABD$? Он тупой тогда и только тогда, когда $BC' > AB$. Ну давайте сравним — оба отрезка мы уже считали: $BC' > AB$ равносильно $\ell_H \operatorname{ctg} \alpha > \frac{\ell_W}{\sin \alpha}$ равносильно $\ell_H \cos \alpha > \ell_W$. Так что, если это неравенство верно, вместо подсчёта BD мы просто выводим ℓ_H . Аналогично, если $\ell_W \cos \alpha > \ell_H$, то ответ ℓ_W .

Задача Л. Система уравнений с хор

Идея: Демид Кучеренко

Разработка: Евгений Карпович

Задачу можно решать разными способами, обсудим один из них (этот способ является одним из самых быстрых для решения задачи, если бы числа x и y выбирались не случайно). Будем решать задачу с помощью метода ветвей и границ — рекурсивного перебора с отсечениями. Для этого будем набирать числа x и y от старших битов к младшим.

Мы можем реализовать функцию $get(i, x, y)$, которая фиксирует в числах x и y биты с номерами $i + 1, i + 2, \dots, 61$, а младшие i битов пока не рассматривает (они остаются нулевыми). Мы будем

фиксировать эти старшие биты таким образом, чтобы хог этих старших битов соответствовал числу b .

Для этого нам достаточно делать следующее:

- Если i -й бит числа b равен единице, то у нас есть два варианта переходов — $get(i - 1, x + 2^i, y)$ и $get(i - 1, x, y + 2^i)$.
- Если i -й бит числа b равен нулю, то у нас есть другие два варианта переходов — $get(i - 1, x + 2^i, y + 2^i)$ и $get(i - 1, x, y)$.

Если во время работы нашей рекурсивной функции мы окажемся в состоянии $i = -1$, в котором $x \cdot y = a$, то мы нашли ответ и можем выходить из рекурсии.

Далее применим ряд оптимизаций, которые помогут ускорить решение и не рассматривать случаи, которые гарантированно не приведут нас к ответу.

- Мы можем считать, что $x \geq y$ (нам не важен порядок переменных, поэтому мы можем работать в таком предположении). Для этого нам достаточно исключить переход $get(i - 1, x, y + 2^i)$, если $x = y$.
- Давайте научимся делать оценку на минимальное произведение, если мы как-то заполним младшие i битов в числах x и y , чтобы их хог был равен b . Пусть $c = (b \& (2^{i+1} - 1))$ — оставили младшие i битов числа b .

Утверждение: произведение чисел x и y будет хотя бы $(x + c) \cdot y$ (напомним, что мы считаем, что $x \geq y$).

Тогда если $(x + c) \cdot y > a$, то мы можем выйти из этого случая в рекурсии, так как гарантированно не найдём ответ.

- Аналогично мы можем сделать оценку на максимальное произведение. Но здесь нужно будет рассмотреть два случая:
 - Если $x = y$, то оценка делается несколько сложнее. Но мы очень мало раз находимся в этом случае, поэтому можем сделать эту оценку более слабой (больше, чем она есть на самом деле). В этом случае давайте оценим максимальное произведение как $(x + 2^{i+1} - 1) \cdot (y + 2^{i+1} - 1)$.
 - Если $x > y$. Аналогично обозначим $c = (b \& (2^{i+1} - 1))$ — младшие i битов числа b . Тогда мы можем оценить максимальное произведение как $(x + ((2^{i+1} - 1) \oplus c)) \cdot (y + 2^{i+1} - 1)$.

Если мы получим, что максимальное произведение (наша оценка сверху на него) меньше a , то мы сможем выйти из рассмотрения этого случая в нашем рекурсивном переборе.

Данное решение работает всё также корректно, но гораздо быстрее, чем наивный рекурсивный перебор, и проходит все тесты.

Задача М. Математический календарь

Идея: Рита Саблина

Разработка: Рита Саблина

В феврале невисокосного года 28 дней, это кратно 7, поэтому число всех дней недели одинаково — 4. Также заметим, что если год начинается в день с номером x , то первое декабря будет в день $(x - 2) \bmod 7$.

В декабре и январе 31 день, поэтому, например, если год начинается в понедельник, то в январе будет пять понедельников, пять вторников, пять четвергов, а всех осательных дней недели по четыре. Декабрь тогда начнется в субботу, и в декабре будет по пять суббот, воскресений и понедельников. В таком случае число понедельников будет на два больше, чем четвергов и пятниц. Заметим, что ситуация, когда количество дней x на два больше, чем другого, может возникнуть только в том случае, если год начался в день x . Значит, в нашей задаче год начинается в четверг. Тогда, чтобы решить задачу, нужно посчитать количество дней, которое прошло с начала года к введенной дате P , ответом будет $(3 + P) \bmod 7 + 1$.