

Соревнование являлось отборочным к всероссийской командной олимпиаде школьников по программированию 2021 для учеников Челябинской области. Задачи были подготовлены членами жюри олимпиады, перечислены в алфавитном порядке:

Будников Михаил, СПбГУ
Ведерников Николай, ИТМО
Гардер Антон, VK
Григорьев Савелий, СПбГУ
Карпович Евгений, СПбГУ

Коробков Роман, ИТМО
Кучеренко Демид, МФТИ
Шиляева Екатерина, СПбГУ
Шпаковский Денис, МФТИ

Задача А. Лунтик и концерты

Пусть S — сумма длительностей всех песен, то есть $S = a + 2 \cdot b + 3 \cdot c$.

Заметим, что так как $a, b, c \geq 1$, то можно провести концерт любой длительности от 0 до S (действительно, если просто действовать по жадному алгоритму и брать трехминутные треки пока можем, затем двухминутные, затем одноминутные, то мы сможем получить любую длительность).

Тогда ответом на задачу будет являться остаток по модулю 2 числа S , так как если S четное, то можно получить первый концерт с длительностью $\frac{S}{2}$, у второго останется длительность $S - \frac{S}{2} = \frac{S}{2}$, и разность длительностей будет 0. Если же S нечетное, то в лучшем случае можно получить разность 1, составим первый концерт с длительностью $\lfloor \frac{S}{2} \rfloor$, а второй с оставшейся длительностью $\lceil \frac{S}{2} \rceil$.

Задача В. Составить число

Сначала определим, сколько цифр будет в искомом числе. Пусть самая дешевая цифра стоит d , тогда если эта цифра не ноль, то цифр будет $\lfloor \frac{x}{d} \rfloor$. Если ноль является самой дешевой цифрой, то найдем самую дешевую цифру из остальных, пусть она стоит p . Тогда цифр будет $\lfloor \frac{x-p}{d} \rfloor + 1$.

Теперь у нас имеется массив cnt , в котором cnt_i — количество цифр i в нашем числе. Исходно у нас получилось, что либо число целиком состоит из самой дешевой цифры, либо оно состоит из всех нулей и одной самой дешевой ненулевой цифры. У нас имеется еще некоторый бюджет x для того, чтобы сделать это число больше. Очевидно, что нам нужно покупать самые большие цифры взамен самых маленьких, чтобы итоговое число стало как можно больше. Переберем цифру, на которую будем заменять (по уменьшению) и цифру, которую будем заменять (по увеличению), будем заменять столько, на сколько хватает денег и менять имеющийся баланс.

Теперь мы знаем из каких цифр состоит число, нужно вывести ответ. Если цифр в числе меньше 50, то просто выведем цифры в порядке убывания, выводя каждую нужное число раз. Если цифр в числе много, то выведем 50 наибольших цифр, и 50 наименьших цифр в обратном порядке.

Задача С. Баба Капа вяжет шарф

Давайте переберем ту букву, которую будем удалять из строки (от 'a' до 'z'), и для каждой буквы независимо найдем минимальное количество символов, которое нужно удалить из строки, чтобы она стала палиндромом.

Пусть мы сейчас рассматриваем букву c . Давайте воспользуемся методом двух указателей: будем поддерживать два указателя l, r , изначально l указывает на начало строки, а r на конец. Теперь будем формировать палиндром: каждый раз будем сравнивать s_l и s_r , если они равны, то можно добавить их в палиндром на соответствующие позиции и перейти к символам $l + 1$ и $r - 1$. Если же $s_l \neq s_r$, то какой-то из символов нужно удалять (иначе не получится палиндром), если $s_l = c$, удалим его (добавим к количеству удаленных символов 1 и перейдем к $l + 1$ -му символу), аналогично, если $s_r = c$, то добавим к количеству удаленных символов 1 и перейдем к $r - 1$ -му символу. Если же $s_l \neq c$ и $s_r \neq c$, то получить из s палиндром удалением только букв, равных c , невозможно.

Асимптотика решения $O(|C| \cdot n)$, где $|C|$ — размер алфавита, т.е. 26.

Задача D. Игра на цикле

Давайте сначала узнаем длины минимальных путей между каждыми соседними вершинами (1 и 2, 2 и 3, ..., n и 1), потратим на это n запросов. Пусть для каких-то вершин длина минимального пути

не равна длине ребра между вершинами, тогда $w_i > S - w_i$, где w_i вес ребра между вершинами, S — сумма весов всех ребер на цикле, и $S - w_i$ — длина найденного минимального пути между вершинами. Можно преобразовать это неравенство в $w_i > \frac{S}{2}$, из него очевидно следует, что таких ребер в цикле может быть не более одного, назовем такое ребро (если оно есть) длинным, у него будет некоторый индекс i .

Давайте теперь узнаем, как найти вес длинного ребра. Так как длинных ребер не более одного, все остальные веса мы нашли корректно, сумма весов всех ребер, кроме длинного, является длиной минимального пути для пары вершин длинного ребра, то есть $m_i = \sum_{j=1, j \neq i}^n m_j$, где m — длины кратчайших путей для всех соседних вершин. Таким образом, теперь необходимо найти тот m_i , который будет равен сумме всех остальных, если в графе есть длинное ребро, то оно будет i -ым (опять

же, такое ребро опять будет одно, так как можно преобразовать равенство в $m_i = \frac{\sum_{j=1}^n m_j}{2}$, отсюда, очевидно, следует, что таких m_i будет не более одного (так как $n > 2$). Теперь для найденного "подозрительного" ребра спросим длину максимального пути между его вершинами, если найденное ребро длинное, то полученная длина будет больше минимального пути между вершинами, и эта длина и будет весом ребра, а иначе длина длиннейшего пути совпадет с длиной кратчайшего пути (и вес этого ребра будет равен длине кратчайшего пути, найденной изначально).

Таким образом, мы потратили всего $n + 1$ запрос и узнали веса всех ребер.

Задача Е. Лунтик и подпоследовательности

Можно заметить, что все подпоследовательности с суммой $s - 1$ образуются, если убрать из исходного массива какие-то нули и ровно одну единицу. Можно независимо посчитать количество способов убрать из массива нули (таким образом, сумма не поменяется), а затем посчитать количество способов убрать из массива ровно одну единицу (таким образом, сумма станет ровно $s - 1$), а затем перемножить эти два числа. Получается, ответ будет равен $2^{c_0} \cdot c_1$, где c_0 — количество 0 в массиве, а c_1 — количество 1.

Задача F. Кузя и конфеты

Научимся для x конфет определять, сколькими способами их можно разложить на кучки. Пусть мы разложили их на кучки $L, L + 1, \dots, R - 1, R$. Тогда выполняется $\frac{(L+R)(R-L+1)}{2} = x \Rightarrow (L+R)(R-L+1) = 2x$. Обозначим $A = L + R$, $B = R - L + 1$, тогда $L = \frac{A-B+1}{2}$, $R = \frac{A+B-1}{2}$. Следовательно, A и B должны быть разной чётности и $A > B$. Так как $2x$ — чётное, то один из множителей должен быть чётным, а второй — нечётным.

Пусть $x = 2^k \cdot t$, t — нечётное, $k \geq 0$. Заметим, что можно построить биекцию между разложениями t на множители (u, v) и парами (A, B) : пусть $u \cdot v = t$, тогда $A = \max(2^{k+1}u, v)$, $B = \min(2^{k+1}u, v)$. Наоборот: если A — чётное, то $u = \frac{A}{2^{k+1}}$, $v = B$, иначе $u = \frac{B}{2^{k+1}}$, $v = A$. Таким образом, количество способов разложить конфеты на кучки — это количество способов разложить t на два множителя.

Вернёмся к исходной задаче. Для каждого элемента массива вычислим число разбиений и найдём максимум. Итоговое время работы $\mathcal{O}(n\sqrt{C})$, где $C = \max a_i$.

Задача G. Корней Корнеевич и XOR

Давайте рассматривать все числа массива и для каждого числа t поддерживать список g_t всех таких чисел y , что можно выбрать на текущем префиксе возрастающую подпоследовательность, в которой xor чисел равен y , и последний элемент этой возрастающей подпоследовательности меньше t .

Пусть мы рассматриваем элемент a_i . Тогда давайте рассмотрим элементы g_{a_i} — здесь будут все значения xor -ов подпоследовательностей, к которым мы сможем добавить элемент a_i . Если в g_{a_i} будет значение f , то мы сможем получить значение $f \oplus a_i$, тогда добавим $f \oplus a_i$ ко всем спискам g от $a_i + 1$ до максимального значения массива a (если добавляемый элемент уже там был, добавлять его не нужно).

Давайте сделаем несколько оптимизаций этого решения.

1. Давайте не будем рассматривать уже рассмотренные значения g_t . То есть, если мы на какой-то итерации рассмотрели g_t , то очистим его, но запомним, что значения xor , которые мы удаляем, добавлять никогда больше не нужно. Этой оптимизации достаточно, чтобы получить время работы $O((\max_a)^3)$, где $\max_a$ — максимальное среди чисел массива a (для каждого числа t и его возможного значения $xor\ f$ мы передадим значение $t \oplus f$ во все состояния $t+1, \dots, \max_a$; всего различных t будет $O(\max_a)$, количество f тоже $O(\max_a)$, и передавать значение каждый раз нужно будет за $O(\max_a)$).
2. Заметим, что когда мы передаем какое-то значение xor , равное f в элементы $t+1, \dots, \max_a$, и оказались в элементе r , в котором это значение xor уже есть, то это значение f есть и во всех элементах, больших r , и поэтому дальше можно не добавлять значение f . С помощью этой оптимизации время работы станет $O(\max_a^2)$, так как для каждого значения xor (а их всего будет $O(\max_a)$), мы сделаем суммарно $O(\max_a)$ операций.

Таким образом, с учетом всех оптимизаций асимптотика решения составит $O(n + (\max_a)^2)$.

Задача Н. Пчеленок и подотрезки

Заметим, что k может являться ответом, только если сумма длин отрезков не превосходит количество элементов массива, то есть $\frac{k \cdot (k+1)}{2} \leq n$. Из этого неравенства можно получить, что максимальное возможное k меньше $\sqrt{2n}$, и при максимальном значении n оно не превосходит 446.

Пусть $dp_{i,j}$ — максимальная сумма на отрезке длины j , при этом мы рассмотрели все элементы на суффиксе i (то есть элементы с индексами $i, i+1, \dots, n$) и уже выбрали отрезки длины $j, j-1, \dots, 1$, в которых суммы возрастают. Давайте научимся пересчитывать значения динамики. Мы можем не взять i -й элемент в отрезок длины j , тогда нужно обратиться к значению динамики $dp_{i+1,j}$, а можем взять i -й элемент, и тогда нас интересует значение динамики $dp_{i+j,j-1}$ — если оно больше, чем сумма на отрезке $i, i+1, \dots, i+j-1$, то на суффиксе i можно выбрать j отрезков с длинами от j до 1, а иначе нельзя. Среди этих двух вариантов нам нужно взять максимум, чтобы максимизировать сумму. Для вычисления суммы чисел на отрезке можно воспользоваться префиксными суммами.

Асимптотика решения — $O(n \cdot \sqrt{n})$.

Задача I. День рождения Милы

Нам доступна операция замены a_i на $\sum_{j=1}^n a_j$. Заметим, что после применения такой операции, a_i становится максимумом в массиве, т.к. все числа положительные. Таким образом, последняя операция в ответе всегда происходит с максимальным числом.

Будем решать задачу с конца с помощью повторения следующих действий:

1. Найдём максимальное число a_k . Если $a_k = 1$, то мы конструктивно построили ответ.
2. Отнимем из a_k сумму всех остальных чисел t . Если $a_k \leq 0$, то построить ответ нельзя.

С каждой операцией максимальное число увеличивается хотя бы на $n-1$, поэтому максимальное количество шагов алгоритма не больше $\frac{10^9}{n-1}$, что в худшем случае достигает 10^9 . Поэтому будем вычитать из максимума a_k значение $\text{cnt} \times t$, где cnt выберем как минимальное количество шагов, чтобы максимум стал меньше какого-то другого числа.

Можно показать, что при таком подходе количество шагов алгоритма будет не больше $\log_{1.6}(10^9) \approx 45$.

Задача J. Диагонали

Сначала научимся находить количество клеток в нижней области. Пусть красная клетка имеет координаты (r, c) . Проведём диагонали влево-вниз и вправо-вниз. Найдём, в каких клетках они пересекают нижнюю строчку (пока считаем, что доска не ограничена слева и справа). Заметим, что левая синяя клетка будет иметь координаты $(1, c-r+1)$, а правая — $(1, c+r-1)$. Обозначим $L = c-r+1$, $R = c+r-1$.

Рассмотрим пример для $r = 3$, $c = 5$.

3					
2					
1					
0					
	3	4	5	6	7

Зелёным отмечены нужные нам клетки из нижней области.

Заметим, что для подсчёта всех зелёных клеток нужно просуммировать нечётные числа от 1 до $(2r - 3)$: $1 + 3 + \dots + (2r - 3) = (r - 1)^2$.

Теперь вспомним, что у нас есть левая и правая граница доски. Если отрезок $[L, R]$ выходит за границы отрезка $[1, m]$, то нужно вычесть излишки.

Рассмотрим пример для $r = 5$, $c = 2$, $m = 3$.

5									
4									
3									
2									
1									
0									
	-2	-1	0	1	2	3	4	5	

Чёрным отмечены клетки из нижней области, которые вылезают за пределы доски. Тогда излишек слева составляет $1 + 2 + \dots + (-L) = \frac{(-L)(-L+1)}{2}$, а излишек справа — $1 + 2 + 3 + \dots + (R - m - 1) = \frac{(R-m-1)(R-m)}{2}$.

Таким образом, считаем количество зелёных и чёрных клеток в совокупности (их $(r - 1)^2$), затем при необходимости вычитаем чёрные клетки слева и справа.

Чтобы найти остальные области достаточно три раза повернуть доску на 90° . При повороте против часовой стрелки клетка (r, c) переходит в клетку $(c, n - r + 1)$. Таким образом, чтобы вычислить площадь области слева, нужно положить $r' = c$; $c' = n - r + 1$; $n' = m$, $m' = n$, а дальше (после поворота доски) запустить ту же самую функцию подсчета количества клеток в нижней области.

Задача К. Обход дерева

$a_1 = b_n$ — номер корня дерева. Пусть первый ребёнок корня является корнем поддерева размера x . Тогда обходы этого дерева — последовательности $a_2, a_3, \dots, a_{x+1}$ и $b_1, b_2, \dots, b_x$, а обходы оставшейся части дерева — последовательности $a_1, a_{x+2}, a_{x+3}, \dots, a_n$ и $b_{x+1}, b_{x+2}, \dots, b_n$, и мы можем для обеих пар обходов решить задачу рекурсивно, после чего подвесить получившееся поддерево ребёнка к корню дерева. Чтобы вычислить x , нужно найти такое j , что $b_j = a_2$, тогда $x = j$. Это можно сделать линейным поиском по последовательности b , и тогда итоговое время работы решения равно $\mathcal{O}(n^2)$.

Задача Л. Лунтик, Луна и игра

Среди любых 5 точек можно найти две с одинаковой чётностью x и y (по принципу Дирихле) и середина отрезка между ними будет ответом.

Если это не удалось, точек не более 4 и можно проверить каждую пару. Если $\gcd(x_1 - x_2, y_1 - y_2) = 1$, на отрезке между ними нет целочисленных точек. Иначе подойдёт точка $(\frac{x_1 + (x_2 - x_1)}{\gcd(x_1 - x_2, y_1 - y_2)}, \frac{y_1 + (y_2 - y_1)}{\gcd(x_1 - x_2, y_1 - y_2)})$.

$\gcd(a, b)$ — наибольший общий делитель чисел a и b .

Задача М. Лунтик и квадраты

Для чисел от 0 до 3 ответы равны соответственно «», «+», «— — +», «— +», их можно построить конструктивно или найти перебором всех комбинаций маленьких гирек.

А ещё, заметим, что приписывание «+ — — +» прибавляет к ответу 4 (т.е. $x^2 - (x+1)^2 - (x+2)^2 + (x+3)^2 = 4$). Теперь для любого числа ответ будет строиться как одна из четырёх базовых строк и повторённая нужное число раз строка «+ — — +» для любых свободных x .

Задача N. Вупсень, Пупсень и 0

Давайте рассмотрим два случая: когда n четное, и когда n нечетное.

Если n четное, давайте разобьем все числа на пары: пусть a_1 и a_2 находятся в одной паре, a_3 и a_4 тоже в одной паре и так далее. В паре a_i, a_{i+1} , пусть $b_i = a_{i+1}$ и $b_{i+1} = -a_i$, тогда сумма $a_i \cdot b_i + a_{i+1} \cdot b_{i+1}$ в каждой паре будет равна 0 ($a_i \cdot a_{i+1} + a_{i+1} \cdot (-a_i) = 0$), поэтому общая сумма $a_1 \cdot b_1 + a_2 \cdot b_2 + \dots + a_n \cdot b_n$ также будет равна 0. Заметьте, что сумма $|b_i|$ равна сумме $|a_i|$, следовательно, сумма $|b_1| + |b_2| + \dots + |b_n|$ не превосходит $\text{MAXA} \cdot \text{MAXN} = 10^9$.

Если n нечетное, то откинем последние 3 числа и вычислим b для массива $a_1, a_2, \dots, a_{n-3}$ таким же образом, как и для четного n ($n-3$ четное, так как n нечетное). Тогда для последних 3 чисел a_{n-2}, a_{n-1}, a_n найдем два числа с суммой, не равной 0 (по принципу Дирихле, всегда будут существовать два числа среди трех, не равных 0, которые либо оба положительные, либо оба отрицательные, и сумма этих двух чисел никак не может быть равна 0), пусть это будут числа a_i, a_j , и третье число будет a_k . Тогда пусть $b_i = -a_k, b_j = -a_k, b_k = (a_i + a_j)$, сумма $a_i \cdot b_i + a_j \cdot b_j + a_k \cdot b_k = 0$, и числа в b не равны 0. Сумма $|b_1| + |b_2| + \dots + |b_{n-3}|$ не превышает $\text{MAXA} \cdot (\text{MAXN} - 1 - 3)$ (так как MAXN четно, а мы рассматриваем случай с нечетным n) и сумма $b_{n-2} + b_{n-1} + b_n$ не превышает $\text{MAXA} + \text{MAXA} + 2 \cdot \text{MAXA} = 4 \cdot \text{MAXA}$, поэтому сумма $|b_i|$ не превышает $\text{MAXA} \cdot \text{MAXN} = 10^9$.

Задача О. Кузя и домашнее задание

Заметим, что отрезок будет простым, если рассмотрев каждый его делитель, мы получим скобочную последовательность, в который минимальный баланс больше либо равен 0 (открывающаяся скобка — если мы умножаем на этот делитель и закрывающая — если делим).

Давайте будем рассматривать элементы массива a и считать массив next_i — максимальная левая граница, такая что если мы хотим сделать i -ю операцию, то мы сможем ее сделать, если $l \leq \text{next}_i$. Для этого для каждого делителя будем поддерживать все индексы его вхождений (если делитель несколько раз встречается в числе, то можно хранить индекс несколько раз). Если у нас i -я операция умножения, то next_i равно i и все делители числа a_i нужно будет добавить, а если операция деления, то нужно будет удалить соответствующее количество делителей для каждого делителя и запомнить минимальное удаленное значение в массив next_i .

Теперь зная этот массив нужно посчитать количество отрезков $1 \leq l \leq r \leq n$, таких что $l \leq \min(l, r)$, где $\min(l, r)$ — это минимальное значение next_i на отрезке l, r .

Это можно сделать с помощью дерева отрезков на минимум за $O(n \cdot \log_2 n)$ (переберем левую границу и с помощью спуска по дереву найдем максимальную правую границу) или с помощью линейных алгоритмов со стеком (для этого будем перебирать все левые границы в порядке убывания и поддерживать стек на минимум).