Задача A. Натуральное деление

Автор задачи: Владимир Сухов, разработчик: Владимир Смаглий

Будем искать взаимно простые a и b.

Обозначим число, образуемое последовательностью чисел после запятой, через c. Тогда верно равенство:

$$\frac{c}{10^n} = \frac{a}{b}$$

Значит, чтобы найти a и b, необходимо найти все общие делители у 10^n и c, а затем поделить на них знаменатель и делитель. Заметим, что общими делителями могут быть только числа 2 и 5.

Это можно сделать следующим кодом:

```
while(c % 5 == 0 && ten_pow_n % 5 == 0) {
    c /= 5;
    ten_pow_n /= 5;
 }

while(c % 2 == 0 && ten_pow_n % 2 == 0) {
    c /= 2;
    ten_pow_n /= 2;
}
```

Осталось проверить, что получившиеся два числа меньше 10^6.

Альтернативно можно найти наибольший общий делитель чисел c и 10^n и сократить дробь на него для получения чисел a и b.

Задача B. Площадь треугольника

Автор и разработчик задачи: Николай Ведерников

Площадь треугольника равна $\frac{c \cdot h}{2}$. Надо понять, когда не существует такого треугольника.

Пусть такой треугольник существует. Построим описанную окружность вокруг треугольника. Так как угол прямой, то гипотенуза будет диаметром. Тогда, чтобы такой треугольник существовал, нужно, чтобы высота была не больше, чем радиус.

Тогда, если $2 \cdot h > c$, то ответ -1, иначе $\frac{c \cdot h}{2}$.

Задача C. Число Чебурашки

Автор и разработчик задачи: Екатерина Ведерникова

Заметим, что на шаге i Чебурашка берет i апельсинов, а Гена берет один апельсин. Итого, общее количество апельсинов, которое они берут на шаге i, равно $i + 1$.

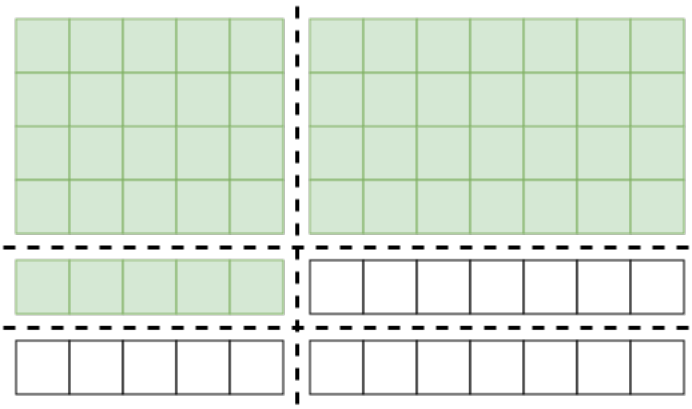
Общее количество апельсинов, взятых за k шагов, будет равно:

$n = (1 + 1) + (1 + 2) + (1 + 3) + \ldots + (1 + k) = \sum_{i=1}^{k} i + k = \frac{k \cdot (k+1)}{2} + k = \frac{k \cdot (k+3)}{2}$

Задача D. Ленивые разрезы

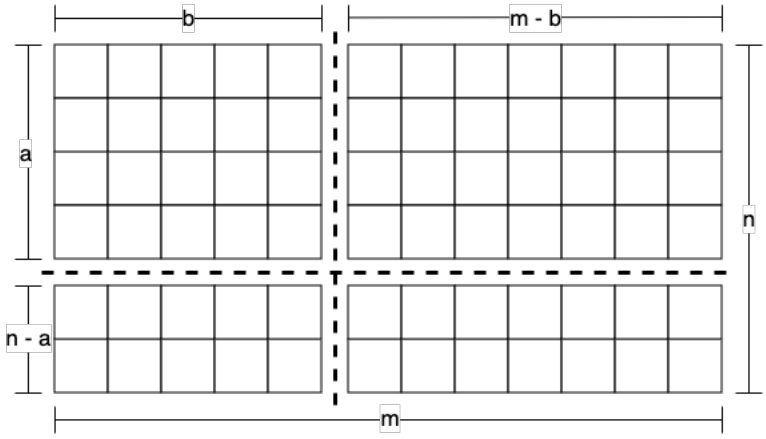
Автор задачи: Никита Голиков, разработчик: Григорий Шовкопляс

Заметим, что для получения площади s всегда нужно не более трех разрезов. Используя один разрез, мы можем отрезать первые k строк, а затем, используя еще два разреза получить из следующей строки любое количество клеток.



Разберем случаи, когда хватит меньше трех разрезов, а именно ноль, один или два. Ноль разрезов может получиться тогда и только тогда, когда: $s = n \cdot m$. Одного разреза хватит, если s кратно n или s кратно m.

При двух разрезах имеет смысл делать один разрез вертикально, а второй горизонтально. Пусть эти разрезы делят клетчатый лист после a первых рядов и b первых столбцов ($0 < a < n$ и $0 < b < m$), как показано на рисунке ниже.



Дальше может быть три способа получить s клеточек:

1. Возьмем левый верхний кусочек, если $s = a \cdot b$
2. Возьмем все кроме левого верхнего кусочка, если $s = n \cdot m - a \cdot b$
3. Возьмем левый верхний и правый нижний кусочки, если $s = a \cdot b + (n - a) \cdot (m - b)$.

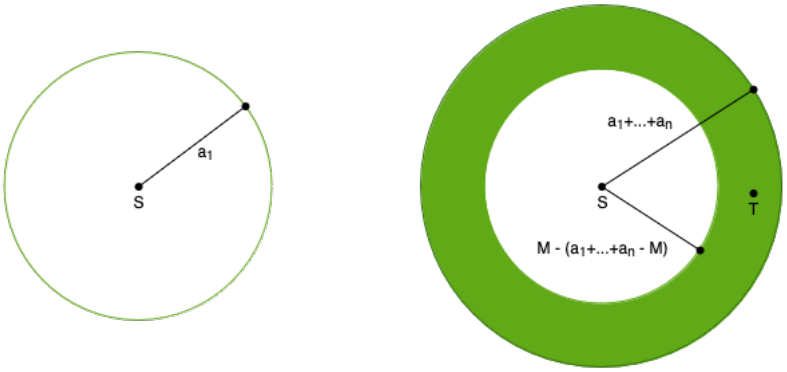
Все способы получения можно проверить за $O(n)$ перебрав первый разрез и проверив за $O(1)$ существование подходящего второго разреза, решив уравнение с одной неизвестной.

Задача E. Декартов Кузнечик

Автор задачи: Демид Кучеренко, разработчик: Григорий Шовкопляс

Назовем стартовую точку точкой S, а конечную T.

Рассмотрим геометрическое место точек (ГМТ) положений Кузнечика. После первого прыжка ГМТ будет окружностью с центром в стартовой точке и радиусом равным первому прыжку, после второго прыжка и далее ГМТ становится кольцом, возможно с нулевым внутренним радиусом.



Таким образом, кузнечик не сможет допрыгать до точки T, если она, либо будет за внешним радиусом кольца, полученного после всех прыжков, либо внутри внутреннего. Обозначим за M максимальный по длине прыжок из набора, за D расстояние между S и T. T достижима из S тогда и только тогда, когда $M - (a_1 + a_2 + \cdots + a_n - M) \leqslant D \leqslant a_1 + a_2 + \cdots + a_n$.

Далее необходимо определить траекторию прыжков. Сделаем несколько допущений для удобства построения решения.

- Пусть траектория начинается и заканчивается в S, а к набору прыжков добавим в конец прыжок длины D.
- $M' = \max(M, D)$
- Прыжок длины M' уберем из набора и будем считать, что его можно выполнить в любой момент.

Траекторией прыжков в таком случае будет являться треугольник со сторонами $a_1+a_2+\cdots+a_k$, $a_{k+1}\cdots+a_n$ и M'. В качестве k возьмем минимальное k такое, что $a_1+a_2+\cdots+a_k+M' \geqslant a_{k+1}\cdots+a_n$.

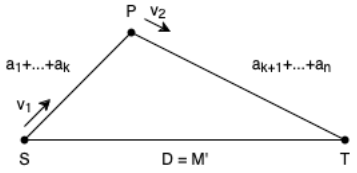
Докажем существование такого треугольника (проверим выполнение неравенства треугольника). Заметим, что по проверенному ранее условию гарантируется, что $M' \leqslant a_1 + a_2 + \cdots + a_n$. Также мы специально выбрали k так, чтобы $a_1 + a_2 + \cdots + a_k + M' \geqslant a_{k+1} \cdots + a_n$, следовательно остается доказать, что $a_1 + a_2 + \cdots + a_k \leqslant a_{k+1} \cdots + a_n + M'$.

Пусть это не так, тогда должно быть верно, что $a_1 + a_2 + \cdots + a_{k-1} + a_k > a_{k+1} \cdots + a_n + M'$. k мы выбирали таким образом, что верно $a_1 + a_2 + \cdots + a_{k-1} + M' < a_k + a_{k+1} \cdots + a_n$. Вычтем первое неравенство из второго, сократив общие члены, получим $a_k - M' > M' - a_k$, что сводится к $a_k > M'$, что есть противоречие. Таким образом, существование треугольника со сторонами $a_1 + a_2 + \cdots + a_k$, $a_{k+1} \cdots + a_n$ и M' доказано.

Теперь, для финального построения траектории, рассмотрим два случая: $M' = D$ и $M' \neq D$.

$M' = D$

В качестве основания треугольника выберем отрезок из S в T. Далее найдем третью точку треугольника со сторонами $a_1 + a_2 + \cdots + a_k$, $a_{k+1} \cdots + a_n$ и $M' = D$. Это можно сделать по формуле координат точки пересечения двух окружностей. Из двух вариантов точек выберем любой. Пусть это будет точка P



Затем вычислим единичные векторы $\vec{v_1} = \vec{SP}/|SP|$ и $\vec{v_2} = \vec{PT}/|PT|$. Тогда, чтобы допрыгать из точки S в точку P, будем вычислять каждую следующую точку $P_{next} = P_{prev} + a_i \cdot \vec{v_1}$, а затем, чтобы допрыгать из точки P в точку T — по формуле $P_{next} = P_{prev} + a_i \cdot \vec{v_2}$.

$M' \neq D$

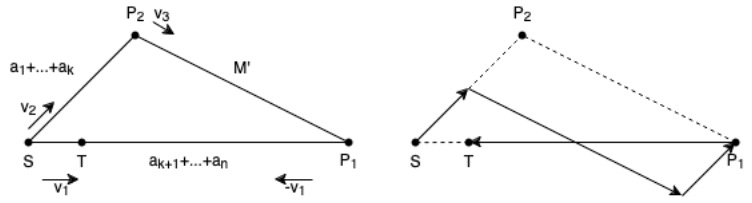
Второй случай немного сложнее. В нем T не является вершиной треугольника, но при этом должна лежать на одной из его сторон, чтобы все получилось. Так как прыжок длины D мы дописывали в конец, точка T должна лежать на стороне треугольника длиной $a_{k+1} \cdots + a_n$.

Также вспомним допущение, что прыжок длины M' мы убирали из набора и считали, что его можно выполнить в любой момент. Так как на самом деле порядок строго определен, при вычислении $a_1 + a_2 + \cdots + a_k$ не будем учитывать M', если первое вхождение максимума достигается в индексе $i \leqslant k$, тогда у нас не будет путаницы с индексами в дальнейшем решении.

Пусть $\vec{v_1} = \vec{ST}/|ST|$ Вычислим вершину треугольника P_1, как $P_1 = S + (a_{k+1} \cdots + a_n) \cdot \vec{v_1}$. Затем вычислим P_2, как третью точку треугольника, аналогично предыдущему случаю. Далее необходимо вычислить два единичных вектора $\vec{v_2}$ и $\vec{v_3}$, сонаправленных с $\vec{SP_2}$ и с $\vec{P_2P_1}$ соответственно.

Затем решение выглядит аналогично предыдущему случаю, мы вычисляем следующую точку из предыдущей с помощью прибавления вектора:

- $a_i \cdot \vec{v_3}$, если $a_i = M'$ и это первое вхождение максимума.
- $a_i \cdot \vec{v_2}$, если $i \leqslant k$
- $a_i \cdot -\vec{v_1}$, если $i > k$



Также необходимо аккуратно рассмотреть случай $S = T$, в нем можно выбрать в качестве $\vec{v_1}$ любой единичный вектор.

Задача F. Робот-доставщик

Автор задачи: Артем Васильев, разработчик: Маргарита Саблина

Требуется покрыть точки в квадрате $n \times n$ с помощью $2n - 2$ отрезков. Способ построения при $n = 3$ приведен в примере. Будем строить ответ для квадрата $(n + 1) \times (n + 1)$ с помощью ответа для квадрата $n \times n$ путем добавления двух дополнительных отрезков.

Для перехода от $n = 3$ к $n = 4$ возьмем последовательность отрезков из условия и продлим последний отрезок до точки $(0; -1)$. После этого будем "заворачивать" квадрат $n \times n$ двумя дополнительными отрезками вдоль границы предыдущего квадрата, покрывая все недостающие точки.

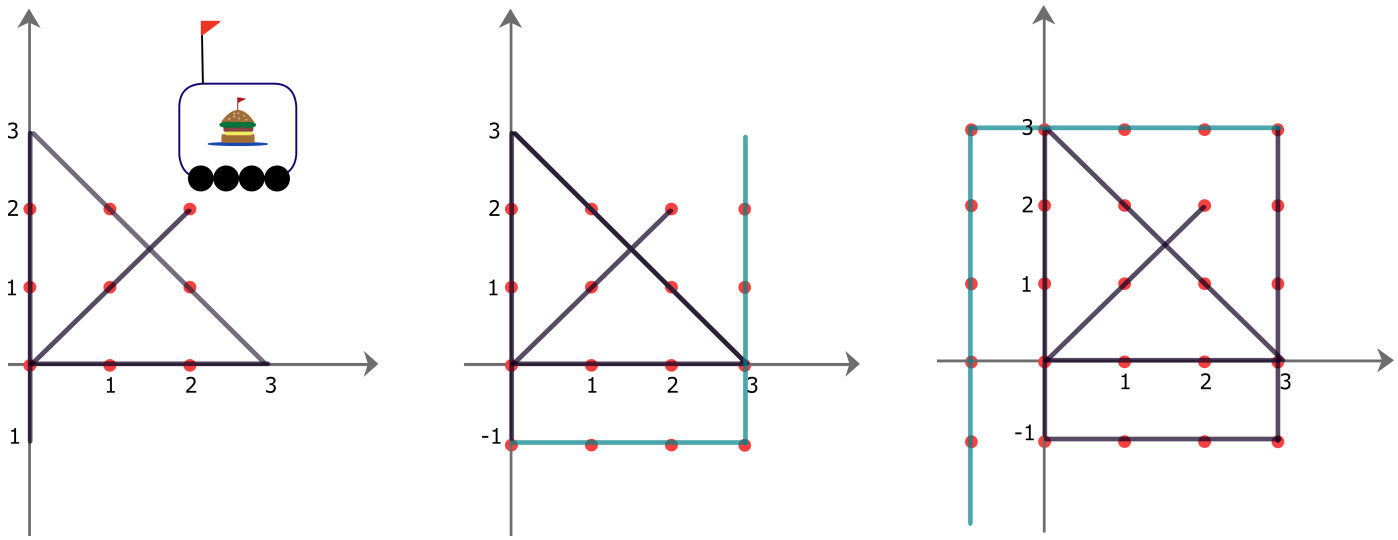
Например, для случая $n = 4$ получится последовательность точек

$$(2; 2), (0; 0), (3; 0), (0; 3), (0; -1), (4; -1), (4, 4).$$

При этом мы сразу продлили последний отрезок так, чтобы можно было продолжить построение для квадрата размером на один больше.

Осталось только сдвинуть начало координат так, чтобы левый нижний угол итогового квадрата $n \times n$ оказался в точке $(0; 0)$.

На рисунке показано, как перейти от спирали для $n = 3$ к $n = 4$ и $n = 5$.



Задача G. Поездка в Москву

Автор задачи: Андрей Станкевич, разработчик: Михаил Первеев

Заметим, что в оптимальном решении Андрей будет двигаться по числовой прямой только направо. Помимо точек 0 и m он, возможно, остановится на некоторых заправочных станциях, а на каждом промежутке между двумя последовательными остановками он будет двигаться с максимально возможной постоянной скоростью таким образом, чтобы ему хватило топлива.

Рассмотрим две точки с координатами x_1 и x_2 ($x_1 < x_2$). Вычислим максимальную скорость, с которой может двигаться Андрей, чтобы доехать из точки x_1 в точку x_2 при условии, что в точке x_1 его автомобиль заправлен до полного бака. Так как при скорости v автомобиль тратит v литров топлива на одну единицу расстояния, а размер бака равен c литрам, максимально возможная скорость равна $v_{\max} = \frac{c}{x_2 - x_1}$. При движении с такой скоростью в момент достижения точки x_2 в автомобиле не останется топлива. Таким образом, минимальное время в пути из точки x_1 в точку x_2 составит $\frac{x_2 - x_1}{v_{\max}} = \frac{(x_2 - x_1)^2}{c}$.

Теперь воспользуемся методом динамического программирования для решения задачи. Введем дополнительные точки с координатами $x_0 = 0$ и $x_{n+1} = m$. Скажем, что $dp[i]$ равно минимальному времени, достаточному, чтобы добраться из точки x_0 в точку x_i, а после этого заправить автомобиль в точке x_i.

Изначально мы находимся в точке x_0, поэтому $dp[0] = 0$. Для того, чтобы вычислить $dp[i]$, необходимо перебрать предыдущую точку, в которой будет выполнена остановка. Формула пересчета динамики выглядит следующим образом:

$$dp[i] = \min_{j=0}^{i-1} \left(dp[j] + \frac{(x_i - x_j)^2}{c} \right) + t_i$$

Ответ находится в $dp[n+1]$ (для удобства скажем, что $t_{n+1} = 0$, так как заправлять автомобиль в Москве не обязательно). Таким образом, мы получили решение за $\mathcal{O}(n^2)$.

Для начала немного изменим формулы для того, чтобы избавиться от промежуточных вычислений в вещественных числах, которые могут ухудшить точность. Для этого домножим полученное уравнение на $dp[i]$ на число c. После этого действия получим формулу:

$$c \cdot dp[i] = \min_{j=0}^{i-1} \left(c \cdot dp[j] + (x_i - x_j)^2 \right) + c \cdot t_i$$

Для удобства обозначим величину $c \cdot dp[i]$ как $dp'[i]$. Применив этот обозначение, мы получим формулу:

$$dp'[i] = \min_{j=0}^{i-1} \left(dp'[j] + (x_i - x_j)^2 \right) + c \cdot t_i$$

Теперь мы можем вычислить значение $dp'[n+1]$, пользуясь этой формулой, и в конце разделить ответ на число c. Это позволит избежать лишних вычислений в вещественных числах и возможной потери точности.

Наконец, оптимизируем решение по времени. Для этого раскроем квадрат в формуле пересчета динамики:

$$dp'[i] = \min_{j=0}^{i-1} \left(dp'[j] + (x_i)^2 - 2 \cdot x_i \cdot x_j + (x_j)^2\right) + c \cdot t_i$$

Теперь сгруппируем слагаемые:

$$dp'[i] = \min_{j=0}^{i-1} \left((dp'[j] + (x_j)^2) + (-2x_j) \cdot x_i\right) + c \cdot t_i + (x_i)^2$$

Заметим, что в данной формуле мы ищем минимум среди некоторого множества линейных функций вида $y = k_j \cdot x + b_j$, где $k_j = -2x_j$, а $b_j = dp'[j] + (x_j)^2$, в точке x_i. После того, как мы вычисляем значение $dp'[i]$, в множество добавляется очередная линейная функция.

Таким образом, нам необходимо выполнять операции двух типов:

- Добавить в рассматриваемое множество функцию $y = k \cdot x + b$ с некоторыми известными параметрами k и b;
- Найти минимум среди всех рассматриваемых функций в некоторой точке x.

Данные операции можно выполнять при помощи техники Convex Hull Trick. Добавить новую прямую можно за амортизированные $\mathcal{O}(1)$, так как угловой коэффициент добавляемой прямой не больше, чем угловые коэффициенты прямых, которые были добавлены ранее. Найти минимум в точке можно найти за $\mathcal{O}(\log n)$ при помощи двоичного поиска. Таким образом получится решение за $\mathcal{O}(n \log n)$.

Также можно заметить, что точки, в которых необходимо вычислять минимум, не убывают, поэтому можно избавиться от двоичного поиска, воспользовавшись методом двух указателей. Достаточно поддерживать указатель на прямую, которая является оптимальной в данной точке. Так как запросы монотонны, указатель будет двигаться только направо. Это позволит достичь времени работы $\mathcal{O}(n)$. Впрочем, для решения задачи это не требовалось.

Задача H. Разноцветный граф

Автор и разработчик задачи: Егор Юлин

Данную задачу можно решать несколькими способами. Рассмотрим решение, которое использует структуру данных *map*.

Для каждого цвета будем хранить количество *плохих* вершин. Плохими вершинами назовём те, из которых исходит более одного ребра цвета c. Количество таких вершин можно считать с помощью *map*, в котором для каждой вершины будем хранить количество соединенных с ней рёбер цвета c.

Тогда цвет является превосходным, если количество плохих вершин такого цвета равно 0. Теперь после каждого запроса за $O(1)$ можно сказать, является ли цвет c превосходным.

Обработка запросов выглядит следующим образом:

- Если цвет был превосходным, а после добавления ребра перестал таковым являться, то из ответа вычтем количество рёбер такого цвета;
- Если цвет не был превосходным, а после удаления ребра стал им, то к итоговому ответу добавим количество рёбер такого цвета;
- Если цвет не был превосходным и после запроса не стал им, то ответ никак не поменяется;
- Если цвет был превосходным и остался им, то к ответу добавится 1.

Количество рёбер каждого цвета можно хранить при помощи массива.

Время работы: $O((n+q)\log n)$.

Задача I. «Галактический Таймлайн»

Автор задачи: Маргарита Саблина, разработчик: Даниил Голов

Для начала поймем, что карточки, у которых на обеих сторонах написаны одинаковые числа, можно не рассматривать отдельно, потому что в правильном таймлайне они идут подряд друг за другом. Поэтому в решении будем считать, что не существует карточек с совпадающими числами на обеих сторонах.

Посмотрим на те карточки, на которых написано минимальное число из всех. Посмотрим на их оборотные стороны и выберем карточку с минимальным из всех на оборотной стороне, пусть она будет X. Заметим, что выбранная карточка может стоять в ответе только на первом месте.

Действительно, если предположить, что X стоит не на первом месте, то рассмотрим карточку, стоящую в таймлайне перед ней, пусть она будет Y. Известно, что число на одной стороне X — минимальное. Соответственно, на карточке Y на этой стороне может стоять только то же самое число, что и на X, иначе порядок не соблюдается. Но в таком случае число на другой стороне X будет меньше, чем число на другой стороне Y, и порядок тоже не соблюдается. Таким образом, карточка X может стоять только на первом месте.

Тогда выберем карточку X, поставим ее на первое место, исключим из рассмотрения, повторим процесс для оставшихся карточек. Карточку, которую выбираем не первой, если нужно, перевернем. Заметим, что таким образом мы выберем n карточек с минимальными числами, то есть просто посортируем карточки по минимальной стороне.

Тогда будем сортировать карточки сначала по минимальному числу на них, а при равенстве минимальных — по максимальному. После такой сортировки проверим, что на стороне с большими числами карточки тоже идут в правильном порядке, и если нет, скажем, что правильный порядок найти невозможно.

Задача J. Шахматный слон

Автор и разработчик задачи: Захар Яковлев

В этой задаче существует много различных решений. Мы приведём одно из конструктивных.

Для начала выполним следующую последовательность операций (по сути, двоичный поиск) и запомним, какие операции выполнились успешно:

1. $(4, 4)$
2. $(2, 2)$
3. $(1, 1)$

После выполнения этих операций слон гарантированно находится на последней 8-й горизонтали или на последней вертикали `h`. Так мы узнаем позицию на его диагонали (расстояния до правого или верхнего края доски). Теперь выполним следующую последовательность действий:

1. $(-4, -4)$
2. $(-2, -2)$
3. $(-1, -1)$

После её выполнения слон гарантированно стоит на 1-й горизонтали или на первой вертикали `a`. Теперь известна длина диагонали и позиция на ней.

Наконец, сделаем ход $(1, -1)$. Так мы определим, выше ли исходная диагональ, чем главная диагональ, или нет. Окончательно, известна диагональ, на которой находился слон, и позиция на ней.

Всего было выполнено ровно 7 операций.

Задача K. Магическое заклинание

Автор и разработчик задачи: Никита Голиков

Сделаем ориентированный граф на $n+1$ вершине, пронумеруем их числами от 1 до $n+1$, у каждого ребра будет цвет — число от 1 до k. Рассмотрим подотрезок $[l_q : r_q]$ массива a. Заметим, что нас интересуют несколько вариантов для использования его в ответе:

1. p входит в $a[l_q; r_q]$ как подотрезок — ответ равен 1
2. $a[l_q : r_q] = p[i : j]$ — ребро $i \to j+1$ цвета k
3. $a[l_q : l_q + x] = p[n - x : n]$ — ребро $n - x \to n + 1$ цвета k
4. $a[r_q - x : r_q] = p[1 : x + 1]$ — ребро $1 \to x + 2$ цвета k.

Для того чтобы определить, какие из случаев имеют место, предподсчитаем массивы left$[i]$ и right$[i]$, хранящие длину максимального подотрезка из i влево и вправо соответственно, входящего в p как подотрезок. Чтобы это сделать эффективно, предподсчитаем обратную перестановку для p, после чего массивы left и right насчитаем линейным проходом. Используя эти массивы и обратную перестановку, можно определить случаи 2-4.

Чтобы эффективно определить первый случай, найдем заранее все вхождения p в массив a, это можно сделать наивно циклом от каждого вхождения p_1 в массив a, суммарно это отработает за линейное время, так как p это перестановка. После этого, для каждого индекса i предподсчитаем ближайшее вхождение справа, и при рассмотрении отрезка $[l_q; r_q]$ найдем ближайшее вхождение справа от l_q, и проверим, что оно влезает в отрезок.

Теперь, если ответ не равен единице, задачу можно переформулировать так: нужно найти кратчайший по количеству ребер путь из вершины 1 в вершину $n+1$, все цвета ребер которого различны. Заметим, что все ребра нашего графа идут из меньшей вершины в большую, поэтому он ацикличен. Также заметим, что количество ребер каждого цвета не больше двух, и если их два, то это ребро из вершины 1, и ребро в вершину $n + 1$.

Без ограничения на цвета задача поиска кратчайшего пути в ациклическом графе решается динамическим программированием ($\mathrm{dp}[v]$ — длина кратчайшего пути из вершины v в вершину $n + 1$) за линейное время. Чтобы учесть, что бывают два ребра одинакового цвета, поступим следующим образом: от каждой вершины v насчитаем два кратчайших пути, у которых разные цвета последнего используемого ребра (иными словами, ребра в вершину $n + 1$). Чтобы это сделать, будем хранить массивы $\mathrm{dp}[v]$ и $\mathrm{dp}_2[v]$, а также массив edgeInd$[v]$, хранящие длину минимального кратчайшего пути, второго кратчайшего пути, отличающегося индексом последнего ребра, а также индекс используемого последнего ребра в кратчайшем пути соответственно. Это можно насчитать линейным проходом с конца, аналогично поиску второго минимума в массиве.

Теперь найдем ответ следующим образом: переберем первое ребро на пути, пусть это ребро $1 \to v$ цвета c. Если edgeInd$[v] \neq c$, то можно обновить ответ через $1 + \mathrm{dp}[v]$, иначе через $1 + \mathrm{dp}_2[v]$. Найдем минимум по всем первым ребрам, после чего восстановим ответ стандартным образом, сохранив массивы предков для $\mathrm{dp}[v]$ и $\mathrm{dp}_2[v]$. Получили решение за линейное время.

Задача L. Мультимино

Автор и разработчик задачи: Михаил Савватеев

Хочется найти какой-то паттерн, повторяя который с разными параметрами, мы будем получать много разных n-миношек, близких по структуре и хорошо соединяющихся друг с другом. Реализаций этой идеи наверняка существует множество, ниже приведена одна из довольно естественных.

Придумаем конструкцию для чётных достаточно больших n. Будем последовательно строить её из примера для $n = 6$ на рисунке слева, добавляя каждый раз во время перехода от n к $n + 2$ дополнительные n-миношки-“зигзаги” сверху разрезаемого квадрата и n-миношки-“лоханки” снизу:

Осталось чуть-чуть модифицировать конструкцию, чтобы она работала и для нечётных n, например, сделать это можно, как на рисунке справа:

Получается последовательность разрезаний, почти идентичная последовательности для чётных n, единственное важное замечание — они корректны только начиная с $n = 7$, так как при $n = 5$ чёрная и синяя n-миношки оказываются равны:

Для маленьких же n подберём ответ вручную. К счастью, самые сложные $n = 4, 5$ уже разобраны в условии; для $n = 2, 3$ тривиально доказывается, что нужного разбиения не существует; для $n = 1$ подойдёт одна 1-миношка, а для $n \geqslant 6$ наша конструкция сверху уже полностью валидна, так что будем применять её.

Задача M. Три точки

Автор задачи: Артем Васильев, разработчик: Екатерина Ведерникова

Будем рассматривать варианты расположения прямой (траншеи) и трёх точек на плоскости (три дома).

Пусть прямая не проходит ни через одну из точек. Тогда есть два случая:

1. Если прямая лежит вне треугольника, образованного этими точками. Воспользуемся параллельным переносом и перенесем прямую в ближайшую из вершин. Одно из расстояний сократится до 0, а остальные уменьшатся (так как это параллельный перенос).

2. Если прямая проходит внутри треугольника, то очевидно, что с одной стороны от неё находятся две точки, а с другой одна. Тогда параллельно перенесем прямую в ближайшую из вершин (из двух, которые лежат по одну сторону). Пусть мы перенесли прямую на расстояние h, тогда расстояние до двух точек уменьшилось на h каждое, а до одной точки увеличилось на h, суммарное расстояние уменьшилось на h. Таким образом, когда прямая проходит хотя бы через одну из точек, суммарное расстояние до неё будет меньше, чем если она не будет проходить ни через одну из них.

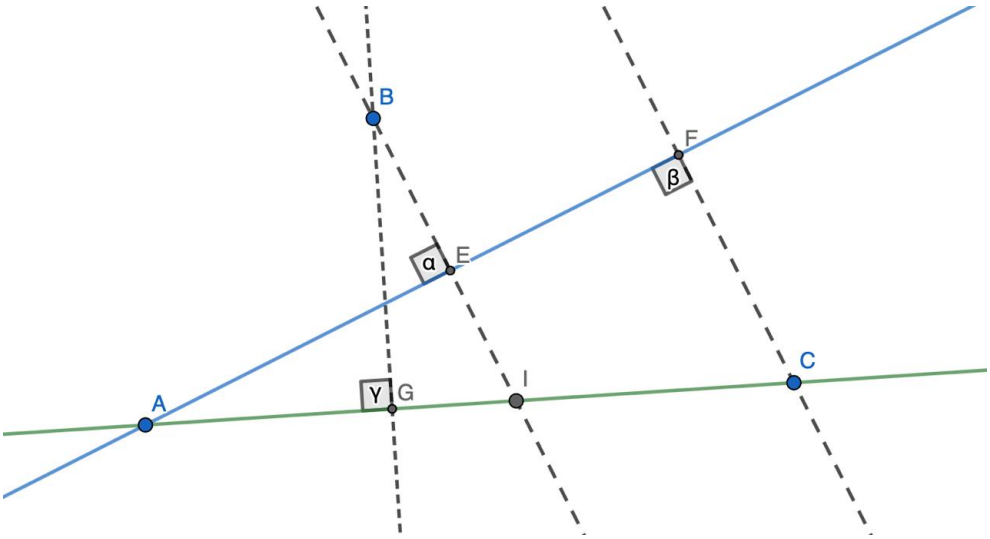
Следовательно, прямая проходит хотя бы через одну точку.

Пусть прямая проходит через две точки, тогда суммарное расстояние до неё будет меньшим в том случае, когда она проходит через две самые удалённые друг от друга точки. Если вспомнить формулу площади треугольника через высоту, то получим, что чем больше сторона, к которой проведена высота, тем меньше сама высота. Суммарное расстояние от всех точек будет равно длине высоты (так как расстояние от остальных точек будет нулевым). Осталось показать, что это оптимальное решение.

Сравним суммарное расстояние до прямой в случае, когда прямая проходит ровно через одну точку и когда прямая проходит через две самые удалённые друг от друга точки.

Рассмотрим два случая.

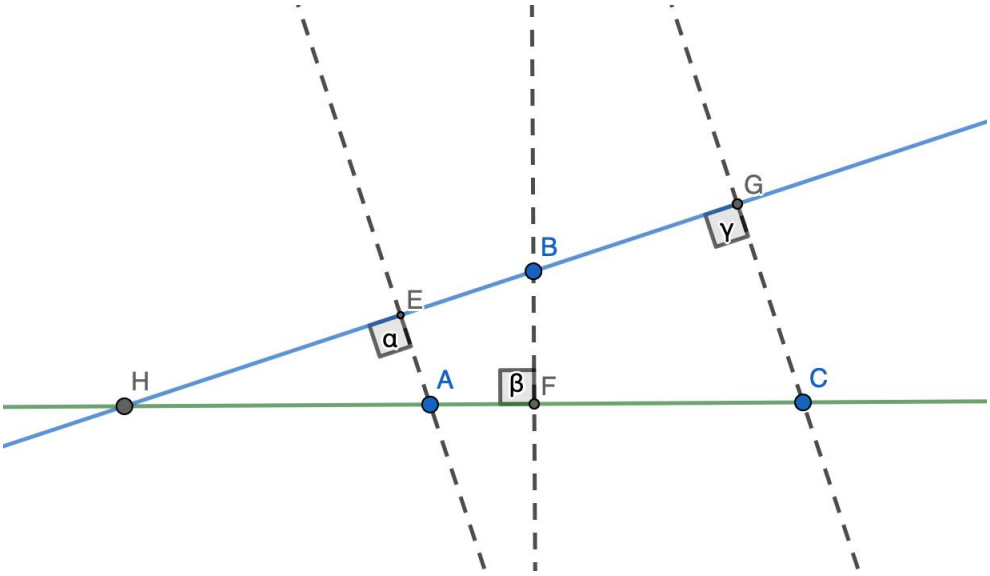
Первый случай:



Когда прямая проходит через одну из точек, которые лежат на прямой (проходящей через две самые удалённые точки). Надо сравнить высоту $\mathtt{BG}$ и сумму высот $\mathtt{BE}$ и $\mathtt{FC}$. Продлим $\mathtt{BG}$ до пересечения с $\mathtt{AC}$ в точке $\mathtt{I}$. Треугольник $\mathtt{BGI}$ – прямоугольный, гипотенуза $\mathtt{BI}$ ($\mathtt{BI} = \mathtt{BE} + \mathtt{EI}$) $>$ $\mathtt{BG}$ (катет). Также заметим, что $\mathtt{EI} < \mathtt{FC}$ (в силу параллельности этих прямых, легко заметить подобие треугольников, а

также то, что эта сторона будет меньше, так как и остальные стороны тоже меньше соответствующих сторон). Таким образом, мы получаем, что расстояние до прямой, проходящей через 2 наиболее удалённые точки, меньше суммы расстояний до прямой, проходящей через одну из этих точек, но не проходящей через вторую.

Второй случай:



Когда прямая проходит через третью точку (которая не лежит на прямой, проходящей через две самые удалённые друг от друга точки). Построили точку $\texttt{H}$ – пересечение этих прямых (в случае если они параллельны, то расстояния будут равны, и очевидно, что $h < h + h$). Угол при этой вершине общий, значит треугольники $\texttt{EHA}$, $\texttt{FHB}$ и $\texttt{GHC}$ подобны между собой. А значит высота $\texttt{BF} < \texttt{EA} + \texttt{GC}$ (помним, что $\texttt{HB} < \texttt{HC}$). И в этом случае оказалось, что если провести прямую через две самые удалённые друг от друга точки, то расстояние получится меньше, чем если провести прямую через третью точку.