Задача 1. Пермь

В задаче были даны две целочисленные точки на плоскости, нужно было найти третью целочисленную точку в заданном прямоугольнике так, чтобы треугольник, образованный точками, был тупоугольный.

Есть два основных способа решения этой задачи — найти третью вершину треугольника явно и перебрать некоторые точки. Начнем со второго способа, поскольку он проще, и на соревновании по программированию не требуется доказывать его корректность, быстрее написать и отправить. Переберем точки, в квадрате размера 3×3 (точек) вокруг данных, и проверим каждую из них, то есть проверим по 8 соседних точек для каждой из данных.

Докажем, что среди них найдется ответ. Начнем с ситуации, когда весь квадрат 3×3 вокруг одной из точек не выходит за границы прямоугольника. Назовем данные точки буквами A и B. Пусть точка C находится слева от A, а точка D симметрично справа от A. Один из углов $\angle CAB$ и $\angle DAB$ — тупой, либо оба угла прямые. Если оба угла прямые, то отрезок AB — вертикальный. Пусть A его нижний конец. Тогда возьмем точку ниже C — она будет являться ответом. Если A это верхний конец, то наоборот, возьмем точку выше C.

Теперь нужно доказать алгоритм в ситуации, когда обе точки располагаются на границе прямоугольника.

Случай 1: Обе точки находятся в диагонально противоположных углах. Они не могут быть соседними, то есть расстояние между ними больше, чем $\sqrt{2}$, в противном случае площади была бы квадратом 2×2, и тогда нет тупоугольного треугольника. Тогда точка справа от одной из них вместе с остальными образует тупоугольный треугольник.

Случай 2: Обе точки находятся в углах, но не по диагонали. Расстояние между точками не менее 3, иначе ответа не существует. Возьмем точку, которая находится по диагонали от точки A (на расстоянии $\sqrt{2}$) — она будет искомой точкой.

Случай 3: Одна из точек находится не в углу, точки имеют общую координату. Пусть точка A в углу, тогда возьмем точку C по диагонали от точки B (на расстоянии $\sqrt{2}$), так, чтобы угол $\angle CBA$ был тупым.

Случай 4: Одна из точек находится не в углу, точки не имеют общих координат. Если расстояние между точками больше, чем $\sqrt{2}$, то мы можем представить их вершинами прямоугольника меньшего размера, и свести этот случай к Случаю 1. Значит эти точки соседние по диагонали, причем лежат на границе. Прямоугольник больше чем квадрат 2×2 (в этом случае ответа нет), а значит он хотя бы 3×2. Точка, лежащая на стороне прямоугольника, соседняя с одной из данных, будет образовывать тупоугольный треугольник (а такая есть, поскольку прямоугольник хотя бы 3×2).

Проверить, является ли треугольник тупоугольным несложно. Нужно проверить, что его площадь ненулевая с помощью векторного (псевдоскалярного) произведения векторов сторон треугольника. Пусть есть вектора $\vec{AB}$ и $\vec{A}$, тогда должно выполняться условие $AB_x \cdot AC_y - AB_y \cdot AC_x \neq 0$.

Чтобы проверить, является ли угол тупым, нужно посмотреть на знак скалярного произведения векторов, образующих угол, он должен быть отрицательным. Пусть есть вектора $\vec{AB}$ и $\vec{A}$, тогда должно выполняться условие $AB_x \cdot AB_x + AC_x \cdot AC_y < 0$.

Первая подзадача, $n, m \leqslant 1000$

Можно было перебрать все возможные точки, и проверить каждую, является ли она ответом с помощью вышеописанной проверки. Решение работает за $O(nm)$.

Вторая подзадача, $1000 \leqslant n, m \leqslant 10^9$

Поле было большое по обеим размерностям, поэтому хорошо работает выбор случайной точки. В подгруппе можно было не проверять треугольник на вырожденность, потому что вероятность попасть в отрезок очень маленькая.

Третья подзадача, $x_1 \neq x_2, y_1 \neq y_2$

У нас есть правая точка и левая точка. Давайте возьмем левую точку, отступим от нее на один вправо (это возможно, так как отрезок не вертикальный и не горизонтальный). Полученный треугольник тупоугольный, кроме случая, когда он прямоугольный (то есть $x_l + 1 = x_r$). В этом случае, отступим на один вниз, если правая точка ниже, и вверх, если правая точка выше.

Решение на 70 баллов

Просто будем перебирать случайные точки, и проверять каждую.

Задача 2. Санкт-Петербург

Для начала рассмотрим ряд простых решений, которые не будут набирать полный балл за каждый тест:

- Можно вывести заявку $[x..y]$, где $x = \min a_i, y = \max a_i$. Такое решение получает 11 баллов.
- Если длина заявки в предыдущем решении более l символов, то вместо x можно написать число вида 99..9 нужной длины. Такое решение вместе с предыдущим получает 21 балл.
- Отсортируем изначальный массив a. Теперь будем поддерживать отрезки чисел, которые будут в ответе. Изначально у нас будет n отрезков $s_1 = [a_1, a_1]$, $s_2 = [a_2, a_2]$, ..., $s_n = [a_n, a_n]$. Далее будем делать следующие операции:
  1. Найдем такой индекс i, что $s_{i+1}.left - s_i.right$ максимально.
  2. Объединим i и $i + 1$ в один отрезок $[s_i.left, s_{i+1}.right]$.
  3. Если длина текущей заявки с объединенными отрезками превысила l символов, то остановим наш процесс. Иначе перейдем к пункту 1.

  Такое решение вместе с предыдущим набирает 39 баллов.

Теперь рассмотрим оптимальные решения:

Первая подзадача, $n, l \leqslant 20$

Отсортируем изначальный массив. За 2^n переберем, будем ли ставить «,» после i-го числа. Числа, между которыми нет запятых склеим в один диапазон. Проверим, что длина удовлетворяет условиям и обновим ответ.

Если таким перебором мы не нашли ответ, то можно для некоторых k заменить минимальное число $> 10^k$ на $10^k - 1$. Так как все числа до 10^9, то можно перебрать нужные k за 2^9.

Время работы $\mathcal{O}(2^9 \cdot 2^n \cdot n)$. Но в строку длины 20 помещается не более 9 чисел, поэтому если перебирать рекурсивно и не делать строку длины больше l, то решение работает быстро.

Вторая подзадача, $n, l \leqslant 100$

Вместо перебора воспользуемся динамическим программированием. Пусть $f_{i,j}$ = минимальное количество лишних чисел в заявке, если мы добавили в заявку первые i чисел, текущая длина заявки j. Тогда:

Если мы записываем одно число, то $f_{i,j} = f_{i-1,j-(length(a[i])+1)}$.

Если мы записываем диапазон, то переберем индекс предыдущего числа t. Тогда $f_{i,j} = \min_{t<i}(f_{t,j-cur})$, где $cur = 1 + length(a_{t+1}) + 2 + length(a_i)$.

Кроме того нужно не забыть про числа вида 99..9, которые экономят нам длину. Пусть $x = 10^q - 1$, t — максимальный индекс, что $a_t < x$. Тогда $f_{i,j} = f_{t,j-cur}$, где $cur = 1 + q + 2 + length(a_i)$.

Тогда оптимальное количество лишних чисел будет лежать в $f_{n,l}$. Восстановим ответ и выведем заявку.

Время работы $\mathcal{O}(l \cdot n^2)$.

Третья подзадача, $n, l \leqslant 5000$

Совмещением предыдущей динамики с жадным решением можно набрать 83 балла.

Заметим, что cur в предыдущих формулах зависит только от длины числ a_t. Поэтому для фиксированного j мы можем посчитать лучшее значение i для каждой длины числа от 1 до 10. С помощью такого массива $opt[j][1..10]$ можно пересчитывать динамику быстрее. Остальное остается как в предыдущем решении.

Время работы $\mathcal{O}(l \cdot n)$.

Альтернативное решение

Пусть $f_{i,j,t}$ = минимальное количество лишних чисел, если i чисел добавили, длина заявки j. Если текущий диапазон открыт к добавлению новых чисел, то $t = 1$. Иначе $t = 0$. Переходы:

- Открыть диапазон: $f_{i,j,0} \Rightarrow f_{i+1,j+1+len(a_i),1}$.
- Добавить число в диапазон: $f_{i,j,1} \Rightarrow f_{i+1,j,1}$.

- Закрыть диапазон: $f_{i,j,1} \Rightarrow f_{i+1,j+2+len(a_i),0}$.
- Добавить одно число: $f_{i,j,0} \Rightarrow f_{i+1,j+1+len(a_i),0}$.

Задача 3. Нижний Новгород

Первая подзадача, $k = 1, c = 0$

Нет обработанных точек, очаг только один. Область заражения будет выглядеть как «алмаз» и будет содержать $1, 5, 13, 25\ldots$ точек для $T = 0, 1, 2, 3, \ldots$. Один из способов найти эти числа - проверить, сколько точек на каждой координате x. Так мы получаем ответ

$$(1 + 3 + 5 + \cdots + 2T - 1) + 2T + 1 + (2T - 1 + 2T - 3 + \cdots + 3 + 1),$$

за исключением $T = 0$, потому что тогда ответ будет 1. Две суммы в скобках являются арифметическими прогрессиями. Если мы сгруппируем внутри скобок слагаемые в пары, мы получим

$$(1 + 2T - 1) + (3 + 2T - 3) + \cdots + (2T - 1 + 1) = 2T + 2T + \ldots 2T = 2T^2.$$

Таким образом, ответ: $2T^2 + 2T + 1$.

Вторая подзадача, $c = 0, T \leqslant 100, x_i, y_i < 100$

Область заражения будет выглядеть как несколько «алмазов», которые перекрываются. Есть несколько разных способов решения этой группы. Одним из способов является постепенное моделирование распространения. На каждом этапе мы просматриваем все зараженные клубни и отмечаем те, которые находятся рядом с любым из них.

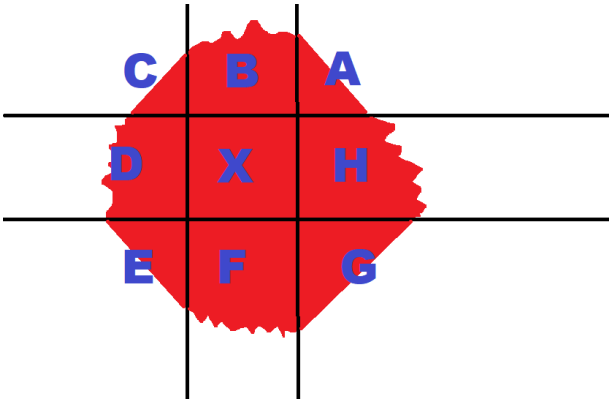
Другой способ состоит в том, чтобы заметить, что зараженные клубни — это те, которые находятся на расстоянии Манхэттена не более T от любой из исходных зараженных точек. Манхэттенское расстояние между двумя точками (x_1, y_1) и (x_2, y_2) определяется как $|x_1 - x_2| + |y_1 - y_2|$. «Алмазы» — это круги в метрике Манхэттена! Поскольку $T \leqslant 100$, все зараженные клубни будут в пределах квадрата 300 $times$300, поэтому, чтобы увидеть, какие из них горят, мы можем пройти через все точки 300 $cdot$ 300 и для каждой проверить расстояние Манхэттена до любого из очагов заражения.

Третья подзадача, $T \leqslant 400, x_i, y_i < 100$

Решение состоит в том, чтобы смоделировать процесс, и сделать это достаточно эффективно. Также нужно обрабатывать политые химикатами клубни.

Четвертая подзадача, $c = 0, x_i, y_i < 100$

Важным наблюдением является то, что «стартовая зона» очень мала по сравнению с T. Таким образом, после того, как жуки распространятся T раз, стартовая зона будет почти похожа на точку, поэтому зона заражения должна выглядеть почти как манхэттенский круг. Предположим, что $T > 200$, если это не так, мы можем смоделировать процесс. Разделим плоскость на 9 областей, как показано на рисунке.



Область X является стартовой рамкой, т.е. все точки, которые удовлетворяют 0 $leqx, y$ leq99. Поскольку $T > 200$, вся стартовая область будет заражена, поэтому она вносит 100 $cdot$ 100 в ответ. Давайте посмотрим, какой вклад вносят другие области.

1. Область A состоит из всех точек, которые удовлетворяют $x, y \geqslant 100$. Точка (x, y) в области A заражена, если $|x - x'| + |y - y'| \leqslant T$ для любой из заданных точек (x', y'). Но $|x - x'| + |y - y'| = (x + y) - (x' + y')$, поскольку $x > x', y > y'$, поэтому нам нужно заботиться только о точке, которая имеет наибольшее значение $x' + y'$. Таким образом, область представляет собой манхеттенский круг, и количество точек можно рассчитать с использованием вычислений подзадачи 1.

2. Область B состоит из всех точек, которые удовлетворяют $y \geqslant 100, 0 \leqslant x \leqslant 99$. Давайте зафиксируем одну из x координат x_0 и посмотрим, какой столбец x_0 дает в ответ. Точка (x_0, y) есть в B, если $|x_0 - x'| + |y - y'| \leqslant T$ для некоторого очага заражения (x', y'). Но $|x_0 - x'| + |y - y'| = y + |x_0 - x'| - y'$, поэтому нам нужно только найти очаг возгорания с максимальной величиной $|x_0 - x'| - y'$, чтобы найти высоту столбца. Таким образом, мы можем просмотреть все 100 значений x_0 и выяснить, какой вклад вносит каждый из них.

Остальные области могут быть решены аналогичным образом.

Пятая подзадача, $x_i, y_i < 100$

Решение на самом деле очень похоже на предыдущее, мы сможем упростить задаче до $= 0$. Начните с моделирования процесса, первые, скажем, 500 ночей. Если зараженные клубни полностью окружены обработанными, то таким образом мы сможем это определить и вывести ответ. В противном случае жуки будут окружать все обработанные клубни к этому времени. Поэтому мы можем игнорировать внутреннюю часть зоны заражения, где расположены все опрысканные клубни, поскольку там ничего нового не произойдет. Зона заражения теперь может рассматриваться как большой тестовый случай с ($= 0$) и может быть решена с использованием метода из предыдущей подзадачи. Правда, новая область составляет 1000×1000 клубней, и может содержать значительно больше, чем 100 зараженных точек, поэтому мы должны быть немного осторожнее в реализации. Один из способов справиться с этим — просто посмотреть на пораженные клубни, которые находятся на краю зоны заражения. Поскольку таких точек около 4000, решение из предыдущей подзадачи теперь вполне укладывается.

Шестая подзадача, $c = 0$

Это подзадача такая же, как и четвертая, хотя координаты могут быть большими, до 10^5. Если $T \geqslant 2 \cdot 10^5$, мы можем фактически использовать то же решение, что и в подзадаче 4. Основные операции в той задаче нужны для того, чтобы посчитать, какой вклад вносят столбцы. Поскольку существует $4 \cdot 10^5$ столбцов, требуется около $4 \cdot 10^5 \cdot n = 4 \cdot 10^7$ операций, что достаточно мало.

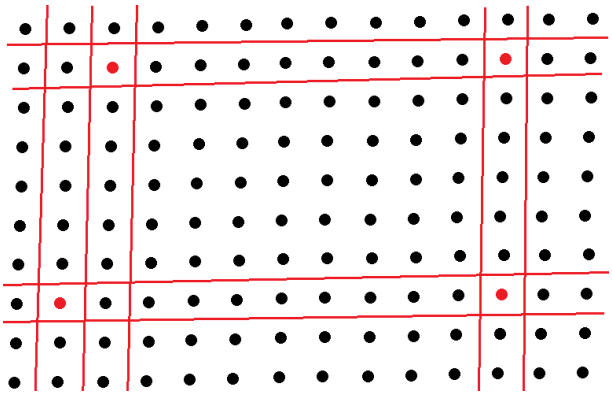
Поэтому, нужно придумать что-то новое для случаев, когда T находится где-то в середине от максимального значения. Наиболее сложным является вычисление количества точек заражения в стартовой области, поскольку даже если $T \leqslant 2 \cdot 10^5$, мы сможем использовать те же методы, что и раньше, для вычисления площади заражения в других областях. Чтобы рассчитать количество точек заражения в стартовой зоне, мы можем сделать так называемую декомпозицию. Давайте переберем x координату (зафиксируем некоторое x_0). Мы хотим вычислить количество точек горения в виде (x_0, y), где $0 \leqslant y < 10^5$. Каждая заданная точка (x', y') даст интервал на этой линии, который указывает, какие кусты находятся на расстоянии не превосходящем T от этого очага. Теперь задача состоит в том, чтобы вычислить количество точек, покрываемых любым из этих n интервалов, и это можно решить за время $O(n \log(n))$, если вы сначала отсортируете интервалы и используете «сканирующую прямую».

Альтернативный подход. Заметим, что наши «алмазы» заражения, это квадраты, повернутые на 45 градусов. Повернем плоскость на 45 градусов, и теперь просто найдем площадь объединения прямоугольников за $O(n \log(n))$, что является стандартной задачей на сканирующую прямую. Поворот плоскости на 45 градусов выполняется по правилу $(x, y) \to (x - y, x + y)$. При таком преобразовании, немного изменятся площади, но это нетрудно учесть.

Седьмая подзадача В пятой подзадаче мы пользовались моделированием, для избавления от обработанных клубней. В этой задаче моделировать слишком долго, поэтому будем действовать подругому. Декомпозиция из прошлой задачи тоже не работает, поскольку интервалы от одного очага могут формировать не один отрезок, а несколько, и их долго вычислять.

Один факт, который мы еще не использовали, заключается в том, что количество интересных клубней всего 200 штук. Мы могли бы использовать сжатие координат: обрабатывать данные в условии координаты как «важные», а остальные как «равные».

Давайте возьмем каждую заданную точку и нарисуем четыре прямые, которые идут между точкой и каждым из ее соседей, параллельно осям x и y . Эти линии разделят плоскость на несколько прямоугольников (около $4 \cdot (n+m)^2$ штук). Интересный факт об этих прямоугольниках — то, что заражение сначала достигает их угла. Если заражение достигнет прямоугольника на стороне, а не в углу, то будут интересные координаты вдоль стороны, и тогда прямоугольник не должен существовать по построению.



Таким образом, для каждого прямоугольника мы хотим вычислить, когда жуки впервые достигнут каждого из его четырех углов. Это легко решить с помощью графом — нам нужно найти кратчайшее расстояние до углов прямоугольника от некоторых других вершин. Давайте построим граф, где все углы являются вершинами. Проведем ребра между парами вершин, которые принадлежат одному и тому же прямоугольнику, где вес — это расстояние между точками. Кроме того, проведем ребра между углами соседних прямоугольников. Для обработки химикатных клубней нам просто нужно избегать проведения каких-либо ребер в этих вершинах. Теперь мы можем найти кратчайший путь от точек горения до всех углов с помощью алгоритма Дейкстры, и мы почти решили задачу. Все, что остается, это, учитывая, что заражение достигает всех четырех углов, вычислить, сколько точек заражается в каждом прямоугольнике. Это можно решить математически, но, поскольку координаты настолько малы, вы можете сделать декомпозиции, подобные тем, что были в предыдущих подзадачах.

Задача 4. Екатеринбург

Во подзачах будем использовать $K = 20$

Подзадача 1, $N \leqslant 20$

Стоим на месте, поддерживаем номер команды, равный заработанным бублям. Из $\langle$`S, i`$\rangle$ переходим в $\langle$`S, i+1`$\rangle$ и зарабатываем бубль. В $\langle$`S, N-1`$\rangle$ зарабатываем бубль и заканчиваем работы ($N = 0$ рассмотреть отдельно).

Подзадача 2, $N \leqslant 60$

Также стоим на месте, но используем то, что можно изменять владельца завода.

- $\langle$`S, i`$\rangle \Rightarrow \langle$`S, i+1`$\rangle$
- $\langle$`0, i`$\rangle \Rightarrow \langle$`0, i+1`$\rangle$
- $\langle$`1, i`$\rangle \Rightarrow \langle$`1, i+1`$\rangle$
- $\langle$`S, 19`$\rangle \Rightarrow \langle$`0, 0`$\rangle$
- $\langle$`0, 19`$\rangle \Rightarrow \langle$`1, 0`$\rangle$

Везде зарабатываем бубль, и как только будет набрано $N-1$ бублей заработываем последний бубль и заканчиваем работы.

Подзадача 3, $N \leqslant 120$

Теперь будем использовать возможность передвигаться между заводами. Пройдем вправо на 11 заводов, и на каждом из них заработаем по 10 бублей. Сделаем это следующим образом, первые 10 команд рабочих отведем на движение вправо, а остальные 10 на заработок бублей.

- $\langle$`S, 0`$\rangle$: `R S 0`
- $\langle$`0, i`$\rangle$: `R 0 i+1` для `i` $= 0, 1, \ldots, 9$
- $\langle$`0, i`$\rangle$: `SC 0 i+1` для `i` $= 10, 11, \ldots, 18$
- $\langle$`0, 19`$\rangle$: `LC 0 10`
- $\langle$`S, i`$\rangle$: `SC S i+1` для `i` $= 10, 11, \ldots, 18$
- $\langle$`S, 19`$\rangle$: `HC`

Это набор инструкций для того, чтобы набрать 120 бублей, чтобы набрать N бублей, будем использовать не 11 заводов нулевой компании, а $\lfloor \frac{N}{10} \rfloor$. А остаток урегулируем на заводе S.

Подзача 4, $N \leqslant 399$

В предущей подзаче мы опять забыли про возможность менять владельца у завода. Теперь давайте пойдем вправо на 20 заводов, зарабатывая бубли и меняя владельца на компанию 1 у всех, кроме последнего.

- $\langle$`S, 0`$\rangle$: `R S 0`
- $\langle$`0, i`$\rangle$: `RC 1 i+1` для `i` $= 0, 1, \ldots, 18$

Теперь мы имеем 19 заводов компании 1 справа от завода S и сразу после них завод компании 0, около которого находимся мы с командой номер 19. Пойдем влево до завода стеклопакетов и на каждом заводе компании 1 будем зарабатывать еще 20 бублей.

- $\langle$`0, 19`$\rangle$: `L 0 19`
- $\langle$`1, i`$\rangle$: `SC 1 i-1` для `i` $= 1, 2, \ldots, 19$
- $\langle$`1, 0`$\rangle$: `LC 1 19`
- $\langle$`S, 19`$\rangle$: `H`

Это набор инструкций для того, чтобы набрать 399 бублей (по 21 бублю на каждом из 19 заводов справа от завода S). Чтобы набрать N бублей, на каждом из 19 заводов будем зарабатывать по $\lfloor \frac{N}{19} \rfloor$ бублей, а остаток наберем в конце на заводе S (нужно будет набрать не более 18 бублей).

Подзадача 5, $N \leqslant 500$

Здесь появляется идея, которая далее будет развиваться оптимизациями до 100 баллов. Будем поддерживать так называемый счетчик, записав его двоичную запись на заводах (компании 0 и 1). Запишем число N в счетчик, и будем уменьшать его на 1 (постепенно с этим зарабатывая бубль), пока счетчик не станет равен 0.

Для этого отведем 9 команд на то, чтобы записать число N

- $\langle$`S, 0`$\rangle$: `R S 0`
- $\langle$`0, i`$\rangle$: `R (N»i)&1 i+1` для `i` $= 0, 1, \ldots, 8$

Теперь нам нужно одно состояние (9) для того, чтобы вернуться к заводу S для начала очередного вычитания единицы. Для вычитания заменяем все 0 на 1, пока не встретим завод компании 1, его заменяем на 0, например, $110100 - 1 = 110011$. Для вычитания необходимо еще 9 команд рабочих (для поддержания текущего номера бита, $10 - 18$). Как только счетчик станет равен 0, а это значит, что при вычитании мы дошли до 8 бита и он равен 0, заканчиваем работы.

- $\langle$`0, 9`$\rangle$: `L 0 9`
- $\langle$`1, 9`$\rangle$: `L 1 9`
- $\langle$`S, 9`$\rangle$: `R S 10`
- $\langle$`0, i`$\rangle$: `R 1 i+1` для `i` $= 10, 11, \ldots, 17$
- $\langle$`1, i`$\rangle$: `LC 0 9` для `i` $= 10, 11, \ldots, 18$
- $\langle$`0, 18`$\rangle$: `H`

Можем получить любое $N < 512$.

Подзадача 6, $N \leqslant 1\,000$

У нас осталась одна неиспользуемая команда рабочих номер 19. При уменьшении счетчика на 1 будем зарабатывать не один бубль, а сразу два, используя эту команду рабочих. Теперь будем записывать изначально на счетчик $\lfloor \frac{N}{2} \rfloor$. В случае нечетного N заработаем еще один бубль при завершении работ.

Подзадача 7, $N \leqslant 5\,000$

Еще одна оптимизация обычного счетчика. Заметим, что мы никак не используем команды рабочих на заводе S. Будем перед каждым вычитанием зарабатывать по 10 бублей на заводе S, прокручивая команды. Будем использовать для этого команды с номерами $9, 10, \ldots, 18$. Теперь мы можем получить любое N, кратное 10. Необходимо как-то набрать остаток от деления на 10, будем использовать для этого команду рабочих 19. При попадании в состояние $\langle$`0, 18`$\rangle$, сменим команду рабочих на 19, пойдем влево до завода S и заработаем недостающие бубли с помощью команд 19 и $1, 2, \ldots, 8$, и только потом завершим работы.

Подзадача 8, $N \leqslant 130\,000$

Для этой и следующих подзадач основной идеей является не уменьшать счетчик, а увеличивать его. После каждого увеличения будем проверять счетчик на равенство N. Для этого необходима одна команда рабочих для операции увеличения счетчика, одна команда для возврата на завод S после увеличения счетчика и еще одна команда для возврата после проверки счетчика на равенство N. Остальные команды можно использовать для поддержания номера бита при проверке счетчика на равенство, то есть 17 команд. Итого можно набрать любое количество бублей $< 2^{17}$.

Подзадача 9, $N \leqslant 260\,000$

Оптимизация аналогичная оптимизации в подзадаче 6, но без дополнительной команды рабочих. Как только мы начинаем операцию увеличения счетчика, будем зарабатывать дополнительный бубль. В таком случае мы заработаем $2X + 1$ бублей, где X — число, на равенство которого мы проверяем счетчик. Значит, если X будет равен $\lfloor \frac{N-1}{2} \rfloor$, то мы заработаем N бублей, в случае нечетного N и $N - 1$ в случае четного, но в таком случае отправим команду рабочих на завод при завершении работ.

Подзадача 10, $N \leqslant 600\,000$

Оптимизация аналогичная оптимизации в подзадаче 7, но для инкрементирующего счетчика.

Подзадачи 11-16, $N \leqslant 22\,600\,000$

Можно пройти еще некоторые группы, комбинируя предложенные выше оптимизации, но для полного решения этого будет недостаточно. Главной идеей для следующего шага является забыть про число N при написании инструкций. Основным наблюдением является то, что максимальное число бублей, которое мы можем набрать ограничено количеством выполняемых инструкций. Запустим решения до его максимального значения и запишем для каждой инструкции, сколько раз она выполняется.

Теперь нам нужно выбрать, в каких инструкция нужно добавить команду для заработка бубля. Для каждой инструкции у нас есть числа ее исполнений, причем многие инструкции являются степенями двойки, возможно с небольшими множителями из-за последующих оптимизаций. Отсортируем операции по уменьшению числа их исполнений, и используем жадный подходи при выборе операций, в которые мы отправим команду. Заметим, что любое N до максимального значения возможно набрать.

Далее будут предложены некоторые оптимизации, которые увеличивают максимальное количество выполняемых инструкций.

Улучшенный счетчик. Так как нас интересует только максимальное значение, оставим только команду рабочих для возврата к заводу S после увеличения счетчика.

- $\langle$`S, 0`$\rangle$: `R S 1`
- $\langle$`1, i`$\rangle$: `R 0 i+1` для `i` $= 1, 2, \ldots, 18$
- $\langle$`0, i`$\rangle$: `L 1 0` для `i` $= 1, 2, \ldots, 19$
- $\langle$`0, 0`$\rangle$: `L 0 0`
- $\langle$`1, 0`$\rangle$: `L 1 0`
- $\langle$`1, 19`$\rangle$: `H`

Такой счетчик делает $\sim 2^{19} \cdot 4$ итераций.

Небольшая оптимизация движений. Заметим, что если при определенной команде рабочих мы двигаемся только влево, то при «переключении» на нее можно пододвинуться вправо. Это увеличит количество выполненных инструкций на $\sim 2^{19} \cdot 2$. Нужно заменить следующие инструкции из предыдущего пункта:

- $\langle$`0, i`$\rangle$: `R 1 0` для `i` $= 1, 2, \ldots, 19$

Используем все команды рабочих на S. Мы опять же используем на заводе S только команду 0. Будем каждый раз прокручивать все команды рабочих на заводе S следующим образом:

- $\langle$`S, i`$\rangle$: `S S i+1` для `i` $= 0, 1, \ldots, 18$
- $\langle$`S, 19`$\rangle$: `R S 1`

Это увеличит количество выполненных инструкций на $\sim 2^{19} \cdot 19$.

Долгое увеличение. Заметим, что очень часто при увеличении счетчика мы останавливаемся на ранних битах числа. Давайте как только мы найдем первый 0 в числе, не будем сразу менять его на 1, а будем увеличивать показатель бита до максимально возможно, но позицию не менять, и только потом изменим владельца завода на 1. Изменим следующие инструкции:

- $\langle$`0, i`$\rangle$: `S 0 i+1` для `i` $= 1, 2, \ldots, 18$
- $\langle$`0, 19`$\rangle$: `R 1 0`

Это увеличит количество выполненных инструкций на $\sim 2^{19} \cdot 17$.

Заключение. Итого если использовать все предложенные оптимизации, то можно достичь $\sim 2^{19} \cdot 42$ итераций.

Задача 5. Липецк

Общее замечание

Поскольку мы можем быстро открывать и закрывать все люки, то мы можем для каждого из них решить независимо — открывать люк или нет, когда по нему пробегает кролик.

Первая подзадача, $n, m, t \leqslant 200$

Пусть каждая клетка является вершиной графа, и у каждой вершины есть ребро в соседа справа. Добавим для люков ребро в соседа снизу. В полученном графе из nm вершин и $O(nm)$ ребер нужно t раз проверить существование пути между двумя вершинами. Запустим t раз алгоритм обхода графа из стартовой вершины. Асимптотика решения $O(tnm)$.

Еще общее замечание

Если нам нужно спуститься на этаж ниже, то будем делать это как можно раньше. Очевидно, если ответа таким образом мы не достигнем, то спустившись позже мы его тем более не достигнем.

И наоборот, если можно достигнуть красную плиту, спустившись позже, то мы можем достигнуть его, спустившись раньше, и пробежав разницу направо на другом этаже.

Вторая подзадача, $n, m \leqslant 2500, t \leqslant 4000$

Будем спускаться каждый раз, когда находимся на люке, и финишная клетка находится ниже. Промоделируем процесс движения кролика для каждого уровня. Путь кролика содержит в худшем случае $n + m$ плит, запустим процесс t раз. Решение работает $O(t(n + m))$.

Третья подзадача, $x'_j - x_j = 1$

Необходимо проверить, есть ли на этаже x_j люк в позициях с y_j по y'_j включительно. Поскольку этажей много, нужно применить сжатие координат. Например, будем хранить map из номера этажа в отсортированный список позиций люков на этом этаже. Также во всех группах заходило хранить значением map'а множество (set) всех люков, или номера люков этого этажа в общем отсортированном списке.

Теперь запустим двоичный поиск на этом массиве (или воспользуемся аналогичным методом в set), и определим наименьшую позицию люка, большую либо равную y_j.

Четвертая подзадача, $k, t \leqslant 5000$

Необходимо для каждого люка предподсчитать величину $next_i$ — номер ближайшего люка на следующем этаже к люку i. Такую величину можно предподсчитать с помощью двоичного поиска/lower_bound, то есть способа, описанного в третьей подзадаче.

Теперь будем моделировать путь между стартовой и конечной клетками кролика, но участки, когда мы движемся направо, дуем проскакивать быстро. Другими словами, будем передвигаться от люка к люку. Очутившись на нужном этаже, нужно проверить, правда ли красная клетка находится не левее, чем клетка, на которой мы оказались. Если на искомом этаже оказаться невозможно, пути тоже нет. Итого мы для каждого уровня будем определять достижимость за количество люков, то есть общая сложность $O(tk + klogk)$.

Альтернативой двоичному поиску является использование метода двух указателей. Если мы будем поддерживать второй указатель на ближайшем справа люке, находящемся на следующем этаже, относительно i-го люка, то оба указателя сдвинутся $O(k)$ раз, и общая асимптотика будет $O(tk)$, но это не требовалось в задаче.

Пятая подзадача

Теперь нужно попробовать идти не по всем люкам, а делать это быстрее. Давайте предподсчитаем не только позицию люка на следующем этаже, но и позицию люка через 2 этажа, через 4 этажа, через 8 этажей и так далее. То есть теперь у нас будет величина $next'[i][p]$ — номер ближайшего люка на этаже $i + 2^p$. Очевидно, что $next'[i][0] = next[i]$. Чтобы вычислить $next'[i][p]$ в общем виде, нужно заметить, что прыгнуть на 2^p этажей, это тоже самое, что два раза прыгнуть на 2^{p-1} этаж. То есть $next'[i][p] = next'[next'[i][p-1]][p-1]$ Не забываем аккуратно обработать ситуацию, когда для одного из люков в прыжках следующего люка не существует. Такой предподсчет имеет $O(klogk)$ состояний, каждый из которых пересчитывается за константное время.

Теперь научимся проверять существование пути. Для каждого пути мы знаем количество этажей, на которое нужно прыгнуть, обозначим его за X. Будем перебирать размер прыжка начиная от наибольшей степени p в сторону уменьшения. То есть если прыгнув на 2^p этажей, мы не спустимся ниже необходимого ($2^p \leqslant X$), то прыгнем на эту величину, и уменьшим X на 2^p. Если какой-то из таких прыжков невыполним, из-за отсутствия необходимого люка, скажем, что уровень невыполним. В конце проверим, что позиция, на которой мы можем оказаться, преодолев исходные X этажей не правее требуемой.

Таким образом на запрос мы умеем отвечать за $O(logk)$, и общая сложность решения $O(klogk + tlogk) = O((k + t)logk)$.

Задача 6. Челябинск

Для решения задачи заметим следующий факт:
Рассмотрим какой-то отсортированный по возрастанию *идеально сбалансированный* массив различных чисел, а именно как в нём будут разбиваться элементы по парам. С кем пойдёт самое маленькое число? Пусть не с максимальным, но тогда $min + a < b + max$, так как $min < b$ и $a < max$. А такого быть не может. Тогда пары всегда будут образовываться из 1-го и последнего, 2-го и предпоследнего

и так далее.

Подгруппа 1.

Определим $M = N + K$. Так как $K = 1$, то мы можем перебрать 1-го лишнего участника и проверить на сбалансированность массив без него за $O(M)$, для этого нужно проверить что все пары вида первый с последним, второй с предпоследним, дают одинаковую сумму. Получаем решение за $O(M^2)$.

Подгруппа 2.

Заметим, что если мы знаем сумму X в каждой паре, то мы можем с помощью, например, *unordered_map*-а проверять есть ли *идеально сбалансированная* последовательность из n элементов с суммой в каждой паре X. Для этого будем идти по всем элементам и пытаться дать им в пару элемент $X - a_i$, который найдем в нашей хеш-таблице.

Тогда для решения этой подзадачи нужно перебрать 3 варианта X(первый с последним, первый с предпоследним и второй с последним), так как среди них точно есть верный X. Это решение за $O(M)$.

Подгруппа 3.

То же самое, что и в прошлой подгруппе, только нужно перебрать максимум 9 вариантов X(3 элемента на префиксе и 3 на суффиксе).

Подгруппа 4.

Так как M - маленькое, то можем решить за $O(M^3)$: перебрать все возможные варианты $X = a_i + a_j$ и запустить на них проверку.

Подгруппа 5.

В этой подгруппе можно полным перебором за $O(2^M)$ найти нужную нам подпоследовательность и проверить, что все суммы пар одинаковые(первый с последним, второй с предпоследним и так далее).

Подгруппа 6.

Воспользуемся идеей из 3 и 4 подгруппы: найдем нужный $X = a_i + a_j$, где i принадлежит префиксу из $k + 1$ элемента, а j из суффикса из $k + 1$ элемента, и запустим проверку.

Подгруппа 7.

То же самое что и в последней подгруппе, но нужно избавиться от константы *unordered_map*, а именно будем идти с 2-мя указателями и набирать каждую пару отдельно: если текущая сумма равна X, то сдвинем оба указателя, если сумма меньше X, то сдвинем левый указатель вправо, если же сумма больше, то сдвинем правый влево.

Подгруппа 8.

Для полного решения нужно как-то уменьшить количество рассматриваемых X. Тогда давайте, если $M < 4 * K$ запустим решение с перебором всех X за $O(M^3)$. Иначе же посмотрим на все возможные суммы $X = a_i + a_j$, где i из префикса длины $2K$, а j из суффикса длины $2K$. Но тогда понятно, что X из оригинальной подпоследовательности длины N встретится хотя бы K раз(так как с каждой стороны максимум K "лишних"элементов). Но таких X, которые встречаются хотя бы K раз не больше $\frac{2*K*2*K}{K} = 4 * K$. Тогда мы можем рассматривать только те X, которые встретились хотя бы K раз. В итоге решение за $O(K^2 + M * K)$.

Другое решение: будем брать случайный элемент a_i и искать ему пару из оригинальной последовательности среди элементов с индексами $[n - i - k, n - i + k]$ и проверять их сумму. Если сам a_i — элемент оригинальной последовательности, то его пара точно будет в этом промежутке, так как лишние элементы могли выместить его максимум на k позиций. Это решение работает быстро, потому что n - большое, а k - маленькое, поэтому шанс попасть в элемент из оригинальной последовательности высок.

Задача 7. Ижевск

Нумерация тестов в разборе совпадает с обычной.

Второй и третий тесты разбираются руками, они не очень большие.

Также можно написать перебор: перебирать число в клетке от младшего к старшему, учитывая делимость в вертикальной и горизонтальной области. Если число последнее в области, то значение определяется однозначно. Такое решение приносит баллы примерно в половине тестов.

В четвёртом тесте дан прямоугольник 100×1, в котором только вертикальные области. Как строится ответ для фиксированной области? Разбиваем число на простые множители с учётом кратности и в каждую клетку пишем по одному простому множителю (если длина области меньше, чем число простых множителей с учётом кратности, то в последнюю клетку запишем все неизрасходованные числа).

Аналогичными соображениями можно решить девятый тест, научившись обрабатывать пересечения в углах. Заметим, что если в пересечении стоит не единица, а в одной из областей есть единичная клетка, не являющаяся пересечением с другой областью, то можно убрать это число из пересечения в другие клетки, не ухудшив ответ. Таким образом, в пересечении стоит не единица, только если для горизонтальной области (области были короткие) количество простых делителей с учётом кратности не меньше длины области. В таком случае относительно вертикальной области надо поставить в оба пересечения какой-то делитель наибольшего общего делителя (если такое, конечно, возможно).

Давайте решим в общем случае задачу. Для начала научимся находить любую корректную расстановку. Построим двудольный граф для каждого простого делителя, где в левой доле будут вертикальные области, в правой - горизонтальные, а ребро проведено, если области пересекаются и произведения обеих имеют данный простой делитель в разложении. Будем считать, что все клетки лежат и в горизонтальной, и в вертикальной области (если клетка лежит только в одной, то создадим фиктивную область с нефиксированным произведением). Заметим, что этот граф можно превратить в сеть: сделаем исток, из него рёбра будут вести в левую долю с пропускной способностью, равной степени вхождения простого в произведение (аналогичные пропускные способности будут у рёбер из правой доли в сток), из левой доли будут вести рёбра в правую, как указано выше с пропускной способностью ∞. Для фиктивных областей пропускная способность равняется ∞. Заметим, что максимальный поток в таком графе расставит числа, удовлетворяя условию.

Дальше требуется соптимизировать решение, чтобы получить наименьшее число единиц. Можно сделать между областями два ребра: одно стоимости 0 и пропускной способности 1, а второе стоимости ∞ и пропускной способности ∞. После этого найти поток алгоритмом $mincost - maxflow$. Таким способом мы стремимся не ставить в пересечение одно и то же простое число больше одного раза. Но это не гарантирует оптимальности решения, потому что разные простые мы никак не контролируем.

С другой стороны, можно сделать все рёбра пропускной способностью единица, а недостающие делители после нахождения потока расставить при помощи аналога алгоритма Куна: у нас у каждой вершины стоит степень вхождения простого в неё, и нам требуется искать аналог удлиняющей цепочки. Предлагается делать следующее: мы проходим по ребру слева направо, увеличивая вхождения простого в оба конца, если справа стало больше, чем нужно, то проходим налево по ребру, где уже стоит это простое, уменьшая вхождения у обоих концов и т. д. Выбирать следующую вершину в пути можно из каких-нибудь жадных соображений или случайно.

Задача 8. Москва

Общие замечания.

- Если в кактусе n вершин, то в нем $\mathcal{O}(n)$ ребер.
- Переходить между маршрутами можно *жадно*. То есть находясь на станции, необходимо садиться на ближайший поезд следующего нужного маршрута.

Подгруппа 1.

В каждом проекте схема метро представляет собой дерево. На каждый запрос поиска кратчайшего расстояния между двумя вершинами запускаем обход графа.

Сложность $\mathcal{O}(n \cdot q)$.

Подгруппа 2.

Преподсчитаем ответ для каждой пары станций обходом дерева - dfs или bfs. По замечанию в начале нужно всегда жадно садиться на ближайший поезд. На запрос будем отвечать за $\mathcal{O}(1)$.

Сложность $\mathcal{O}(n^2 + q)$.

Подгруппа 3.

Будем использовать двоичные подъемы, т.е. прыжки на степени двойки из каждой вершины. Необходимо насчитать их при подъеме снизу вверх $up[v][k]$ и при спуске сверху вниз $down[v][k]$.

Научимся пересчитывать эти величины. Пусть уже посчитали $up[v][k]$, $k = 1 \dots t$, хотим вычислить $up[v][t+1]$. Через $up[v][t]$ минут можно подняться на 2^t вершин вверх и оказаться в вершине v. Остается дождаться ближайшего поезда наверх, обозначим это время Δ, тогда $up[v][t+1] = up[v][t] + up[u][t] + \Delta$. Величины $down[v][k]$ считаются аналогично в силу того, что поезда ходят по маршруту в обе стороны

Любой запрос - вертикальный путь или два вертикальных пути, соединенных вершиной. Чтобы найти эту соединительную вершину (LCA) можно использовать любой классический алгоритм, например, опять же двоичные подъемы.

Сложность построения $\mathcal{O}(nlogn)$. На каждый запрос отвечаем за $\mathcal{O}(logn)$.

Сложность $\mathcal{O}((n+q)logn)$.

Подгруппа 4.

В каждом графе ровно один цикл. На каждый запрос поиска кратчайшего расстояния между двумя вершинами запускаем обход графа. Заметим, что нет смысла крутиться на цикле несколько раз, поэтому достаточно лишь выбрать направление обхода цикла и пройти нужный участок. Т.к. цикл ровно один, то чтобы выбрать направление обхода, можно попробовать пройти по и против часовой стрелки и выбрать минимум.

Сложность $\mathcal{O}(n \cdot q)$.

Подгруппа 5.

Изменим решение подгруппы 3 в сторону полного решения. Нужно научиться работать с циклом. Заметим, что если путь попал на цикл, то нужно выбрать оптимальное направление обхода, а затем сойти с цикла. Для этого можно использовать, например, те же самые двоичные подъемы. Т.е. насчитаем прыжки на степени двойки от каждой вершины цикла влево и вправо аналогично величинам up. Теперь находить время на прохождение дуги цикла можно за $\mathcal{O}(logn)$. Остается выбрать минимальную по времени одну из двух дуг.

Сложность $\mathcal{O}((n+q)logn)$.

Подгруппа 6.

Предподсчитаем ответы между каждой парой станций, используя алгоритм Флойда или Беллмана-Форда. Пусть $dist[u][v]$ - минимальное время пути от u до v. Тогда ребра строятся динамически, т.е. вес ребра (s,t) основывается на величине $dist[u][s]$, к которой нужно прибавить время ожидание ближайшего поезда. Время ожидание можно находить за $\mathcal{O}(1)$ формулой.

На запрос будем отвечать за $\mathcal{O}(1)$.

Сложность $\mathcal{O}(n^3+q)$.

Подгруппа 7.

Предподсчитаем ответы между каждой парой станций, используя алгоритм Дейкстры. На запрос будем отвечать за $\mathcal{O}(1)$.

Это решение заходило и в 8 подгруппу.

Сложность $\mathcal{O}(n^2logn+q)$.

Подгруппа 8.

На каждый запрос будем обходить весь граф. Для каждого пути нужно выбрать оптимальный обход всех циклов, на который он попадает. Это можно делать для каждого цикла отдельно и независимо.

Сложность $\mathcal{O}(n \cdot q)$.

Подгруппа 9.

В этой подзадаче нужно собрать и реализовать все идеи из предыдущих подзадач. Сначала из кактуса нужно получить дерево. Чтобы выделить циклы, можно, например, использовать алгоритм поиска мостов. Далее строим на сжатых циклах двоичные подъемы. Внутри каждого цикла имеем построенные двоичные подъемы, описанные в подзадаче 5. Итого $\mathcal{O}(nlogn)$ на предподсчет и $\mathcal{O}(qlogn)$ на каждый запрос.

Сложность $\mathcal{O}((n+q)logn)$.