Задача A. 4B и зоопарк

Будем рассматривать людей в произвольном порядке, и каждого человека попытаемся добавить в группу. Есть 3 способа это сделать:

1. взять друга, который не состоит в группе, и образовать с ним пару;
2. найти уже сформировавшуюся пару, в которой оба человека — наши друзья, и образовать с ними тройку;
3. найти уже сформировавшуюся тройку, в которой один из людей — наш друг, и образовать с ним пару, удалив его из тройки. Таким образом от тройки останется пара.

Нетрудно видеть, что все эти действия корректные. Осталось доказать, что хотя бы одно из этих действий всегда можно будет выполнить.

Докажем от противного: пусть нашёлся человек, с которым ни одно из трёх действий нельзя выполнить. Этот человек должен не дружить ни с кем из свободных людей, ни с кем из троек, а также не должен дружить ни с какой парой целиком. Таким образом, он может дружить только с одним человеком из каждой пары, и больше ни с кем. Всего пар на данный момент не более $\frac{n-1}{2}$, а значит нашёлся человек, дружащий не более, чем с $\frac{n-1}{2}$ людьми. Но это противоречит тому, что каждый человек дружит хотя бы с $\lceil \frac{n}{2} \rceil$ людьми (по условию).

Таким образом, каждого человека можно определить в группу. Итоговое решение имеет сложность $O(N^2)$.

Задача B. Считалочка

В первый раз в ряду стоят люди с позициями $1, 2, 3, \ldots, n$. Потом остаются люди с позициями $k, 2k, 3k, \ldots$. На i-й итерации ряд состоит из людей с номерами $k^i, 2k^i, 3k^i, \ldots$.

Если на i-й итерации людей в ряду осталось меньше k, это значит, что $k^{i+1} > n$. Таким образом, номер последней итерации — это наибольшее такое число a, что $k^a \leqslant n$. Последний человек в ряду на последней итерации имеет номер $k^a \cdot b$.

Чтобы вычислить ответ, нужно максимизировать a в этой формуле с помощью цикла, затем максимизировать b по формуле $b = \lfloor \frac{n}{k^a} \rfloor$. Ответом является цифра $(k^a \cdot b - 1) \mod 10$.

Задача C. Маджонг

Чтобы проверить $Tsumo$, нужно рекурсивно или с помощью вложенных циклов перебрать все варианты взять четыре тройки и двойку. Количество таких вариантов не превосходит $14^5 \cdot 2^4$ (если явно перебирать стартовые кости и тип каждой тройки), а если не брать кости дважды, то количество можно ограничить сверху числом $12 \cdot 9 \cdot 6 \cdot 3 \cdot 2^4 = 31104$. Если хоть один из вариантов подходит, то ответ — «`Tsumo`».

Иначе, чтобы проверить $Tenpai$, нужно собрать либо пару не полностью, либо одну тройку не полностью. Это можно сделать аналогичным перебором, в котором разрешается один раз засчитать одну кость за пару или две кости за тройку, если они одинаковые или идут подряд или через одну. Если хоть один из этих вариантов подходит, то ответ — «`Tenpai`».

Задача D. Таможенные пошлины

Для каждой посылки посчитаем, какую прибыль мы можем получить, если заменим цифру у веса или у стоимости. Для этого найдём самую старшую цифру, меньшую 9, и заменим её на 9. Потом нужно решить, где замена выгоднее. Для этого посчитаем разность между измененной ценой и изначальной ценой для стоимости и веса. Логично, что выгоднее заменить цифру там, где эта разность вышла больше.

Так как нам можно сделать всего k замен, под замены нужно выбрать такие посылки, в которых искомая ранее разность была как можно больше. Для этого достаточно отсортировать посылки по убыванию этого параметра и заменить цифру в k первых посылках. Итого получим решение за $O(Nlog(N))$.

Решение могло получить WA из-за нехватки точности. На самом деле почти все вычисления в этой задаче можно производить в целых числах, а действительные использовать для подсчёта ответа. Результаты некоторых команд показали, что решение через long double также может получить AC.

Задача E. Among Us

Так как «`I`» содержится только в слове «`Impostor`», то достаточно посчитать, сколько раз встречается «`I`» в строке. Это и будет искомым числом.

Задача F. Золотые яблоки

Будем считать без ограничения общности, что $a \leqslant b$ (если $a > b$, нужно поменять их местами). Пусть Маша покупает яблоки по очереди, и после каждой покупки цены меняются в соответствии с условием. Тогда понятно, что пока цена у Васи меньше, Маша будет покупать яблоки только у него. Когда цены сравняются, Маша будет по очереди покупать яблоки то у Васи, то у Пети.

Далее нам нужно несколько формул, которые легко вывести, зная что

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}.$$

В общем случае решение выглядит так: нужно купить $b - a$ яблок у Васи, после чего купить $\lfloor (N - (b - a))/2 \rfloor$ яблок у обоих продавцов, и, возможно, купить ещё одно яблоко в конце.

Задача G. Идеальный отряд

Переберём, кто из людей выступает в качестве война (возможно, никто), кто — в качестве мага (возможно, никто), кто — в качестве клирика (возможно, никто), кто — в качестве разбойника (возможно, никто). Сделать это можно рекурсивно или с помощью четырёх вложенных циклов. После чего посчитаем количество людей, и если оно больше текущего максимума — обновим текущий максимум.

Чтобы определить, кого стоит нанять, будем поддерживать это в отдельном массиве. Если в очередной итерации цикла ответ увеличился, сбросим этот массив. Независимо от того, был ли сброшен массив, если максимум был достигнут — занесём в массив все роли, на которых в текущей итерации цикла нет человека.

Итоговая сложность алгоритма $O(N^4)$. Существует более быстрое решение с помощью алгоритма Куна.

Задача H. Полёт на ядре

Переформулируем задачу: нужно найти пару точек A, B такую, что расстояние от центра координат до соединяющей эти точки прямой минимальное.

Пусть O — центр координат. Расстояние от O до прямой AB равно отношению удвоенной площади треугольника OAB к длине отрезка AB. Если координаты точки A равны (x_1, y_1), а координаты точки B равны (x_2, y_2), то расстояние от O до прямой AB равно

$$\frac{|x_1 \cdot y_2 - y_1 \cdot x_2|}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}.$$

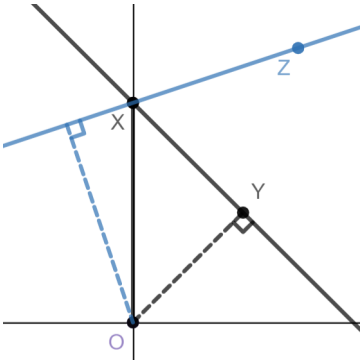
Отсортируем точки по полярному углу, который можно вычислить как $atan2(y, x)$. После чего рассмотрим все пары точек A и B такие, что выполняется одно из следующих условий:

1. точки A и B соседние в порядке сортировки (или одна из них — первая, другая — последняя),
2. угол AOB развёрнутый,
3. угол AOB положительно направленный и максимальный возможный для фиксированной A,
4. угол AOB отрицательно направленный и максимальный возможный для фиксированной A.

Рассмотреть все пары точек, удовлетворяющие первому условию, можно простым проходом по отсортированному массиву. Рассмотреть все пары точек, удовлетворяющие остальным условиям, можно проходом по массиву двумя указателями.

Для каждой пары точек посчитаем расстояние от центра координат до соединяющей их прямой и возьмём минимум. Итого получим решение за $O(Nlog(N))$.

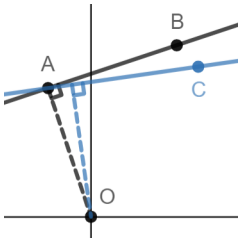
Докажем, что ответ будет найден. Подробное доказательство каждого случая упустим для краткости, но везде будет использоваться следующее утверждение: если имеются три точки X, Y и Z такие, что $min(\angle OXY, 180° - \angle OXY) < min(\angle OXZ, 180° - \angle OXZ)$, то расстояние от O до прямой XY меньше расстояния от O до прямой XZ. Это можно доказать, например, расписав площадь треугольника как произведение двух сторон на синус угла между ними.



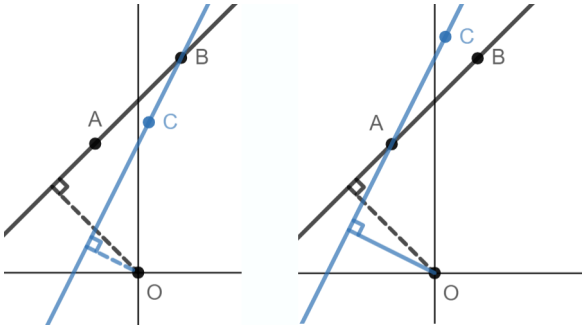
Если есть точка, совпадающая с центром координат, или две такие точки, что центр координат лежит на проходящей через них прямой, то ответ равен 0 и будет найден.

Иначе каждая точка имеет уникальный полярный угол. Пойдём от противного: пусть ближайшая к O прямая — эта прямая AB (возможно, не единственная), а алгоритм нашёл в качестве ответа более далёкую от O прямую. Далее рассмотрим углы OAB и OBA.

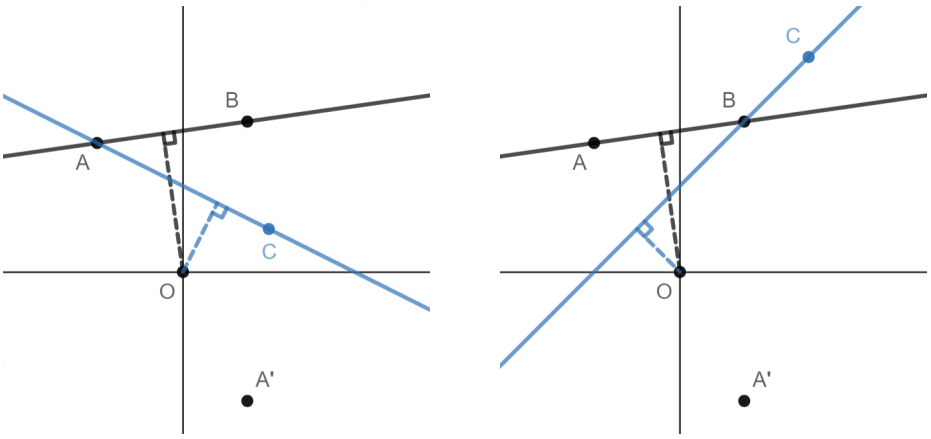
Случай 1: среди углов OAB и OBA есть прямой. Пусть без ограничения общности угол OAB прямой. Тогда алгоритм рассмотрел какую-то другую прямую AC, не совпадающую с прямой AB. Однако расстояние от O до AC меньше или равно расстояния от O до AB. Противоречие.



Случай 2: среди углов OAB и OBA есть тупой. Пусть без ограничения общности угол OAB тупой. Так как алгоритм не рассмотрел прямую AB, то есть какая-то точка C внутри угла AOB такая, что алгоритм рассмотрел прямую AC. Тогда C не может лежать на прямой AB. Если C находится внутри треугольника AOB, то расстояние от O до BC меньше расстояния от O до AB. Если C находится снаружи треугольника AOB, то расстояние от O до AC меньше расстояния от O до AB. В любом случае получаем противоречие.



Случай 3: углы OAB и OBA оба острые. Пусть A' — точка, симметричная A относительно O. Так как алгоритм не рассмотрел прямую AB, то есть какая-то точка C внутри угла $A'OB$ такая, что алгоритм рассмотрел прямую AC. Тогда C не может лежать на прямой AB. Если C и O лежат по одну сторону от прямой AB, то расстояние от O до AC меньше расстояния от O до AB. Если C и O лежат по разные стороны от прямой AB, то расстояние от O до BC меньше расстояния от O до AB. В любом случае получаем противоречие.



Итак, во всех трёх случаях либо существует прямая AC или BC, более близкая к центру координат, чем AB; либо алгоритм рассматривал прямую AC, находящуюся на том же расстоянии до центра координат. Это противоречит тому, что AB — самая близкая к центру координат прямая, и прямая на таком расстоянии не была найдена.

Задача I. Квадрокоптер

С помощью алгоритма Дейкстры найдём кратчайшее расстояние до каждой клетки. После чего будем симулировать движение квадрокоптера. Есть две возможные траектории:

1. квадрокоптер улетает с поля,
2. квадрокоптер попадает в цикл.

В первом случае нужно найти самую первую клетку, посещённую квадрокоптером (в порядке посещения), такую, что Вова попал туда не позже квадрокоптера. Время посещения квадрокоптером этой клетки и будет ответом. Если такой клетки нет, то ответ равен -1.

Во втором случае путь квадрокоптера делится на предпериод и период. Предпериод — это все клетки, которые квадрокоптер посетит один раз, период — все клетки, которые квадрокоптер посетит бесконечное количество раз. Если есть клетка, в которую Вова попал не позже квадрокоптера, то ответ — время первого посещения квадрокоптером такой клетки. Иначе ответ все равно существует.

Вычислим длину периода, а также для каждой клетка запомним время первого посещения её квадрокоптером. Пусть длина периода равна t, время первого посещения клетки равна s_{ij}, а кратчайшее расстояние от стартовой позиции Вовы до клетки равно d_{ij}. Если клетка принадлежит периоду, то квадрокоптер посетит эту клетку в моменты времени $s_{ij}, s_{ij}+t, s_{ij}+2t, \ldots$. Нужно найти минимальное такое x_{ij}, что

$$s_{ij} + x_{ij} \cdot t \geqslant d_{ij}.$$

Это равносильно

$$x_{ij} \geqslant \frac{d_{ij} - s_{ij}}{t},$$

откуда

$$x_{ij} = max(0, \lceil \frac{d_{ij} - s_{ij}}{t} \rceil).$$

Минимум по значениям $s_{ij} + x_{ij} \cdot t$ для всех клеток в периоде и будет ответом в этом случае.

Итоговая сложность алгоритма $O(wh \cdot log(wh))$.

Задача J. Интересные разговоры

Заметим, что

$$(a_1 - a_2) + (a_2 - a_3) + \ldots + (a_n - a_1) = 0.$$

Видно, что сумма положительных и отрицательных скобок равны по абсолютному значению. Из этого получаем следующее разбиение: все разговоры, в которых $a_i - a_{i+1} \geqslant 0$, кладём в первое множество, а остальные — во второе. Отдельно стоит рассмотреть случай, когда все a_i равны. В таком случае подойдет любое разбиение на два непустых множества.

Задача K. Вася и прогулка по бесконечной дороге

Можно выписать для i-го интересного столба следующее равенство: $k_i = ai + b$, где k_i — номер i-го интересного столба, a — количество столбов, мимо которых Вася проходит за час, b — номер столба, перед которым Вася начинает свою прогулку.

Параметры a и b неизвестны, но можно заметить, что в примере ввода/вывода даны k_{2019} и k_{2020}. Затем нужно выписать для каждого из них вышеупомянутое равенство, составить по двум полученным равенствам систему уравнений и решить её относительно a и b. Эта система уравнений изображена ниже:

$$\begin{cases} k_{2019} = a \cdot 2019 + b \\ k_{2020} = a \cdot 2020 + b \end{cases}$$

Задача L. Прыжок с парашютом

При движении с горизонтальной скоростью 1 м/с и ускорением свободного падения 10 м/с^2 траектория задаётся уравнением $y = -5x^2$. Таким образом, начать из точки (x_1, y_1) и попасть в точку (x_2, y_2) можно только если $y_1 - y_2 \geqslant 5 \cdot (x_1 - x_2)^2$, и это займёт $\sqrt{\frac{y_1 - y_2}{5}}$ секунд.

Решение за $O(N^2)$ времени: отсортируем точки по убыванию координаты y (точки — это все облака, а также стартовая позиция), после чего напишем динамическое программирование в этом порядке. Для каждой точки нужно посчитать наибольшее время, за которое мы можем в неё попасть. Для стартовой позиции это время равно 0. Для каждого облака рассмотрим все точки выше него, из которых можно в него попасть, и пересчитаем таким образом значение в этом облаке.

Чтобы ускорить решение, заметим, что если максимальное возможное значение y равно $MAXY$, то в точку (x_i, y_i) можно попасть только из таких точек (x, y), что $|x - x_i| \leqslant \sqrt{\frac{MAXY}{5}}$. В этой задаче $MAXY = 10^4$, следовательно должно выполняться $|x - x_i| \leqslant 44$.

Следующее замечание: если из точки A можно попасть в точку B, и из них обеих можно попасть в точку C, то пересчитывать значение динамического программирования из A в C не имеет смысла. В самом деле, плотности положительны, следовательно путь из A в C через B будет всегда дольше пути из A в C напрямую. Таким образом, чтобы пересчитать значение динамического программирования в точке C, нужно для каждой подходящей координаты x найти самую нижнюю точку B, из которой C достижима, и пересчитать ответ только из B. То есть, в каждом столбце нас интересует не более одной точки.

Итак, решение выглядит следующим образом: для каждой координаты x выпишем все точки с такой координатой x, упорядочив их по координате y. Будем рассматривать точки в порядке убывания координаты y, для каждой точки нужно посчитать наибольшее время, за которое можно в неё попасть. Для стартовой позиции это время равно 0. Для каждого облака с координатами (x_i, y_i) рассмотрим все координаты x от $x_i - 44$ до $x_i + 44$ включительно, для каждой координаты найдём самую нижнюю точки (x, y) такую, что $y - y_i \geqslant 5 \cdot (x - x_i)^2$. Это можно сделать с помощью бинарного поиска, так как в каждом столбце у нас уже выписаны все точки (их не более $min(N, MAXY)$). У нас получится не более 89 точек, из каждой из них пересчитаем динамическое программирование. Решение работает за $O(N \cdot \sqrt{MAXY} \cdot log(min(N, MAXY)))$.

К сожалению, жюри не умеет сдавать эту задачу на Python.