Задача A. Нехолодный кофе

Заметим, что если $22 \leqslant Q$, то кофе всегда будет пригодным для питья Аязу, и ответ в таком случае — 1". Иначе кофе остывает на 1 градус каждые x секунд, и тогда он остынет до непригодной температуры за $(100-Q+1)*x$ секунд, то есть будет пригодным для питья на одну секунду меньше — $(100 - Q + 1) * x - 1$.

Задача B. Место команды

Для начала отсортируем команды по не возрастанию баллов, а при равенстве баллов — по названию. Если $k = 1$ или k-я и $(k-1)$-я команда набрали разное количество баллов, то существует хотя бы одна команда, которая заняла k-е место. Тогда нужно выводить названия всех команд, начиная с k-й, до тех пор, пока мы не дойдем до конца либо не встретим команду, которая набрала меньше баллов.

При таком решении, для прохождения первой подзадачи достаточно написать любую квадратичную сортировку $O(n^2)$, а для решения на полный балл можно воспользоваться быстрой сортировкой или сортировкой слиянием. Такое решение будет работать за $O(n \log(n))$.

Задача C. Оригинальные сайты

1. Первая группа ($O(n^2)$):

Используем наивный подход. Для каждого сайта из второй части каждой из n пар (b_i) проверяем, не встречается ли он в первой части (среди всех a_j) и не учитывался ли он уже в ответе (в уже отсмотренных значениях b_i). Это делается двумя вложенными циклами, что приводит к общей сложности $O(n^2)$. Этот метод подойдет для небольших значений n.

2. Вторая группа ($O(n)$):

В данном случае предполагается, что существует ограничение на размер чисел (например, U). Используем массив размером U, чтобы отметить все сайты a_i, которые заимствуют. Проходим по всем парам, отмечаем a_i. Затем для каждого b_i просто проверяем, не стоит ли отмеченный флаг. Здесь так же нельзя забывать, что нам нужно посчитать уникальные значения b_i, так что флаг одновременно стоит ставить и для b_i. Это даст сложность $O(n)$ с дополнительной памятью $O(U)$.

3. Третья группа ($O(n)$ в среднем):

Здесь используем хэш-таблицу, в которую добавляем все сайты a_i, так как числа могут быть большими. Затем для каждого b_i проверяем наличие в хэш-таблице. Здесь так же нельзя забывать, что нам нужно посчитать уникальные значения b_i, так что их тоже нужно учитывать в хэш-таблице по мере прохода. Итоговая сложность в среднем $O(n)$, благодаря постоянному времени вставки и поиска в хэш-таблице.

Задача D. Один раз отрежь, семь раз отмерь

Подзадача 1

Чтобы решить данную подзадачу, давайте переберем, какую точку второго многоугольника нужно передвинуть в первую точку первого многоугольника. Выбрав данную точку, мы можем вычислить, на сколько ее нужно сместить по x и y. После чего нужно проверить, совпадут ли все остальные точки второго многоугольника с соответствующими точками первого многоугольника, если каждую из них сдвинуть на то же расстояние по x и y. Итоговая сложность — $O(n^2)$.

Подзадача 2

В отличие от предыдущей подзадачи, в этой существует 4 различных варианта поворота. Мы можем перебрать все 4 варианта, после чего применить решение из *подзадачи 1*. Для удобства легче всего поворачивать многоугольник относительно точки $(0, 0)$. При таком повороте точка с координатами (x, y) перейдет в точку $(y, -x)$ (если поворачивать по часовой стрелке). Сложность алгоритма поворота многоугольника — $O(n)$, а итоговая сложность — $O(n^2)$.

Подзадача 3

Чтобы полностью решить данную подзадачу, нам нужно решить *подзадачу 2* для изначального многоугольника, после чего мы можем отразить его относительно оси X. И еще раз решить *подзадачу 2* для получившегося многоугольника. Итоговая сложность — $O(n^2)$.

Подзадача 4

Самое узкое место — это решение *подзадачи 1*. Давайте научимся решать ее быстрее.

Для начала найдем самую левую точку первого многоугольника (если их несколько, то самую нижнюю из них), тоже сделаем со вторым многоугольником. Пусть это будет i-я точка первого многоугольника и j-я — второго. Тогда для любого k $(0 \leqslant k \leqslant n-1)$ нам нужно сравнивать точку с индексом $(i+k)\ mod\ n$ первого многоугольника с точкой $(j+k)\ mod\ n$ — второго. Итоговая сложность — $O(n)$.

Задача E. Ночь перед олимпиадой

Подзадачи 1 и 2

Построим граф, где люди будут вершинами, некоторые из которых будут соединены взвешенными ребрами.

Заметим, что ответом будет -1 тогда и только тогда, когда существует вершина, до которой мы не можем добраться из вершины 1.

Пусть x — сложность задачи, которую решит Данил. Исходя из ограничений задачи, нам достаточно перебрать x от 1 до 100 (и отдельно проверить $x = 10^9$). Для отдельного x мы можем построить новый граф на основе исходного следующим образом: если вес ребра меньше x, то в новом графе вес ребра будет равен 1, иначе — 0. В получившемся графе нам нужно найти кратчайшее расстояние до всех вершин. Если расстояние до самой дальней вершины будет не более k, то данный x нам подходит.

В зависимости от того, каким алгоритмом мы будем находить кратчайшие пути — мы решим одну или обе подзадачи. Для нахождения кратчайших путей можно использовать один из следующих алгоритмов: алгоритм Дейкстры для полных графов $O(n^2)$, алгоритм Дейкстры для полных разреженных $O(n^2)$, 0-1bfs $O(n+m)$.Итоговая сложность — $O((n+m)h)$, где $h = 100$.

Подзадача 3

Заметим, что если мы можем взять в качестве задачи со сложностью x, то можем и взять задачу со сложностью $x-1$. Поэтому воспользуемся бинпоиском, чтобы найти такое максимальное x. Итоговая сложность — $O((n+m)\log(H))$, где $H = 10^9$.

Задача F. Спортивные знакомства

1. Учеников, которые любят только футбол: это $b - c$, где b — общее число футболистов, а c — число тех, кто занимается обоими видами спорта.

2. Учеников, которые любят только баскетбол: это $a - c$, где a — общее число баскетболистов.

3. Учеников, не занимающихся никаким спортом: это $n - (a + b - c)$, где n — общее количество учеников. Здесь $(a + b - c)$ учитывает всех, кто занимается хотя бы одним видом спорта.

4. Ответ: количество учеников, не занимающихся спортом, плюс максимум среди любящих только футбол и только баскетбол, плюс 1:

$$\text{Ответ} = (n - (a + b - c)) + \max(b - c, a - c) + 1$$

$\max(b - c, a - c)$ обеспечивает максимальное число из тех, кто занимается только одним видом спорта. Это гарантия того, что Вася в худшем случае обязательно встретит всех людей, играющих только в 1 вид спорта.

Дополнение на единицу: "+1"гарантирует, что, даже при максимизации встреч с одной из групп, Вася познакомится как минимум с одним представителем футбольной или баскетбольной группы.

Задача G. Коробка в коробке

Подзадача 1

Пусть мы рассматриваем коробку с размерами x, y, z $(1 \leqslant x \leqslant a, 1 \leqslant y \leqslant b, 1 \leqslant z \leqslant c)$. Заметим, что по условию задачи $z = x+y$, поэтому нам достаточно перебрать все возможные x и y и проверить, что $x + y \leqslant c$ и $x \cdot y \cdot (x + y) \geqslant v$. Такое решение имеет сложность $O(a \cdot b)$.

Подзадача 2

Давайте решим задачу для некоторого фиксированного x. Заметим, что если для некоторого y объем коробки будет не меньше v, то это будет верно и для больших y. Пусть мы нашли минимальный такой y. Тогда мы можем прибавить к ответу $\min(b - y + 1, c - x - y + 1)$. Можно доказать, что

для вычисления ответа необходимо перебрать не более $\sum_{x=1}^{a} \min(b, \frac{v}{x^2})$ пар (x, y), что будет равно $O(b \cdot v^{1/4} + v^{3/4})$.

Подзадача 3

Чтобы получить полный балл, будем действовать как и в предыдущей подзадаче, но для поиска минимального y, для которого $x \cdot y \cdot (x + y) \geqslant v$, воспользуемся бинарным поиском. Такое решение имеет сложность $O(a \log b)$.

Также заметим, что задачу можно решить методом двух указателей, т.к. y не может увеличиваться при увеличении x.

Задача H. Большое число

Это задача на реализацию. Каждую из цифр можно записать в программе в виде массива 5×3 (либо массива из 5-и строк).

Подзадача 1

В данной подзадаче можно было перебрать все 10 цифр для каждой из позиций. Аккуратно нужно было рассмотреть случай с лидирующими нулями.

Подзадача 2

В данной подзадаче нужно перебирать k, для этого нужно менять местами строки в таблице, либо в таблицах для каждой цифры.

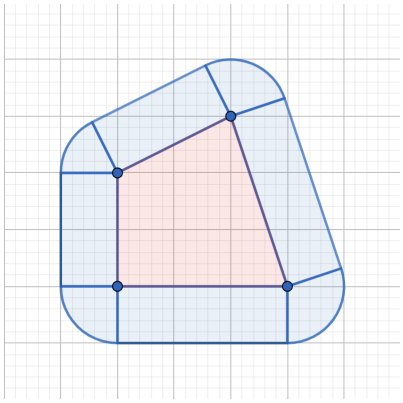
Подзадача 3

Чтобы получить полный балл, нужно реализовать все тоже самое, что и во второй подзадаче, но сделать это наиболее оптимально. Сделать это можно многими способами, вот некоторые из них:

- Можно заметить, что для любой цифры кроме 1 можно восстановить k однозначно. Поэтому можно найти любую цифру, которая не равна 1, и узнать k, после чего восстановить ответ.
- Можно сравнивать цифры более оптимально, либо некоторым образом захешировать цифры и сравнивать хеши.

Задача I. Империя хомяков

Империя через некоторое время r будет выглядеть следующим образом



красный многоугольник — начальные границы империи, а приобретенные территории можно разбить на несколько прямоугольников и секторов. Тогда суммарную площадь можно найти по следующей формуле

$$S = S_0 + \pi \cdot r^2 \ + L \cdot r$$

где S_0 — изначальная площадь империи, L — периметр многоугольника. Откуда можно найти r бинарным поиском или решив квадратное уравнение. Отдельно нужно рассмотреть случай, если изначальная площадь будет уже не меньше необходимой в запросе, тогда ответ на такой запрос будет 0.

Чтобы получить полный балл необходимо заранее подсчитать S_0 и L. Тогда отвечать на запрос можно за $O(1)$. Итоговая сложность $O(n + q)$.

Задача J. Один цвет — ребра нет

Мысленно уберем ребра, связывающие две вершины дерева разного цвета. Остались компоненты связности вершин одного цвета. Заметим, что ответ для исходной задачи эквивалентен ответу текущей, так как для ответа нужны вершины лишь одного цвета, поэтому удаленные ребра не повлияют на него. Предположим, что мы посчитали размер максимального независимого множества вершин (МНМВ) для каждой из компонент, тогда заметим, что размер МНМВ определенного цвета c будет равен сумме размеров МНМВ каждой компоненты цвета c. Далее решим задачу при помощи динамического программирования по поддеревьям. Заведем двумерный массив $dp_{v,j}.v = 1 \dots n, j = 0 \dots 1, dp_{v,0}$ - максимальное количество вершин в поддереве v, которое можно взять, если не брать вершину v, и $dp_{v,1}$ - макс. кол-во вершин в поддереве v, если возьмем вершину v. Подвесим дерево за первую вершину, запустим из нее обход в глубину (DFS), пусть мы в вершине v, запустим сначала DFS от ее детей, а потом пересчитаем для нее $dp_{v,0} = \sum max(dp_{u,0}, dp_{u,1}), (u,v) \in E, u$ и v одного цвета. $dp_{v,1} = (\sum dp_{u,0}) + 1, (u,v) \in E, u$ и v одного цвета. Если мы в первой вершине или родитель v имеет другой цвет, то в заранее заведённый массив cnt_c, где c - цвет текущей вершины, добавим $max(dp_{v,0}, dp_{v,1})$. Ответ - это максимальный cnt_c по всем цветам c.

Задача K. Юные блогеры

Для решения задачи нужно научится сравнивать друзей по крутости: пусть $a1$ и $s1$ — возраст и количество подписчиков первого друга, а $a2$ и $s2$ — второго. Тогда первый друг круче второго, если выполняется условие: $\frac{s1}{a1} > \frac{s2}{a2}$ or $\frac{s1}{a1} == \frac{s2}{a2}$ and $a1 < a2$. Зная это, можно выявить самого крутого друга и определить его возраст. Для этого можно выбрать любого друга в качестве текущего лидера, а затем пройтись по списку всех друзей, сравнивая каждого с текущим лидером и обновляя его, если найдется более крутой друг.

Задача L. МаксИмальная сумма

Заметим, что побитовый «И» двух положительных целых чисел не более, чем минимальное из двух операндов. Поэтому максимальный побитовый «И» двух положительных целых чисел с суммой n не более, чем $\frac{n}{2}$. Для четного n ответом будет пара $a = \frac{n}{2}, b = \frac{n}{2}$. Рассмотрим битовое представление нечётного числа n, оно будет иметь вид $M011 \dots 1$, где M - это маска на префиксе. Заметим, что как бы мы не распределили суффикс, состоящий только из единиц, по двум маскам ответа, чтобы сумма была равна n, максимальное побитовое «И» будет содержать только нули на этом суффиксе. Учитывая этот факт, в качестве ответа для нечётного n подходит пара $a = \lfloor \frac{n}{2} \rfloor$, $b = n - a$.

Задача M. Делители

Подзадача 1

Мы можем перебрать все числа от a до b и для каждого подсчитать количество делителей. Подсчитать количество делителей для некоторого числа n можно за $O(\sqrt{n})$. Сложность такого решения — $O(b\sqrt{b})$.

Подзадача 2

Количество делителей можно найти по формуле

$$d(n) = (a_1 + 1)(a_2 + 1) \dots$$

где a_1, a_2, ... — степени в разложении данного числа на простые множители.

Давайте пред подсчитаем для всех положительных чисел, не превосходящих b, наименьший простой делитель. Сделать это можно с помощью решета Эратосфена. После этого мы можем быстро факторизовать любое число из отрезка от a до b и подсчитать количество делителей. Сложность такого решения — $O(b \log b)$

Задача N. Чудеса света

Давайте научимся считать какое множество чудес света можно увидеть начиная путешествие с заданной координатой Y.

Чудо света под номером i в таком случае обозримо, тогда и только тогда, если оно видно из координаты (x_i, Y) в рамках прогулки.

Если заметить этот факт и переформулировать условие на видимость чуда света, то мы поймем, что оно видно, если не найдется другого чуда света с такой же координатой $x_j = x_i$ и координатой y_j лежащей между y_i и Y.

Опираясь на вышеописанные рассуждения разобьем все чудеса света в блоки по координатам x.

В рамках одного блока проходя между двумя чудесами света мы видим оба из них, проходя же перед первым или после последнего только одно.

Теперь задача свелась к задаче на отрезки и нам нужно найти минимальную точку Y покрытую большинством отрезков(отрезки строятся по блокам с равными x).

Теперь это задача свелась к сканлайну и решаема за $O(n \cdot log(n))$.

Задача O. Обнаружить заклинание

Начнем с жадного алгоритма проверки вхождения строки b внутрь другой строки a как подпоследовательности: Поддерживаем индекс текущего символа (начинаем с первого (нулевого)) строки b, пока идем по строке a слева направо. Когда встречаем символ, равный символу в строке b по текущему индексу, увеличиваем этот индекс. Таким образом, мы жадно набираем префикс строки b, входящий в строку a как подпоследовательность. Если в конце этот префикс равен длине всей строки b, - она полностью входит в a как подпоследовательность.

Теперь будем так же жадно набирать префикс строки b и для каждого префикса строки a запоминать длину этого префикса (входящего в этот префикс строки a как подпоследовательность). Аналогично для каждого суффикса a посчитаем длину набранного внутри как подпоследовательности суффикса b - для этого достаточно просто в обоих строках идти справа налево.

Теперь переберем позицию pos строки a и (с помощью уже посчитанных для каждой позиции величин) возьмем длину префикса b, входящего внутрь префикса a, заканчивающегося в pos, и длину суффикса b внутри суффикса a, начинающегося в pos + 1. Если сумма этих величин больше или равна длине b - 1, это значит, что мы его нашли с учетом одного пропущенного символа.