

Задача А. Середина игры

Заметим, что за любой результат в игре суммарно даётся 2 очка, значит и сумма очков в любой момент должна быть чётной. Если $A + B$ нечётное, то ответ «Error».

Если $A \geq 2$ и $B \geq 2$, то определить однозначно невозможно: пусть $M = \min(A, B)$, невозможно понять, выиграл ли каждый по M раз или же было $2 \cdot M$ ничьих.

Для остальных случаев ответ $\lfloor \frac{A}{2} \rfloor, \lfloor \frac{B}{2} \rfloor, A \bmod 2$.

Задача В. Ужин для интровертов

Оценка. Пусть L_i, R_i — количество свободных мест слева и справа для i -го интроверта. Тогда по условию $L_i + R_i \geq K$. Пусть людей за столом X , просуммируем по всем людям. $\sum L_i = \sum R_i = N - X$. Тогда $2 \cdot N - 2 \cdot X \geq X \cdot K, 2 \cdot N \geq X \cdot (K + 2), X \leq \frac{2 \cdot N}{K + 2}$.

Пример. Будем для заданного X строить пример на минимальное N , полученное из оценки сверху. Заметим, что если мы добьемся того, что в примере сумма каждого $L_i + R_i = K$, то и в этом примере само собой станет минимальным подходящим под оценку, так как все неравенства станут равенствами.

Если $K = 2 \cdot y$, то между каждыми двумя соседями оставляем по y мест. Если $K = 2 \cdot y + 1$, то чередуются y и $y + 1$ пустых мест, в случае нечетности X получится рядом два отрезка по $y + 1$ мест, что даст дополнительную единичку в сумму всех L_i и R_i , что учтется формулой сверху, так как это единственный случай, когда $X \cdot (K + 2)$ будет нечетным, а с другой стороны неравенства стоит $2 \cdot N$ — четное число.

Задача С. Плохие ставки

Если игрок ставит на сумму $S < N$ или $S > N \cdot K$, то шансов победить у него нет. Разберём остальные случаи. Если $N = 1$, тогда любые две ставки равновероятны. Если $N > 1$, тогда нужно посчитать матожидание $\frac{N \cdot (K + 1)}{2}$, чем дальше S от него (то есть значение $|S - \frac{N \cdot (K + 1)}{2}|$ больше), тем меньше шансов на победу. Достаточно сравнить посчитанные значения.

Задача D. Ключало

Если менять i -ю деталь, то можно уменьшить отклонение на $\frac{1}{s_i}$. Поэтому вначале лучше всего изменять детали, которые в стандарте весят меньше всего. Тогда можно изменять детали в порядке убывания. Если текущее отклонение $K_c - \frac{|a_i - s_i|}{s_i} > K$, то деталь можно привести к стандарту напрямую, затем уменьшить K_c . Если же $K_c - \frac{|a_i - s_i|}{s_i} \leq K$, то нужно решить систему уравнений $K_c - \frac{x}{s_i} \leq K$ и $K_c - \frac{x - 1}{s_i} > K$. Решением этой системы является $x = \lceil \frac{K_c - K}{s_i} \rceil$.

Чтобы избежать проблем с точностью, все вычисления следует проводить в дробном виде.

Задача Е. Маска для монстров

Очевидно, что кратчайшая маска должна идти по границе многоугольника, причём от одной вершины по кругу до другой. Тогда ответом будет разность периметра и длины наибольшей стороны.

Задача F. Наибольший наибольший общий делитель

Пусть G — искомый НОД. Поскольку на отрезке есть хотя бы два числа, делящихся на G , то $G \leq R - L$. Переберём все G от $R - L$ до 1 и для каждого смотрим, есть ли хотя бы два числа делящихся на него. Это можно сделать простой проверкой $L \leq \lfloor \frac{R}{G} \rfloor \cdot G - G$. Если это условие выполняется, то два искомого числа равны $\lfloor \frac{R}{G} \rfloor \cdot G - G$ и $\lfloor \frac{R}{G} \rfloor \cdot G$.

Задача G. Прогрессивный NoSQL

Заведём множество уже выданных имён и словарь, в котором для каждого введённого имени будет храниться предыдущий дописанный числовой суффикс. Когда приходит новый запрос, если его нет в множестве, добавляем это имя в множество, а в словаре записываем 0. Если же запрос есть в множестве, то пытаемся увеличить предыдущий дописанный числовой суффикс до тех пор, пока не найдётся имя, которого нет в множестве, после этого обновляем этот числовой суффикс.

Нетрудно заметить, что этот алгоритм будет работать за $O(Q \cdot \log_{10} Q)$.

Задача Н. Марго покидает Мегабайтбург

Создадим граф, в котором вершинами будут пустые клетки, рёбрами с весом 0 — ходы из пустой клетки (i, j) в пустые клетки из $(i - 1, j)$, $(i + 1, j)$, $(i, j - 1)$, $(i, j + 1)$, а также рёбра с весом 1 — ходы из пустой клетки (i, j) в пустые клетки из $(i - 2, j)$, $(i + 2, j)$, $(i, j - 2)$, $(i, j + 2)$. В таком графе можно запустить 0-1 BFS, который посчитает кратчайшее расстояние от общежития до аэропорта, это расстояние означает минимальное количество прыжков, которое понадобится, чтобы добраться от общежития до аэропорта. Это расстояние надо сравнить с K , и вывести соответствующий ответ.

Задача I. Snakes&Snakes

Вначале посчитаем, куда ведёт каждый телепорт (чтобы избежать цепочек телепортов). Для этого заведём массив e_i , в котором если $p_i = 0$, то $e_i = i$, иначе $e_i = e_{i-p_i}$. Если заполнять этот массив в порядке ввода, то все предыдущие значения будут посчитаны.

Заведём массив d_i — минимальное количество ходов, чтобы добраться до i -й клетки. Изначально $d_0 = 0$, остальные $d_i = +\infty$. Будем обрабатывать клетки без телепорта в порядке ввода, если клетка была обработана раньше, то мы её пропускаем, после этого обновляем $d_{e_{i+x}} = \min(d_{e_{i+x}}, d_i + 1)$, где x — числа от 1 до 6, помечаем клетку, как обработанную. Затем обрабатываем клетку с номером e_{i+6} , но только если она не была уже обработана, причём $d_{e_{i+6+x}}$ будет обновляться также значением $d_i + 1$. Будем продолжать обрабатывать такие клетки до тех пор, пока не встретим уже обработанную, после этого переместимся в следующую клетку в порядке ввода.

Можно легко понять, что поскольку телепорты всегда отправляют назад, то на каждом шаге расстояния будут посчитаны только для префикса, причём если $i < j$, то $d_i \leq d_j$. Это означает корректность алгоритма, который будет работать за $O(N)$.

Задача J. Бинарные деревья

Заметим, что операция переподвешивания обратимая, поэтому сведём оба дерева к бамбуку. Докажем, что одно дерево можно свести к бамбуку за не более чем $\frac{N-1}{2}$ операций. Покажем это по индукции — для дерева из одной вершины это можно сделать за $0 \leq \frac{1-1}{2}$ операций. Пусть теперь дерево размера k , если у корня один сын, то решаем для его поддерева за $\frac{k-2}{2} < \frac{k-1}{2}$. Если два сына — пусть размеры их поддеревьев a и b , тогда $k = a + b + 1$. Сводим каждое поддерево к бамбуку за $\frac{a-1}{2}$ и $\frac{b-1}{2}$ операций соответственно, затем переподвешиваем второе поддерево к самой нижней вершине первого. Всего выйдет операций $\frac{a-1+b-1}{2} + 1 = \frac{a+b}{2} = \frac{k-1}{2}$.

Задача K. Апокалипсис

Заметим, что заражение распространяется на одинаковое расстояние от изначального многоугольника, и его площадь в i -й день равна $S_i = 2^i \cdot S_0$. Пусть расстояние, на которое распространилось заражение в i -й день, равно d_i . Тогда $S_i = S_0 + P_0 \cdot d_i + \pi d_i^2$, где P_0 — периметр изначального многоугольника. Это квадратное уравнение, найдя решения для которого, можно найти d_i для каждого из дней. Нетрудно заметить, что при данных ограничениях на входные данные $d_{70} > 10^{10}$, это означает, что любое из поселений будет заражено в первые 70 дней, поэтому достаточно быстро посчитать расстояние D_j от каждого поселения до изначального многоугольника и найти первый день i , когда $d_i \geq D_j$.

Чтобы быстро считать расстояние от точки до выпуклого многоугольника достаточно сделать несколько шагов:

1. Проверить, лежит ли точка внутри многоугольника за $O(\log N)$, если лежит внутри, то расстояние равно 0, иначе выполнить следующие шаги;
2. Построить касательные из точки к многоугольнику за $O(\log N)$, тем самым найдутся две вершины многоугольника с индексами l и r ;
3. Многоугольник разбивается на две части найденными вершинами. Поскольку многоугольник выпуклый, то в одной из частей расстояния до точки будут представлять из себя выпуклую функцию, по которой можно запустить тернарный поиск минимального значения. На самом деле, достаточно найти ближайшую вершину, а затем выбрать минимальное расстояние до смежных с этой вершиной отрезков-сторон. Этот шаг алгоритма работает также за $O(\log N)$.

Таким образом, алгоритм нахождения подходящего дня для каждого поселения работает за $O(\log N)$, а всё решение будет работать за $O(Q \cdot \log N)$.