Задача A. 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 $A \geqslant 2$ и $B \geqslant 2$, 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 $\lfloor \frac{A}{2} \rfloor, \lfloor \frac{B}{2} \rfloor, A \mod 2$.

Задача B. Ужин для интровертов

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

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 n 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 C. 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. Клюкало

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 неубывания. 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 $K_c - \frac{|a_i - s_i|}{s_i} \leqslant K$, то нужно решить систему уравнений $K_c - \frac{x}{s_i} \leqslant K$ 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 E. 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 $G \leqslant R - L$. 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 $L \leqslant \lfloor \frac{R}{G} \rfloor \cdot G - G$. Если это условие выполняется, то два искомых 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 хранится 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 H. 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.