Задача A. Построить башни

Заметим, что минимальная стоимость ханойской башни равна $B + R \cdot K$. Это означает, что Вадим сможет построить не больше, чем $\lfloor \frac{N}{B+R \cdot K} \rfloor$ башен. Если это количество больше нуля, то Вадим сможет потратить оставшиеся деньги на кольца для одной башни, чтобы увеличить её высоту. Таким образом, высота наибольшей башни будет равна $K + \lfloor \frac{N \mod (B+R \cdot K)}{R} \rfloor$.

Задача B. Пододеяльник

Ответом будет прямоугольник $[(X, Y), (-X, Y), (-X, -Y), (X, -Y)]$. Его периметр будет равен $4 \cdot (X + Y)$.

Докажем, что никакой другой выпуклый многоугольник не сможет улучшить данную оценку. Пусть есть выпуклый многоугольник на N вершинах, разберём каждую из его сторон. Спроецируем текущую сторону на ось Ox и на ось Oy. По неравенству треугольника сумма длин этих проекций не меньше длины исходной стороны. Значит, сумма длин всех сторон многоугольника не больше, чем $\max_{i \in [1..N]}(x_i) - \min_{i \in [1..N]}(x_i) + \max_{i \in [1..N]}(y_i) - \min_{i \in [1..N]}(y_i) \leqslant 4 \cdot (X + Y)$. Значит, никакой другой выпуклый многоугольник не будет иметь периметр больший, чем $4 \cdot (X + Y)$.

Задача C. Завали программу Егора / Опять переполнение / Очень длинный int

Рассмотрим 2 случая. При $T = 0$ в двоичном разложении K ищем вторую справа 1. Ясно, что если таких нет, то K - степень двойки и ответ = 1, а если такая есть, то необходимо и достаточно умножить K на минимальную такою степень 2, что эта 1 уйдёт за пределы числа и ответ = N − индекс этой единицы + 1, считая индекс единицы с 0. При $T = 1$ в двоичном разложении K ищем первую справа 1, правее которой есть 0. Ясно, что если таких нет, то K представимо как $2^N - 1$ и ответ = 1, а если такая есть, то необходимо и достаточно сделать с K столько операций, что эта 1 уйдёт за пределы числа, и ответ = N − индекс этой единицы + 1, считая индекс единицы с 0.

Задача D. Сергей и базы данных

Давайте заметим, что отрицательные суммы считать не выгодно и поэтому мы будем считать только положительные суммы. Также заведём ассоциативный массив и будем по ключу a_i добавлять b_i. Затем переведём наш ассоциативный массив в массив пар и отсортируем монотонно убывающие по значению. Далее разберём два случая:

1. Когда положительных элементов больше или равно K, тогда выводим все K первых элементов;
2. Когда положительных элементов меньше или равно K, тогда выводим все первые попавшиеся положительные, а затем выводим несуществующий a_i, необязательно попадающий в диапазон $[0, 10^5]$, например, 10^9.

Тогда конечная асимптотика будет равна $O(N \cdot \log N)$.

Задача E. Марго делает домашнее задание

Для начала остортируем исходный массив. Тогда ответ на вопрос первого типа можно получать при помощи бинарного поиска. При этом делать честное прибавление какого-то числа X ко всем элементам массива будет слишком долго. Предположим, что мы сделали несколько прибавлений + X_1, + X_2, ..., + X_k, и теперь нам нужно ответить на некоторый вопрос вида ! Y. В таком случае каждый элемент массива увеличился на $diff = X_1 + X_2 + ... + X_k$. Тогда вопрос ! Y для текущей версии массива эквивалентен вопросу ! $(Y - diff)$ для изначального массива. Благодаря этому мы можем не изменять массив после сортировки, а только обновлять значение $diff$, когда получаем вопрос второго типа.

Задача F. Сбалансированная олимпиада

Переберём все подотрезки данного массива. Зафиксируем левую границу и будем увеличивать правую. Также заведём два массива — cnt_i означает количество задач сложности i на текущем подотрезке; eq_i означает количество сложностей задач, количество которых равно i. При добавлении

новой задачи со сложностью A_r в подотрезок мы уменьшаем $eq_{cnt_{A_i}}$ на 1, прибавляем к cnt_{A_i} единицу, затем увеличиваем $eq_{cnt_{A_i}}$ на 1. После этого делаем проверку, что $cnt_{A_i} \cdot eq_{cnt_{A_i}} = N$, это означает, что набор на подотрезке от левой границы до текущей правой границы является сбалансированным. Из всех таких нужно выбрать наибольший по длине. Это решение работает за $O(N^2)$.

Задача G. Сломанная программа 2

Посчитаем для каждой клетки и для каждого направления из неё расстояние до ближайшей стены. Это можно сделать стандартным алгоритмом за $O(N \times M)$ для каждого из направлений (например, для движения вверх, если сверху в соседней клетке находится стена, то расстояние равно нулю, иначе оно равно сумме расстояния до стены для клетки сверху и единицы). Посчитав это, можно быстро понять, в какой клетке окажется робот после каждой из команд: если это расстояние больше или равно, чем данное количество ячеек в данной команде, то робот просто переместится в эту клетку, иначе — робот переместится в соседнюю от стены клетку и потеряет соответствующее количество прочности. Если прочность стала меньше либо равна нулю, то робот ломается на этой команде.

Задача H. Миграция великанов

Представим себе все данные в виде неориентированного графа, в котором вершины — это места обитания, а рёбра — рейсы. Заметим, что из любой вершины возможно перейти в не более, чем одну другую вершину — ребро до которой самое дешёвое. Построим новый ориентированный граф, в котором будут такие же вершины, а рёбра останутся только те, по которым можно пройти из вершины. Заметим, что если великан окажется в какой-либо вершине, то он останется в той же вершине, в которой начинал великан из этой вершины, кроме случая, когда есть путь обратно. Это означает, что можно запустить поиск в глубину на новом графе от каждой вершины. Если в этом поиске мы приходим в вершину, в которой есть ребро назад, то это финальная вершина для всех на пути к этой вершине от старта; если же мы пришли в уже обработанную вершину, то ответ для неё будет совпадать и для всех вершин на пути к ней. Если применить эти оптимизации, то сложность решения будет равна $O(N + M)$.