Задача A. Рисунок

Из определения *соответствующих* рисунков следует, что либо в итоговых состояниях лампочек «1» соответствует «#», а «0» – «.», либо «0» соответствует «#», а «1» – «.».

Значит для решения достаточно рассмотреть ответы для обоих вариантов и взять из них минимальный. Для того, чтобы найти ответ для одного из вариантов нужно потратить 1 единицу времени для каждой лампочки, которая не совпадает с соответствующим ей типом.

Задача B. Бугристый массив

Заметим, что если некоторый элемент строго больше своих соседей, то соседние элементы могут быть только строго меньше соответствующих им соседних элементов. Это означает, что если $a_1 < a_2$, то $a_2 > a_3$ и наоборот.

Для решения рассмотрим 2 случая сравнения a_1 и a_2 в итоговом массиве. Пусть $a_1 < a_2$ в итоговом массиве, тогда если это неравенство уже выполнено, не будем изменять a_2, иначе увеличим a_2 до значения $a_1 + 1$. Это является оптимальной стратегией, т.к. увеличение a_2 только «улучшает» требуемое неравенство $a_2 > a_3$, а значит, если нам необходимо применить некоторое количество операций для достижения $a_1 < a_2$, выгодно применять их к a_2. После достижения $a_1 < a_2$ увеличение a_2 только «улучшает» $a_2 > a_3$, однако уменьшение a_3 влияет на это неравенство также, но возможно «улучшает» неравенство $a_3 < a_4$, а значит, если нужно выполнить $a_2 > a_3$, не хуже будет применять операции к a_3. Таким образом, это оптимальное количество операций, которые нужно применить к a_2.

Случай, когда $a_1 > a_2$, рассматривается аналогично. После применения операций к a_2 рассмотрим неравенство, связывающее a_2 и a_3, которое будет однозначно определено предыдущим. Выполним этот процесс всего $n - 1$ раз для каждого из двух вариантов сравнения a_1 и a_2. Получим 2 возможных ответа, из которых выбираем наименьший. Асимптотика решения: $\mathcal{O}(n)$.

Задача C. Шахматы

Для решения задачи запустим рекурсивную функцию из точки, где находится конь. Эта функция будет переходить во все возможные поля, куда может пойти конь (с учётом того, что из поля «0» нельзя никуда ходить) и поддерживать текущее количество съеденного материала, а также запускать себя на не более чем 3 шага. Тогда при достижении клетки достаточно обновлять максимальное количество взятого материала(изначально его можно установить в -1), которое можно собрать, достигнув этой клетки. Ответ будет находиться в клетке «0» .

Задача D. Олимпиады

Рассмотрим граф, в котором есть вершины от 1 до n, соответствующие школьникам, в котором рёбра соответствуют лучшим друзьям. В нём из каждой вершины исходит 1 ребро и в каждую вершину входит 1 ребро. Рассмотрим произвольную вершину x, переместимся по ребру, исходящему из неё, затем по ребру из вершины, в которую мы пришли и т.д. В некоторый момент мы придём в ранее посещённую вершину. Она не может быть отличной от x, так как это означало бы, что в неё можно попасть по двум рёбрам. Значит первой совпадающей вершиной будет x. Это верно для любой вершины графа, значит граф разбивается на набор из простых циклов.

Каждый школьник участвует в олимпиаде, на которую зарегистрировался его лучший друг. Покажем, что школьник x должен участвовать во всех олимпиадах из «своего» цикла. Рассмотрим любую олимпиаду из этого цикла. Если кто-либо участвует в ней, то тот ученик, для которого участник является лучшим другом, тоже участвует в ней. Повторив это рассуждение конечное число раз, получим, что все в цикле должны участвовать в олимпиаде любого школьника этого цикла. Заметим, что если это верно для всех циклов, то все условия верны, а значит в этом случае достигается минимум, равный сумме квадратов длин всех циклов. Для решения остаётся посчитать длины циклов. Это можно сделать, создав массив не посещённых чисел от 1 до n и рассмотрев все числа от 1 до n. Когда алгоритм встречает число, которое не было посещено, нужно пройтись по соответствующему циклу и найти его длину.

Задача E. Эксперимент

Рассмотрим случай, когда все частицы одного типа. Тогда для быстрого решения задачи можно

хранить все энергии в множестве (`std::multiset` в C++), которое поддерживает добавление, удаление и поиск минимума за $\mathcal{O}(\log n)$, и число d, показывающее, на сколько должны быть увеличены все элементы множества. Тогда операция 3 заключается в увеличении d. Для добавления x нужно добавить $x - d$, и аналогично с удалением. Сумму можно поддерживать отдельно и пересчитывать после каждого запроса.

Тогда для решения задачи будем поддерживать такую структуру для каждого типа частицы, а также сумму всех элементов и множество минимумов. При выполнении операции только одна структура изменяется, значит, можно учитывать только её изменения для пересчёта суммы и множества минимумов. Таким образом, данное решение работает за $\mathcal{O}(q \log q)$.