Разбор задачи «Числа»

Во-первых, заметим, что у любого числа, не превосходящего 10^9, есть не более 9 различных простых делителей: $2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 = 6469693230 > 10^9$.

Во-вторых, понятно, что у набора чисел есть нееденичный общий делитель тогда и только тогда, когда эти числа делятся на какое-то одно простое число.

Поскольку все числа нужно разбить на группы, то возьмем первое число A_1 (или любое другое) и будем формировать группу чисел для него. Числа в группе должны делиться на некоторый простой делитель числа A_1, поэтому посчитаем все простые делители A_1, запишем их в массив и будем пробовать для каждого из них построить разбиение. Зафиксировав простой делитель p_1 числа A_1, отнесём в первую группу все числа, которые делятся на p_1. В качестве A_2 выбираем число, которое не вошло в первую группу. Зафиксировав его простой делитель p_2, относим во вторую группу все числа, делящиеся на p_2. Все числа, которые остались обязаны попадать в третью группу чисел. Можно либо вычислить НОД оставшихся чисел стандартным образом, либо поступить как и прежде – взять третье число A_3 и перебрать его простые делители.

Поскольку простых делителей у каждого из чисел A_1, A_2, A_3 не более 9 и только для них мы запускаем вложенные циклы перебора делителей, то время такого алгоритма будет $9(O(n) + 9(O(n) + 9(O(n)) + O(\sqrt{N}) + O(\sqrt{N})) + O(\sqrt{N}) \approx 9^3 N$.

Разбор задачи «Летняя школа»

Заметим, что если известно, сколько человек суммарно увозит первый автобус на вокзал и сколько в аэропорт, то легко посчитать время, которое потребуется, чтобы увезти всех участников – остальных учеников на вокзал и в аэропорт увозит второй автобус. Единственное, что необходимо понять – куда поедет каждый автобус в последний раз, ведь возвращаться уже будет необязательно. Для этого нужно перебрать все четыре возможности, по две для каждого автобуса.

Однако такое решение неоптимально. Более быстрый способ – решать задачу двоичным поиском по ответу T. Такой способ подходит потому, что, если за какое-то время можно увезти всех учеников в нужные места, то и за большее время тоже можно это сделать. Остается понять, как проверить, можно ли увезти всех за данное время T'. Для этого переберём количество людей, которое увезёт суммарно первый автобус на вокзал. Тогда остальных людей на вокзал должен увезти второй автобус. На это они потратят какое-то время T''. Значит, у них остаётся $T' - T''$ времени на то, чтобы развезти всех в аэропорт. Однако, нужно опять перебрать 4 возможности, куда поедут первый и второй автобус в последний раз. Вычисляем, сколько людей может увезти каждый автобус за оставшееся время, и заключаем, хватает ли времени T', чтобы отвезти всех желающих, или нет.

Разбор задачи «Прогрессия»

Чтобы найти необходимую арифметическую прогрессию, достаточно перебрать все прогрессии, которые встречаются в исходной последовательности, и выбрать среди них требуемую. Любая такая прогрессия задается своим началом и концом. Заметим, что достаточно смотреть лишь те прогрессии, которые нельзя продолжить вправо, поскольку у остальных заведомо длина не может быть максимальной. Поэтому достаточно для каждого элемента посчитать длину прогрессии, которая начинается с него.

Однако, этот вариант решения не является оптимальным: если мы построили какую-то прогрессию длины больше 2, например, 1 5 9 13, то нет смысла проверять прогрессии, которые начинаются со второго, третьего и т.д. элемента (5 и 9 в таком примере). Действительно, если прогрессия, начинается со второго элемента, то она будет подпоследовательностью в исходной прогрессии, поскольку её начальный элемент в ней лежит и шаг такой же, поэтому такие прогрессии не будут максимальными по длине. Например, если прогрессия закончилась на 13, как в примере, то с 5 и 9 мы получим 5 9 13 и 9 13, соответственно, и ничего более.

Таким образом, можно хранить номер последнего элемента в предыдущей арифметической прогрессии и следующую прогрессию начинать строить с него, пропуская все промежуточные элементы, если они есть. Время такого алгоритма линейное от количества элементов в исходной последовательности.

Разбор задачи «Звонки»

Рассмотрим граф, в котором вершины – это участники команды, а направленные ребра связывают того, кто звонит с тем, кому он звонит. На время забудем о направлениях на рёбрах и разобьём граф на компоненты связности. Поскольку в исходном графе из каждой вершины выходит одно направленное ребро, то, если в компоненте k вершин, то и рёбер будет k штук. Поэтому каждая компонента связности содержит ровно один цикл и «приклеенные» к нему деревья. Если теперь вспомнить про направления, то из условия задачи следует, что направления ребер все идут вдоль цикла, то есть по циклу можно пройти по направленным рёбрам. Также в каждом приклеенном дереве из каждой вершины можно по направленным рёбрам добраться до цикла.

Понятно, что задачу нужно решать для каждой компоненты связности отдельно. Для каждой компоненты есть две возможности – если кроме цикла ничего нет, тогда понятно, что мы должны выбрать, как минимум, одного человека и одного всегда хватит, чтобы новость распространилась по всему циклу. Вторая возможность – есть приклеенные к циклу деревья. В каждом таком дереве обязательно есть вершины, в которые не входит ни одного направленного ребра. Тогда мы обязаны выбрать соответствующих этим вершинам учеников – поскольку им никто не звонит, то они могут узнать новость только от капитана. Но тогда нетрудно понять, что они смогут распространить новость по всей компоненте связности: те вершины, которые находятся в каком-то дереве, но в них входит направленное ребро могут получить новость по этому ребру, поскольку, если пойти по входящим ребрам назад, то мы всегда упрёмся в того, у кого нет ни одного входящего ребра, а он знает новость. Наконец, до вершин в цикле новость тоже дойдёт, поскольку она дойдёт до той вершины, которая является общей для цикла и какого-то дерева, а потом распространится по циклу.

Суммируя все замечания, задачу можно решить следующим образом. При чтении данных подсчитываем у каждой вершины количество входящих в неё ребер. Затем запускаем из каждой вершины, в которую не входят рёбра, обход графа в глубину или в ширину и находим все вершины достижимые из этих вершин. Оставшиеся вершины находятся в компонентах связности, которые являются циклами. Далее считаем количество таких компонент связности, например, опять запуская обход графа в глубину или в ширину. Ответом будет сумма числа вершин, в которые не входят никакие рёбра, и числа компонент связности, которые являются циклами.

Разбор задачи «Зайцы и капуста»

Отметим точки на плоскости, которые соответствуют столбам и вилкам капусты. Из условия понятно, что вершины каждого нового забора образуют выпуклый многоугольник, как и исходный забор. Очевидно, что задача о нахождении количества точек внутри выпуклого многоугольника эквивалентна нахождению количества точек снаружи.

Для каждой пары столбов можно посчитать сколько вилков капусты лежит по каждую сторону от прямой, соединяющей эти два столба. Тогда, поскольку все вилки лежат внутри исходного многоугольника, то нужное количество капусты снаружи многоугольника из запроса будет равно $m - \sum_{i=1}^{q} K_i$, где K_i – количество вилков капусты для i-ой стороны, которые лежат с многоугольником из запроса по разные стороны. Такая реализация будет иметь асимптотику $O(n^2 \cdot m + \sum_{i=1}^{q} K_i)$.

Теперь возьмём произвольный столб. Сдвинем начало координат в точку, соответствующую данному столбу. Отсортируем все точки, кроме выбранного столба, по полярному углу. Теперь пройдя по этому массиву можно легко узнать сколько вилков капусты лежит с каждой из сторон. Асимптотика такого алгоритма $O(n \cdot (n+m) \cdot log_2(n+m) + \sum_{i=1}^{q} K_i)$.