Разбор задач 3-го (заключительного) этапа Всесибирской открытой олимпиады школьников по информатике 2017/2018

Задача A. Фёдор и палиндромы.

Заметим, что порядок букв в словах не важен: из некоторого слова, переставляя буквы, можно получить палиндром, если в нём есть не более одной буквы, которая встречается нечётное количество раз. Будем представлять слова в виде масок – целых чисел, в двоичной записи которых i-й бит равен единице тогда и только тогда, когда соответствующая буква встречается в слове нечётное число раз (таким образом, каждое такое число имеет 26 бит в двоичном представлении).

Чтобы посчитать количество пар строк, удовлетворяющих условию задачи, достаточно перебрать все пары и проверить для каждой такой пары сформулированное выше условие возможности получения палиндрома. Однако перебор всех пар не будет укладываться по времени так как требует порядка $O(N^2)$ операций, где N – количество слов в тесте Фёдора. Чтобы избежать перебора всех пар, посчитаем также количество повторений среди всех полученных масок для каждой из масок. Для этого можно воспользоваться ассоциативным массивом (например std::map или std::unordered_map в $C++$). Перебирая по очереди слова, точнее соответствующие им маски, будем смотреть сколько уже есть масок таких, что для каждого битового разряда сумма битов по модулю 2 даст 0 или не более чем в одном разряде сумма по модулю 2 даст 1. Для этого можно воспользоваться операцией xor – будем перебирать все маски с единственным единичным битом, а также нулевую маску – брать xor этих масок с текущей маской и прибавлять к ответу количество масок, полученных в результате xor.

Задача B. Химические реакции.

Естественно, эту задачу можно решать моделированием: пытаться применять доступные реакции к уже имеющимся веществам. Понятно, что через определённое число шагов новые вещества уже не будут получаться и поэтому мы получим нужный список веществ. Однако данный алгоритм будет неэффективным по времени.

Для решения задачи будем использовать следующие структуры данных:

1) множество полученных элементов (в $C++$ например можно завести std::set) – изначально сделаем его пустым;

2) массив реакций, в котором для каждой реакции будем хранить какие из её реагентов уже получены (в начале заполняется в соответствии с исходными данными);

3) массив веществ, в котором для каждого элемента будет содержаться список реакций, в которых он участвует.

При помощи этих данных можно решить задачу следующим образом: определим функцию, обрабатывающую некоторое вещество, при этом каждый элемент будем обрабатывать не более 1 раза. Запустим эту функцию от всех веществ, которые имеются в начале. При каждом вызове функция проверяет содержится ли данное вещество в множество полученных. Если содержится, то выходим из функции, если нет, то добавляем его в множество и для всех реакций, в которые входит этот элемент, помечаем, что мы его получили. Если в какой-то из этих реакций окажется, что оба реагента получены – вызываем функцию рекурсивно от результата этой реакции.

Таким образом в результате в множестве полученных элементов будет лежать ответ.

Задача C. Весёлые сообщения.

Для решения данной задачи можно применить метод динамического программирования. Заметим, что не имеет смысла частично удалять символы между участками подряд идущих закрывающихся скобочек, т.к. такие удаления не изменяют ответ. Посчитаем относительно начальной строки два массива чисел: первый массив – длины промежутков между закрывающимися скобочками, а второй массив – длины отрезков, состоящих только из закрывающихся скобочек.

Для динамики заведём трехмерный массив $dp[i][j][l]$, в котором будем хранить максимальную сумму, которую можно получить располагаясь на участке с номером i (под участком имеется в виду отрезок из символов ")"), ранее удалив j символов из строки и получив длину последнего отрезка (в позиции, на котором мы находимся) равную l. В данном случае l может быть больше начальной длины в позиции i, т.к. мы могли удалить некоторые символы, и отрезки из ")" объединились.

Третий параметр нам необходим, чтобы быстро пересчитывать значение динамики. Из позиции, в которой мы стоим, можно попытаться удалить следующий отрезок символов и объединить два отрезка, состоящих из закрывающихся скобочек, либо ничего не делать и просто перейти к следующему шагу. В результате, необходимо найти максимум среди всех значений динамики на последнем отрезке.

Задача D. Антон и xor.

Давайте для начала рассмотрим более простую задачу, в которой все числа исходной последовательности равны либо 0, либо 1. Тогда мы можем просто посчитать количество нулей и количество единиц, и выбрать $x = 1$, если единиц оказалось больше, чем нулей, и 0 – в противном случае. Понятно, что какие-то бОльшие значения в качестве x нам выбирать нет смысла – это выльется в то, что для всех чисел, которые есть в исходной последовательности, мы добавим в каком-то двоичном разряде ещё 1, что может только увеличить сумму.

Теперь вернёмся к исходной задаче. Заметим, что когда мы выполняем операцию xor над числами, то можем рассматривать все двоичные разряды независимо друг от друга. Значит мы можем применять тот же подход, который был описан выше для каждого разряда по отдельности, и таким образом построить x.

Так как числа у нас не превосходят 10^{18}, то нам достаточно просмотреть 60 младших разрядов. Также заметим, что считать сумму при таком подходе не требуется, достаточно посчитать количество нулей и единиц в каждом из разрядов. Итоговая временная сложность $O(N)$.

Задача E. Прямоугольный треугольник.

Заметим, что положительные числа a, b, c такие, что $a \le b \le c$, являются сторонами прямоугольного треугольника тогда и только тогда, когда $a^2 + b^2 = c^2$. Действительно, достаточно понять, что выполнено неравенство треугольника – длина большей стороны, меньше суммы двух наименьших. В нашем случае имеем $(a + b)^2 > c^2$, поэтому требуемое неравенство выполнено. Второе соображение заключается в том, что если изменять одно из чисел, например, a на не более чем k, то мы можем получить все числа от $a - k$ до $a + k$, при этом квадрат числа также принимает все промежуточные значения от $(a - k)^2$ до $(a + k)^2$. При этом надо учесть, что отрицательные значения числа использовать нельзя. Таким образом нужно проверить изменения для каждого

из чисел, при этом каждое число может в треугольнике оказаться либо катетом, либо гипотенузой. Зафиксировав число и его роль в треугольнике нужно лишь проверить будет ли среди промежуточных значений выполнено равенство из теоремы Пифагора или нет. Заметим, что все вычисления можно выполнять в целых числах: чтобы избежать вычисления квадратного корня, можно сравнивать квадраты чисел.