Задача A. Нейросеть

Рассмотрим следующую задачу: пусть сумма $S = a_1 + \ldots + a_n$ фиксирована, и нужно максимизировать произведение $a_1 \cdot \ldots \cdot a_n$. Это известная задача, и она решается так: сначала положим $a_i = \lfloor \frac{S}{n} \rfloor$, а потом прибавим по единичке к некоторым из a_i так, чтобы сумма стала равна S. Теперь для решения исходной задачи будем бинарным поиском искать минимальную сумму S такую, что при ней $\max a_1 \cdot \ldots \cdot a_n >= K$. Здесь есть один тонкий момент, который заключается в том, что для фиксированной S $\max a_1 \cdot \ldots \cdot a_n$ может не влезть в стандартные целочисленные типы. Для этого можно перед домножением на a_i проверять, правда ли $\lceil \frac{K}{a_i} \rceil \leqslant a_1 \cdot \ldots \cdot a_{i-1}$, и если это так, то можно досрочно завершить проверку на этом шаге, не вычисляя произведение до конца.

Задача B. Посылки в заморозку

Предварительно найдем лидера до заморозки. Затем, при обработке запроса второго типа будем считать, что все задачи, которые данная команда посылала в заморозку, ей решены, то есть последние посылки по каждой отправленной в заморозку задаче — корректны, а все посылки других команд в заморозку — некорректны. Исходя из этого посчитаем результат команды и сравним с лидером до заморозки.

Задача C. Счастливые дни

Давайте разобьем нашу перестановку на простые циклы. Простой цикл — это последовательность элементов a_1, a_2, $\ldots$, такая, что при применении перестановки элемент a_1 переходит в a_2, элемент a_2 — в a_3 и т.д., последний элемент в последовательности переходит в a_1.

Простые циклы перестановки можно найти следующим образом. Начнем с элемента 1. Он переходит в $p[1]$, тот в свою очередь в $p[p[1]]$, и т.д., до тех пор, пока мы не вернемся в 1. Получили один простой цикл. После этого берем первое число, в котором мы еще не были, и продолжаем поиск следующего цикла с него, и т.д.

Отлично. Теперь заметим, что если мы возводим цикл в степень, равную длине цикла, то все элементы цикла возвращаются на свои места. Таким образом для возведения перестановки в степень S, если цикл имеет длину l, то сдвиг по циклу нужно на самом деле применить $S \bmod l$ раз.

Рассмотрим все циклы в перестановке, пусть длина i-го цикла равна l_i. Так как $\sum l_i = n$, то количество различных длин циклов (количество различных чисел среди l_i) не превышает $2\sqrt{n}$.

Действительно, допустим, что это не так. Пусть есть $2\sqrt{n} + 1$ различных длин. Их сумма не меньше чем $1 + 2 + 3 + \cdots + 2\sqrt{n} + (2\sqrt{n} + 1) = (2\sqrt{n} + 1)(2\sqrt{n} + 2)/2 = (4n + 6\sqrt{n} + 2)/2 > n$

Теперь давайте для каждой уникальной длины вычислим остаток S при делении на нее. Вычислять остаток можно за $O(M)$, где M — количество цифр в записи числа S. Таким образом, будет сделано $O(M\sqrt{n})$ операций.

Теперь для каждого простого цикла мы знаем, в какую степень (не превосходящую его длины) его надо возвести, и просто в лоб применим эту операцию к каждому циклу. Это можно сделать за $O(n)$.

Осталось просто найти инверсии. Сделать это можно за $O(n \log(n))$ любым стандартным алгоритмом, например, сортировкой слиянием.

Задача D. День города

Пусть $\mathrm{in}(v)$ — кратчайший путь от 1 до v, $\mathrm{out}(v)$ — кратчайший путь от v до n. Тогда ответ на задачу — это всегда $\mathrm{in}(v) + \mathrm{out}(u)$ для каких-то вершин v, u смежных с одной и той же вершиной w (кроме, возможно, случаев, когда надо занулить ребра из 1 или n, но эти случаи рассматриваются тривиально). Поэтому достаточно для каждой вершины посчитать минимальную сумму $\mathrm{in}(v) + \mathrm{out}(u)$ по смежным вершинам. Как же найти такую сумму, не перебирая все пары v, u? Очевидно, просто взять минимум $\mathrm{in}(v)$ и минимум $\mathrm{out}(u)$. При этом v и u могут совпасть, но это не проблема, так как этот случай просто говорит, что нам заведомо невыгодно занулять ребра, смежные с текущей вершиной.

Есть и другое решение. Построим новый граф, где каждой вершине v будут соответствовать три вершины v_-, v_0, v_+. Для каждого ребра (v, u, c) исходного графа добавим ориентированные ребра (v_-, u_-, c), (u_-, v_-, c), $(v_-, u_0, 0)$, $(u_-, v_0, 0)$, $(v_0, u_+, 0)$, $(u_0, v_+, 0)$, (v_+, u_+, c), (u_+, v_+, c). Тогда ответом будет минимум из кратчайших путей от 1_- до n_0 и n_+.

Задача E. Преобразование выражения

Заметим, что задачу можно решать при помощи бинарного поиска по значению наибольшего слагаемого (назовем эту величину M). Действительно, если для некоторого M мы смогли построить корректное выражение, удовлетворяющее всем условиям, то это же выражение будет корректным и для любого большего значения M. Разумеется, M может находиться в пределах от 0 до N — эти числа можно использовать в качестве границ бинарного поиска.

Пусть теперь у нас зафиксирована величина M и мы хотим построить корректное выражение, используя слагаемые, не превосходящие M. Эту задачу можно решить при помощи динамического программирования. Обозначим за $D[P, T]$ минимально возможную сумму, которую можно получить из исходного выражения, начиная с позиции P (здесь и далее позиции в строке нумеруются с 0) и используя не более T замен. Если при этом получить корректную строку нельзя, то просто запишем в качестве результата достаточно большое число, например, $N+1$. При $P = L$ мы имеем $D[L, T] = 0$ для любого T — так как мы использовали всю строку и больше слагаемых нет. Теперь значения D нужно посчитать для $0 \leqslant P < L$, $0 \leqslant T \leqslant K$. Давайте переберем величину C — количество символов, из которого будет состоять очередное слагаемое; разумеется, C должно быть в пределах от 1 до $\lceil \log_{10}(M) \rceil$. Далее для каждого C переберем величину R — сколько замен в исходной строке мы хотим сделать для этого слагаемого, $0 \leqslant R \leqslant C$. Теперь у нас есть часть исходной строки (длиной C) и мы можем сделать в ней R изменений чтобы она представляла собой некоторое число. Конечно, мы хотим получить наименьшее возможное слагаемое X — таким образом мы удовлетворим условию $X \leqslant M$ если это возможно, а так же получим минимальную возможную сумму слагаемых при таком подходе. Теперь надо понять, каким же образом произвести изменения в строке. Заметим несколько свойств:

1. любой символ оптимально будет заменять на '1' (если это первый символ в строке и строка состоит хотя бы из двух символов), либо на '0' (в противном случае);
2. в выбранной части строки нужно заменить все символы '+' на цифры в соответствии с пунктом 1; если это невозможно, то не рассматриваем данный вариант значений C и R;
3. если остались еще замены, то нужно заменять первые символы строки, которые не соответствуют пункту 1.

Итого, пусть мы получили число X, преобразовав часть строки длиной C и сделав R замен в соответствии с описанными выше правилами. Если $X \leqslant M$, то мы можем сделать переход — оставшаяся часть строки будет начинаться с позиции $P_1 = P+C+1$ (после слагаемого надо добавить знак '+', исключение составляет случай, когда строка закончилась), а количество замен, которое мы можем сделать будет $T_1 = T - R$. Если же мы не окажемся в конце строки, то, возможно, потребуется еще одна замена, чтобы записать '+' после слагаемого, т.е. $T_1 = T - R - 1$ в таком случае. Теперь мы можем обновить значение $D[P, T]$: $D[P, T] = Min(D[P, T], X + D[P_1, T_1])$. Если в итоге мы получим, что $D[0, K] \leqslant N$, то для данного M решение существует, в противном случае, для данного M решения нет.

Таким образом, формально асимптотическая сложность алгоритма есть $O(L^2 \cdot \log^3(N))$.

Задача F. Контроль светофоров

Эта задача была самой простой на олимпиаде. Всего есть четыре варианты действий монтажников: все сделать правильно, перепутать только реле, перепутать только порядок проводов, и перепутать и то и другое.

Нам дано, что хотел сделать Иван Дмитриевич и что получилось. Переберем все четыре варианта и посмотрим, какие лампочки загорятся в таком случае. Если это совпадает с тем, что получилось — то этот вариант действий монтажников допустим, запоминаем этот вариант.

Перебрав все четыре варианта, посмотрим на полученный список допустимых вариантов. Если он пустой, то надо вывести `ERROR`. Если в нем только один вариант действий, то выведем соответствующую строку (`OK`, `1`, `2` или `3`). Если же в нем более одного варианта, то ответ `?`.