Задача A. Данис Хэмминг

В подгруппах предлагалось реализовать различные переборы.

Найдем первый разряд, в котором числа отличаются, и обозначим его за k (единицам соответствует первый разряд). Тогда заметим, что ответ не может превышать k, ведь остальные разряды всегда совпадают.

Приведем пример, на котором достигается k:

$$x = \dots a99999999\dots, \qquad y = \dots(a+1)00000000\dots,$$

где a — цифра из l в k-м разряде.

Задача B. Улица

Заметим, что ответ — количество соседних пар различных элементов $+1$.

Подгруппа 1: за $O(n)$ после каждого запроса пересчитаем ответ, получим $O(nq)$.

Полное решение: на самом деле после изменения одного элемента меняются лишь 2 пары соседних элементов, их и обновим, получив $O(n+q)$.

Задача C. Данис Лобода

Подгруппы 1 и 2: перебираем возможную длину и сравниваем для каждого символа количество вхождений.

Подгруппа 3: мы перебираем только делители n, далее для каждого блока создаем массив подсчета и далее сравниваем их. Получаем $O\big(n \cdot d(n) \cdot a\big)$, где $d(n)$ — количество делителей числа, a — размер алфавита. Но на самом деле мы делаем лишь $s(n)$ сравнений, где $s(n)$ — сумма делителей числа n. В нашем случае она не превосходит 2160576. Итого получаем

$$O\big(n \cdot d(n) + s(n) \cdot a\big).$$

Подгруппа 4: вместо того, чтобы каждый раз пробегаться по строке, будем узнавать количество букв a через префиксные суммы, получим $O\big(s(n)\big)$, а в свою очередь $s(n)$ не превосходит 22686048.

Полное решение: раньше мы сравнивали вектор, и это было долго, но ведь можно превратить вектор в число через хеширование:

$$h_s = \mathrm{cnt}_a \cdot \mathrm{base}^0 + \mathrm{cnt}_b \cdot \mathrm{base}^1 + \mathrm{cnt}_c \cdot \mathrm{base}^2 + \dots$$

Теперь получаем чистое $O\big(s(n)\big)$, как и в прошлой подгруппе (хеш узнаем через префиксные суммы).

Задача D. Задача без Даниса

Подгруппа 1: все длины строк равны. Если в алфавите m букв, то можно получить m^l слов, остается перебрать m и сравнить это число с n.

Наблюдение 1: нам точно хватит n букв.

Подгруппа 2: будем перебирать размер алфавита, выписывать все возможные слова, сортировать и брать минимально возможное, большее предыдущего для каждого слова. Всего слов не более

$$10^5 + 10^4 + 10^3 + 10^2 + 10^1 < 2 \cdot 10^5.$$

Наблюдение 2: можно делать бинпоиск по ответу, ведь до какого-то момента размера алфавита не хватает, а потом всегда достаточно.

Подгруппы 3 и 4: внутри бинпоиска будем делать жадную проверку, минимально увеличивая предыдущее слово. Если длина нового слова должна быть меньше, то обрезаем суффикс, иначе достаточно лишь дописать нули (вместо букв используем числа от 0 до m).

Полное решение: необходимо присваивать на отрезке и искать ближайший слева, меньший $m-1$, с этим прекрасно справляется дерево отрезков. Обычное заходит на 79 баллов, неявное на 100.

Также можно в стеке в сжатом виде хранить отрезки, и тогда каждый отрезок не более одного раза добавится и удалится.