Задача A. Аудитория

Определим диапазоны, в которые попадают номера больших, средних и малых аудиторий. Тогда по номеру аудитории мы сможем определить её тип.

Заметим, что в любую большую аудиторию всегда следует идти по первой лестнице, а в любую среднюю — по четвёртой лестнице. С малыми аудиториями дело обстоит чуть сложнее.

Обозначим через $2 \cdot y$ количество больших аудиторий (напомним, что количество аудиторий каждого вида является чётным числом).

Тогда средних аудиторий будет $4 \cdot y$, а малых — $14 \cdot x$ ($= 2 \cdot 4 \cdot y + 3 \cdot 2 \cdot y$).

Суммарное количество аудиторий будет $20 \cdot x$, и это значит, что рисунок к задаче легко масштабировать на любое количество аудиторий, соответствующее описанному плану размещения.

Пусть k — номер аудитории. Опираясь на рисунок, можно выделить следующие диапазоны номеров аудиторий и соответствующие им лестницы:

- для $1 \leqslant k \leqslant y$ — большая аудитория, подниматься надо по лестнице #1
- для $y + 1 \leqslant k \leqslant 2 \cdot y$ — малая аудитория, подниматься надо по лестнице #1
- для $2 \cdot y + 1 \leqslant k \leqslant 5 \cdot y$ — малая аудитория, подниматься надо по лестнице #2
- для $5 \cdot y + 1 \leqslant k \leqslant 8 \cdot y$ — малая аудитория, подниматься надо по лестнице #3
- для $8 \cdot y + 1 \leqslant k \leqslant 9 \cdot y$ — малая аудитория, подниматься надо по лестнице #4
- для $9 \cdot y + 1 \leqslant k \leqslant 13 \cdot y$ — средняя аудитория, подниматься надо по лестнице #4
- для $13 \cdot y + 1 \leqslant k \leqslant 14 \cdot y$ — малая аудитория, подниматься надо по лестнице #4
- для $14 \cdot y + 1 \leqslant k \leqslant 16 \cdot y$ — малая аудитория, подниматься надо по лестнице #3
- для $16 \cdot y + 1 \leqslant k \leqslant 18 \cdot y$ — малая аудитория, подниматься надо по лестнице #2
- для $18 \cdot y + 1 \leqslant k \leqslant 19 \cdot y$ — малая аудитория, подниматься надо по лестнице #1
- для $19 \cdot y + 1 \leqslant k \leqslant 20 \cdot y$ — большая аудитория, подниматься надо по лестнице #1

Как именно оформить запись этого набора условий — вопрос вкуса автора решения.

Задача B. Шаг вперёд, три шага назад

Решение этой задачи — аккуратная симуляция описанного процесса. Опишем его так, как должна выглядеть итерация цикла.

Будем перебирать баллы, полученные решениями, в порядке их «поступления».

Во-первых, потребуется поддерживать текущий максимум, для чего достаточно сравнивать очередное значение с уже имеющимся максимальным (сначала лучшей попыткой будет самая первая, но потом всё может измениться).

Во-вторых, потребуется поддерживать счётчик неудачных попыток улучшения. Важно заметить, что решение (пусть оно набирает u баллов), полученное непосредственно в результате «улучшения» текущего максимума, никогда не является причиной возвращения к текущему максимуму. Для этого придётся рассмотреть от одного до трёх следующих решений.

Любое из следующих двух решений, набирающее баллы, меньшие u, является причиной немедленного возвращения к текущему максимуму, а решение, следующее за ним, должно рассматриваться как полученное непосредственно в результате улучшения текущего максимума.

Если ни одно из указанных выше решений не привело к возвращению к максимуму, потребуется рассмотреть ещё одно решение. Это решение может только привести к изменению текущего максимума (или не привести, если набранные баллы меньше максимума). То, которое следует за ним, совершенно точно будет попыткой улучшения текущего максимума.

Отдельной обработки требует случай, когда формально должно произойти возвращение к текущему максимуму, но решение, которое должно приводить к возвращению, является последним, и возвращения не происходит. Такая ситуация рассмотрена в примерах входных данных.

Псевдокод решения может выглядеть так:

```
для каждого решения b[i] из входных данных
    если b[i] больше текущего максимума mxb
        mxb = b[i]
        индекс текущего максимума idx_mxb = i
        счетчик попыток count = 0
    иначе если b[i] <= mxb и count == 0
        более плохое решение pb = b_i
        count = count + 1
    иначе если count <= 2
        если b[i] < pb
            сбросить счетчик попыток count = 0
            если решение не последнее
                увеличить количество возвращений к решению с индексом idx_mxb
        иначе увеличить счетчик попыток count = count + 1
    иначе если count == 3
        сбросить счетчик попыток count = 0
            если решение не последнее
                увеличить количество возвращений к решению с индексом idx_mxb
```

Задача C. Сложный баланс

Опишем полное решение.

Сформируем для каждой из задач B и C по паре вспомогательных массивов (или по одному массиву / списку, в зависимости от реализации): $mb[]$, $indmb[]$ и $mc[]$, $indmc[]$.

На позиции $\#j$ в массиве $mb[]$ будет находиться максимальный из элементов $b[0], b[1], \ldots, b[j]$, а в массиве $indmb[]$ — индекс этого элемента.

Аналогично, на позиции $\#j$ в массиве $m[]$ будет находиться максимальный из элементов $[0], [1], \ldots, [j]$, а в массиве $indm[]$ — индекс этого элемента.

Заметим, что если найдётся несколько элементов с максимальным значением, следует хранить меньший из индексов этих элементов.

Более формально, $mb[j] = \max\limits_{i=0}^{j} b[i]$, $indmb[j] = \min\limits_{i=0}^{j} i \Big|_{b[i]=mb[j]}$.

Теперь можно выбрать один из массивов $mb[]$ или $mc[]$ в качестве «опорного» и, проходя по его элементам, сопоставлять ему элемент из другого массива таким образом, чтобы в сумме индексы этих элементов давали n. Максимальная сумма, полученная в результате такого перебора, и будет ответом, а количество попыток будет определено суммой фактических индексов (из массивов $indmb[]$ и $indmc[]$).

Также можно использовать «сжатые» версии массивов $mb[]$ и $mc[]$, в которые помещаются только очередные максимумы; в этом случае перебор элементов нужно будет выполнять с помощью техники двух указателей.

Важно обратить внимание на обработку случаев, когда все попытки тратятся на решение одной задачи.

Задача D. От L до L и до M и обратно

Опишем полное решение.

По имеющейся строке сформируем два вспомогательных массива $left[]$ и $right[]$.

Элемент $left[j]$ будет содержать количество букв l, находящихся левее позиции $\#j$ и не далее, чем на расстоянии $k+2$ от неё (можно вычеркнуть k символов, и ещё две позиции для l останется).

Аналогично, элемент $right[j]$ будет содержать количество букв l, находящихся правее позиции $\#j$ и не далее, чем на расстоянии $k+2$ от неё.

Теперь будем перебирать символы исходной строки и, как только найдётся символ m, будем вычислять для него количество вариантов получить пару l слева и количество вариантов получить пару l справа.

Предположим, что слева от очередного символа m имеется q букв l. Тогда способов выбрать два из них существует $Q = q \cdot (q-1)/2$. Действительно, имеется q способов взять одну букву l,

после чего останется $q-1$ буква l. Деление пополам нужно потому, что каждая пара символов будет получаться дважды: например, выбирая два из трёх символов, получим пары (1, 2), (1, 3), (2, 1), (2, 3) (3, 1) (3, 2). Пары, отличающиеся только порядком символов, будем считать неразличимыми с точки зрения нашей задачи.

Аналогичным образом, если справа от этого символа m имеется p букв l, то способов выбрать два из них существует $P = p \cdot (p-1)/2$.

Чтобы получить количество комбинаций $llmll$, нужно вычислить произведение $P \cdot Q$.

Ответом на вопрос задачи будет сумма таких произведений, подсчитанных для каждого символа m в строке.