Задача A. Перси Джексон и боги Олимпа

Автор задачи: Даниил Орешников, разработчик: Константин Бац

Первые группы тестов расчитанны на аккуратный разбор случаев. Не будем приводить здесь его описание, просто ограничимся тем, поскольку ответ в таком случае всегда от 0 до 2, достаточно проверить, можно ли получить ответ 0 или 1, а последовательности с $|a_i - a_{i+1}| \leqslant 1$ выглядят достаточно просто.

Обобщив это решение, можно получить решение **третьей группы**, однако проще придумать другое, подводящее в том числе к следующим группам: переберем, какое a_i будем менять, и его новое значение, после чего для каждого такого варианта за время $\mathcal{O}(n)$ пройдемся по массиву и вычислим ответ. По всем вариантам изменений останется только выбрать минимальный возможный ответ, и это и будет конечным ответом на задачу.

В следующих группах можно заметить несколько ключевых идей:

- нет смысла менять число, не участвующее в паре с максимальной на текущий момент разницей;
- если таких пар несколько непересекающихся по элементам, то хотя бы в одной из них числа не изменятся, и ответ не уменьшится;
- иначе — есть не более двух чисел, которые потенциально имеет смысл менять;
- если выбрано определенное a_i, то его значение имеет смысл менять на $\left\lfloor \frac{a_{i-1}+a_{i+1}}{2} \right\rfloor$, так как именно при нем минимизируется отличие a_i от соседних элементов (в случае крайних элементов имеет смысл присвоить $a_1 \leftarrow a_2$ или $a_n \leftarrow a_{n-1}$).

Таким образом, если применить идею, позволяющую не перебирать новое значение числа, получится решение **четвертой группы**, а если применить идею, позволяющую не перебирать позицию изменяемого элемента за $\mathcal{O}(n)$, получится решение **пятой группы**.

Полное решение требовало заметить, что из всех разностей соседних элементов мы меняем только две, поэтому если посчитать все разности соседних элементов и найти среди них три максимальных, по сделанному изменению можно будет однозначно понять, чему будет равен максимум в итоговом массиве. Например, если три максимальных разности равны d_1, d_2 и d_3, и нашим изменением мы поменяли d_1 на d'_1 и d_3 на d'_3, то в новом массиве максимальная разность будет равна $\max(d'_1, d_2, d'_3)$.

Таким образом, переберем не более двух позиций, которые имеет смысл менять, сделаем конкретное изменение и учтем изменения максимальных разностей, если хотя бы одна из них изменилась. Минимальный из полученных ответов (которых не более двух) и будет ответом на задачу.

Задача B. Перси Джексон и загадочные сны

Автор задачи: Даниил Орешников, разработчик: Константин Бац

В **первой и второй группах** работали полные переборы различного вида: в первой можно было за $\mathcal{O}(n!)$ перебрать произвольный порядок удаления символов, а во второй можно было применить различные отсечения этого перебора. Например, перебирать маски удаляемых символов за $\mathcal{O}(2^n)$, проверять, что остающиеся символы дают в объединении строку t, и после перебирать порядок удаления. Можно показать, что при существовании хотя бы одного решения перебор с мемоизацией в правильном порядке быстро его находит, а при отсутствии решения — быстро завершается.

Для строки, состоящей только из букв ‘a’ и не более одной буквы ‘b’ для **третьей группы тестов** можно было придумать отдельное решение. Действительно, t является подстрокой s только если имеет такой же вид, и при наличии буквы ‘b’ содержит не меньше символов ‘a’ слева и справа.

Таким образом, решение сводится к несложному разбору случаев: если буквы ‘b’ в строке s нет, любую непустую подстроку s можно получить удалениями второго символа, а если есть — надо проверить длины префиксов из s и t (нельзя получить больший и нельзя из непустого получить пустой), и проверить суффиксы — почти всегда можно получить меньший, кроме случая, когда требуется удалить все буквы ‘a’ справа от ‘b’, буква ‘b’ стоит на четной позиции, а из префикса ничего не удаляется.

Собственно, это наблюдение может подтолкнуть **полному решению**, а четвертая и пятая группы расчитаны на его упрощенные версии. Для полного решения необходимо было заметить, что буквы можно удалять жадно, следуя по шагам алгоритма проверки того, что t — подпоследовательность s.

Заметим, что в общем случае любой символ может быть удален, если было хотя бы одно удаление перед ним: если он изначально стоит на четной позиции, его можно удалить сразу, а иначе его можно удалить сразу после того, как произошло первое удаление символа, стоящего раньше (тогда этот сдвинется на четную позицию).

Алгоритм, проверяющий, что t — подпоследовательность s, проходит по символам s и t двумя указателями, и, если символы совпадают, то двигает оба указателя, иначе «пропускает» очередной символ в s. Тогда будем идти этим алгоритмом и поддерживать флаг «был ли уже удален символ из s». Как только мы встречаем символ, который необходимо удалить,

- если он — первый удаляемый символ, и стоит на четной позиции, удалим его и выставим флагу значение `true`;
- если он — первый удаляемый символ, и стоит на нечетной позиции, необходимо удалить предыдущий символ, который мы не удалили (а указатель на позицию в t сдвинуть на 1 влево), после чего либо удалить текущий, либо взять его на замену удаленному, если они равны;
- если он удаляется после какого-то другого, то есть флаг выставлен в `true`, то его можно просто пропустить и удалить.

Доказательство корректности такого жадного алгоритма мы приводить не будем, но доказать корректность легко можно у альтернативного алгоритма:

1. переберем первый удаляемый символ в s, пусть он стоит на позиции i;
2. тогда i должно быть нечетно, а префиксы s и t длины $i-1$ должны совпадать;
3. помимо этого, оставшиеся после символы в s должны содержать оставшиеся символы t как подпоследовательность.

Поскольку алгоритм поиска подпоследовательности на самом деле может найти для каждого суффикса s максимальный содержащийся в нем суффикс t, можно сделать такой предподсчет за $\mathcal{O}(n)$, а затем проверять каждый i описанным выше алгоритмом за $\mathcal{O}(1)$. Пятая группа, к слову, была расчитана на такое решение, но без предподсчета, с независимой проверкой для каждого суффикса.

Задача C. Перси Джексон и царство Аида

Авторы задачи и разработчики: Даниил Орешников и Константин Бац

Как обычно, **первая группа тестов** расчитана на переборное решение. Ограничения в этой группе позволяют перебрать стягиваемое ребро и после за $\mathcal{O}(2^m)$ найти минимальный остов, перебрав все возможные наборы ребер и проверив их на связность.

Для **второй и третьей групп** в случае графов специального вида достаточно было проанализировать, как устроено остовное дерево в таких графах. Если граф изначально является деревом, то он сам является своим единственным остовом. При стягивании произвольного ребра вес минимального остова уменьшается на вес этого ребра, поэтому выгодно стягивать ребро максимального веса. Аналогично, в случае графа, образующего один цикл, после стягивания произвольного ребра минимальный остов будет содержать все ребра, кроме максимального — тогда если стянуть максимальное ребро, в минимальном остове не будет двух максимальных ребер, и это будет оптимальным ответом.

Если граф является вершинным кактусом, то он представляет из себя дерево на циклах (то есть независимо были построены несколько непересекающихся циклов, а затем после их сжатия на том, что получилось, было построено дерево). Таким образом, более аккуратный разбор случаев, сочетающий в себе два предыдущих решения, проходил **шестую группу**.

Четвертая группа — просто обобщение переборного решения под произвольный алгоритм поиска минимального остова в графе. Можно перебрать стягиваемое ребро, в явном виде построить новый граф и запустить на нем алгоритм Прима или алгоритм Краскала. Если при этом заметить, что стягивание ребра аналогично тому, что его вес становится равен 0, а множество вершин никак не меняется, можно изначально отсортировать ребра по весу и после m раз запускать алгоритм Краскала, каждый раз пропуская определенное ребро — такое решение проходит **пятую группу**.

Для **полного решения** было необходимо воспользоваться свойством корректности алгоритма Краскала. Воспользуемся тем, что если на графе уже построен минимальный остов, то при добавлении нового ребра в граф достаточно добавить его к построенному минимальному остову и удалить максимальное ребро на образовавшемся цикле.

Построим минимальный остов алгоритмом Краскала, после чего переберем стягиваемое ребро. Если оно уже лежит в минимальном остове, вес остова уменьшается ровно на его вес. Иначе — уменьшается на максимальное ребро на пути между стягиваемыми вершинами.

Поиск максимума на пути в дереве — классическая задача, которую можно решать с помощью двоичных подъемов: посчитаем для каждой вершины и для каждого d от 0 до $\log_2 n$ величины $\texttt{up}[d][v]$ и $\texttt{max}[d][v]$ — верхний конец вертикального пути в дереве с нижним концом в v и максимальное ребро на нем, соответственно. Параллельно с подъемом до `lca` будем насчитывать максимальное ребро, и, найдя его, обновим ответ.

Задача D. Перси Джексон и встреча с Эридой

Автор задачи: Дмитрий Грунтов, разработчик: Даниил Орешников

Сразу заметим, что поскольку единственное число k стоит в центре последовательности, все остальные броски кубика должны давать значения от 1 до $k-1$, поэтому различных половин последовательности бывает не больше $(k-1)^{\frac{n-1}{2}}$. На самом деле их еще меньше. Рассмотрим правую половину. Если мы уже выбрали i-й ее элемент, то элемент с номером $2i$ однозначно задан (как и $4i$, $8i$, и так далее). Поэтому имеет значение только расстановка чисел на нечетных позициях, и различных последовательностей не больше $(k-1)^{\lceil \frac{n-1}{4} \rceil}$.

Собственно, эта оценка достигается в **первой группе**, в которой нет ограничений на пары подряд идущих чисел: каждую позицию на нечетном расстоянии от центра можно выбрать произвольным образом независимо от других, а четные позиции расставятся автоматически. А если каждую половину можно независимо выбрать $x = (k-1)^{\lceil \frac{n-1}{4} \rceil}$ способами, то ответом будет x^2.

Вторая и третья группа похожи решением на первую. Сразу избавимся от пар любимых чисел, содержащих число k — их автоматически будет не более одной. Для оставшихся просто переберем все возможные половины, содержащие не больше одного вхождения каждой пары рекурсивным генератором и перемножим соответствующие количества (половины, содержащие первую пару, можно объединять только с содержащими вторую или не содержащими ни одну).

Полный перебор всех последовательностей работает за $\mathcal{O}((k-1)^{n-1})$, что слишком долго и проходит только **пятую группу**. Перебор, сразу при генерации учитывающий условие на одинаковые элементы, работает за $\mathcal{O}((k-1)^{\frac{n-1}{2}})$, что тоже не укладывается в ограничения других групп. Если же перебирать половины независимо, получится $\mathcal{O}((k-1)^{\lceil \frac{n-1}{4} \rceil})$.

Для **шестой группы** надо было обобщить описанную выше идею: сделаем полный перебор всех половин, запретим любимым парам появляться больше одного раза, и для каждой половины посчитаем битовую маску `inc`, кодирующую множество пар любымых чисел, которые в нее вошли. Поскольку пары любимых чисел читаются слева-направо, а для левой и правой половины условие на одинаковые значения разное (для левой половины расстояния до центра растут справа-налево), посчитаем также маску $\texttt{inc}^R$ вошедших развернутых любимых пар.

Теперь посмотрим, какие пары половин можно объединить в последовательность. Возьмем половину a и половину b. Запишем развернутую a, число k и b подряд друг за другом. Вошедшие в итоговую последовательность любимые пары — это $\texttt{inc}^R(a) \mid \texttt{inc}(b)$. И такое объединение можно сделать только если их пересечение $\texttt{inc}^R(a) \,\&\, \texttt{inc}(b)$ пусто.

Переберем все возможные маски m_1, и им в пару все возможные маски m_2, и если $m_1 \,\&\, m_2 = 0$, добавим к ответу произведения количества половин, имеющих $\texttt{inc}^R = m_1$, и количества половин,

имеющих $\texttt{inc} = m_2$. Это решение работает за $\mathcal{O}(\text{перебор} + 4^m)$.

Одной из возможных оптимизаций было перебирать не все маски, а только «достижимые», то есть такие, для которых сгенерировалась хотя бы одна соответствующая половина. В **четвертой и седьмой группах** количество достижимых масок достаточно небольшое, и разной степени аккуратности решения с таким перебором проходили одну или две из них.

До **полного решения** остается только соптимизировать перебор: для каждой m_1 ей в пару можно поставить только подмаски $\sim m_1$, а такие пары m_1 и подмаски $\sim m_1$ можно перебрать за $\mathcal{O}(3^m)$. Такое решение набирало полный балл.