Задача A. Порядок

Это самая простая задача комплекта. Чтобы ее решить, нужно обратить внимание на два факта.

Первый важный факт: набор чисел (a, b, c) отсортирован, если $a < b < c$ или $a > b > c$ (напомним, что в задаче гарантировано попарное несовпадение любых двух значений).

Второй важный факт: надо проверять сортированность в столбцах, а не в строках.

Если (h_1, h_2, h_3) отсортированы — выводим H, иначе N. Аналогично для (w_1, w_2, w_3) и (t_1, t_2, t_3).

Задача B. Наивысший приоритет

В этой задаче по описанию процесса назначения приоритетов письмам нужно было выяснить, сколько раз происходило назначение приоритета 1.

Письмо получало приоритет 1, если было важнее всех предыдущих писем.

Решение, набирающее 60 баллов, состояло в проверке для каждого письма этого утверждения за $O(N)$. Для этого достаточно перебрать в цикле все предыдущие письма и сравнить их приоритеты с текущим. Итоговая асимптотика — $O(N^2)$.

Решение, набирающее 100 баллов, должно было хранить минимальный из итоговых приоритетов прочитанных нами писем. Если итоговый приоритет текущего письма оказался меньше текущего минимума, это значит, что в момент прочтения письму выдали приоритет 1 (увеличился ответ), а также обновили минимальный приоритет.

P.S. Больше слов от тех, кто писал все эти пояснения, примечания и таблицы.

Авторы задач пребывают в недоумении, почему именно эта задача вызвала столько непонимания.

В задаче описан естественный процесс: Кеша читает одно за другим письма и сортирует их по важности, постоянно поддерживая их относительный приоритет друг относительно друга.

Обратите внимание, что если в момент прочтения письма оно было важнее какого-то уже прочитанного — оно не перестанет быть важнее него в последующем.

Нас интересует, какие письма получали приоритет 1 в момент прочтения. Приоритет 1 означает, что все письма до него имели приоритет ниже (большее число), чем у прочитанного письма.

Но если в момент прочтения все предыдущие были менее важными (числа были больше), то и после завершения чтения всех писем их относительная важность осталась такой же.

Следовательно, письмо получало когда-то приоритет 1, если на момент его прочтения все прочитанные до этого письма имели итоговый приоритет ниже (числа были больше), чем итоговый приоритет этого письма.

Задача C. Маски

В этой задаче требуется просто аккуратно выполнить симуляцию процесса, описанного в условии.

Для начала инициализируем переменные, которые будут содержать ответы:

- количество масок = 0,
- минимальный срок ношения маски = $t + 1$ (обратите внимание, что тут совсем необязательно было использовать константы, содержащие множество девяток и/или нулей, в которых вы сами могли запутаться),
- максимальный срок ношения маски = 0.

Далее заведём две переменные — количество мест, которые Кеша уже посетил в этой маске, и суммарное время, в течение которого Кеша эту маску носит. Изначально они обе равны 0.

Теперь для каждого места, которое посещает Кеша, будем проверять следующие условия (все три пункта не зависят друг от друга:

- если Кеша не может продолжать в данной маске, но в ней он побывал хотя бы в одном месте, то обновляем ответы текущими значениями, после чего обнуляем количество мест, посещённых в текущей маске, и суммарное время, в течение которого Кеша эту маску носит.
- если количество посещённых мест равно 0, то Кеша надел новую маску, и надо увеличить счетчик масок.

– если условия ношения маски в новом месте соблюдены, то увеличиваем суммарное время ношения маски.

Важный частный случай — не забудьте обновить минимум/максимум после того, как Кеша посетил все места: возможно, что время использования и количество использований маски не превышено, но эту маску нужно учесть в использованных.

Задача D. Корпоративный чат

Чтобы набрать баллы за подзадачи с небольшими значениями n и t_j, было достаточно перебрать все возможные времена перерыва между первым и последним сообщением и для каждого времени вычислить суммарную задержку. Итоговая сложность такого решения $O(NT)$.

Решение, набирающее большее количество баллов, должно было учитывать следующий факт: Кеше выгодно проверять почту только в моменты времени прихода сообщений. В любом другом случае он проверит все те же сообщения, но к задержке перед каждым сообщением добавится дополнительное время.

Чтобы набрать 60 баллов, было достаточно перебрать время каждого сообщения и вычислить суммарную задержку, если бы мы читали сообщения именно в момент его прихода. Сложность такого решения $O(N^2)$.

Для решения, набирающего 100 баллов, проведём следующие оптимизации решения на 60 баллов: обозначим за p_i сумму всех t_j для $(1 \leqslant j \leqslant i)$. Также добавим, что $p_0 = 0$.

Предположим, что Кеша решил сделать перерыв и прочесть сообщения в момент времени t_i.

В таком случае суммарная задержка для сообщений, которые Кеша прочитает в этот перерыв, будет равна $(t_i - t_1) + (t_i - t_2) + \ldots + (t_i - t_i) = i \cdot t_i - (t_1 + t_2 + t_3 + \ldots + t_i) = i \cdot t_i - p[i]$.

К ней также нужно будет добавить задержку для всех сообщений, пришедших позже момента t_i, и прочитанных одновременно с последним сообщением: $(t_n - t_{i+1}) + (t_n - t_{i+2}) + \ldots + (t_n - t_n) = t_n \cdot (n - i) - (t_{i+1} + t_{i+2} + \ldots t_n)$.

Обратим внимание, что $(t_{i+1} + ti + 2 + \ldots + t_n) = (t_1 + t_2 + \ldots + t_n) - (t_1 + t_2 + \ldots + t_i) = p_n - p_i$.

Запишем итоговую формулу:

Задержка $= t_i \cdot i - p_i + t_n \cdot n - t_n \cdot i - p_n + p_i$

Задержка $= t_n \cdot n - p_n - i \cdot (t_n - t_i)$.

Величины t_n, n, p_n — сумма всех t в массиве — не меняются от сообщения к сообщению.

Следовательно, для минимизации задержки нам необходимо минимизировать с минусом (максимизировать) величину $i \cdot (t_n - t_i)$. Обратим внимание, что достаточно просто вычислить такую величину для каждого сообщения и найти самое раннее сообщение, у которого данная величина максимальна (в оригинальной формуле она с минусом, поэтому для минимизации задержки мы максимизируем данную величину).

Итоговая сложность решения $O(N)$.

Задача E. И снова о приоритетах

Обозначим за p_i итоговый приоритет (то, что мы хотим вывести).

В решении, набирающем 20 баллов, достаточно пересчитывать все текущие приоритеты в цикле, как и описано в условии: для всех $j < i$ таких, что $p_j \geqslant p_i$ увеличим p_j на 1. Предложенное решение работает за $O(N^2)$.

Для групп 3, 4 ($j - 2$ или $j - 49 \leqslant q_j$) рассмотрим следующий алгоритм. Сохраним для каждого текущего приоритета его позицию $pos_p r$ — такой индекс, что в текущий момент приоритет составляет $p[pos]$. Если Кеша читает письмо с приоритетом $q_j = j - d$, то мы увеличиваем приоритеты только писем $j - d$, $j - d + 1$, …, $j - 1$. Для каждого обновленного приоритета прибавим $+1$ по соответствующим позициям, а затем обновим массив соответствующих позиций. В конце итерации мы занесём в $pos_{q_j} = j$.

Это решение работает за $O(N \cdot D)$, где D — максимальное «отклонение» от максимального возможного значения. В группе 3 $D < 3$, в группе 4 $D \leqslant 50$, а $N \leqslant 10^5$.

Группы 5, 6 в целом аналогичны группам 3 и 4, но имеется несколько отличий:

– во-первых, нам теперь невыгодно прибавлять +1 ко всем обновлённым приоритетам. Поэтому мы сделаем следующий ход — мы прибавим единицу вообще всем приоритетам, но сделаем это быстро, а для более высоких приоритетов вычтем эту единицу из каждого явно.

– во-вторых, теперь для корректного поддержания позиций нам потребуется связный список, от головы которого мы будем перемещаться по более высоким (меньшим) приоритетам, а после вставлять нашу текущую позицию в список, не выполняя никаких лишних действий по перемещению всех элементов списка.

Осталось понять, как быстро прибавлять всем. Заметим, что к письму с номером j прибавляли 1 неявно при прочтении всех писем с номерами $j+1, j+2, \ldots, n$ — суммарно прибавили $(n-j)$.

То есть итоговый алгоритм такой: для каждого приоритета вычитаем 1 из каждого меньшего, а в конце к p_j прибавляем $(n-j)$.

Такое решение так же работает за $O(ND)$, где D в данных подгруппах — верхняя оценка на q_j.

Чтобы получить 100 баллов, можно было написать следующее решение:

– Изначально приоритеты всех писем равны тем приоритетам, которые мы получили на вводе.

– Найдём самое последнее (самое правое) письмо с приоритетом 1. Его приоритет уже точно ничто не изменит, так что занесём его в ответ. Оно же, в свою очередь, увеличит приоритет всех писем, прочитанных до него (стоящих левее). На место же самой единицы поставим число $N+1$ — заведомо большее любого другого приоритета.

– Следующим шагом найдём самое правое письмо с приоритетом 2 (приоритета 1 уже нет). Аналогичным образом занесём его в ответ, увеличим все числа слева от 2 и заменим 2 на $N+1$.

– Будем повторять описанный процесс, пока не заполним весь ответ.

Описанные операции можно выполнять за $O(NlogN)$ с помощью структуры данных «Дерево отрезков» (Segment Tree).

Подробнее про эту структуру данных вы можете узнать по ссылкам:

https://codeforces.com/edu/course/2/lesson/4

https://codeforces.com/edu/course/2/lesson/5 (усложненные темы)

Также существуют решения данной задачи с помощью дерамиды (декартово дерево, Cartesian Tree, treap; описание, например, здесь: https://habr.com/ru/post/101818), а так же с помощью дерева Фенвика (описание можно прочесть, например, здесь: https://petr-mitrichev.blogspot.com/2018/02/a-fenwick-bound-week.html).