Перекладывание ответственности

Автор задачи: Даниил Орешников, разработчик: Мария Жогова

Первая подгруппа расчитана на прямую реализацию описанного в условии процесса. Будем поддерживать указатель на текущую задачу x и количество уже распределенных в каждой задаче элементов $\texttt{cnt}[i]$. Тогда, выдавая разработчику очередной элемент, возьмем очередной элемент из задачи x, увеличим $\texttt{cnt}[x]$ на 1, после чего будем перемещать x по кругу вперед, пока не встретим следующую задачу, у которой $\texttt{cnt}[i] < c_i$. Решение работает за время $\mathcal{O}(n \cdot \max(c_i))$, поскольку перемещений x будет не больше, чем столько.

Во **второй подгруппе** можно было легко набрать баллы, проанализировав сколько элементов какой из двух задач достанется каждому разработчику. Не теряя общности, пусть $c_1 \leqslant c_2$, тогда первые $2c_1$ элементов будут распределяться по очереди из двух задач, а оставшиеся будут взяты из второй. В таком случае первому разработчику достанутся элементы суммарной сложности $\left\lceil \frac{c_1}{2} \right\rceil \cdot w_1 + \left\lfloor \frac{c_1}{2} \right\rfloor \cdot w_2$, а второму — все оставшиеся. Случай $c_1 > c_2$ полностью аналогичен.

Для **третьей подгруппы** также работает решение первой, однако приведем более сложное решение, которое будет шагом на пути к полному. Введем определение «фазы», которое будет дальше полезно для полного решения. Фазой t назовем отрезок времени, в который распределяются t-е элементы из каждой задачи, в которой есть хотя бы t элементов. Фазы, таким образом, будут иметь номера от 1 до $\max c_i$.

В условиях третьей подгруппы фаз будет не больше двух. Если все $c_i \leqslant 2$, то каждый разработчик, кроме, возможно, одного, получит один или два своих элемента в рамках одной фазы. Для того разработчика, которому, возможно, достанется последний элемент из первой фазы и первый из второй, посчитаем ответ отдельно, а для остальных достаточно построить префиксные суммы на участвующих в каждой фазе задачах, и находить итоговый вес как сумму на отрезке в рамках одной фазы.

Для следующих подгрупп обобщим описанное решение. Заметим, что среди всех фаз есть не более n различных. Действительно, все фазы с первой по $\min c_i$ одинаковы и состоят из всех задач по очереди. Следующие фазы до второго по величине c_i одинаковы и состоят из всех задач, кроме тех, в которых $c = \min c_i$, и так далее.

Заметим, что элементы, достающиеся очередному разработчику, образуют либо отрезок какой-то фазы, либо суффикс последней фазы, некоторое целое число следующих фаз целиком (возможно, ноль) и какой-то, возможно пустой, префикс следующей фазы. Отсортируем все задачи по убыванию c_i и посчитаем префиксные суммы w_i на полученном порядке задач.

Теперь будем перебирать разработчиков по очереди, поддерживая номер фазы и номер задачи в фазе, на которых мы остановились на предыдущем разработчике. Для решения **четвертой подгруппы** достаточно найти сумму на соответствующем отрезке или суффиксе текущей фазы за $\mathcal{O}(n)$, после чего обработать все предназначенные ему целиком фазы, и снова за $\mathcal{O}(n)$ добавить элементы из префикса следующей фазы, если надо.

Чтобы быстро обработать фазы, которые целиком отойдут данному разработчику, посмотрим на номер фазы p, в начале которой мы сейчас находимся: пусть он лежит между c_i и c_{i+1}, где c отсортированы по возрастанию. Тогда следующие $c_{i+1} - p + 1$ фаз одинаковы, и в каждой из них выбираются элементы из $n-i+1$ оставшихся задач. Сумму w_i всех задач в фазе мы предподсчитали, поэтому, если разработчику осталось выдать $\texttt{rem} \leqslant (c_{i+1} - p + 1) \cdot (n - i)$ элементов, выдадим ему $\left\lfloor \frac{\texttt{rem}}{n-i} \right\rfloor$ следующих фаз целиком за $\mathcal{O}(1)$, и префикс следующей за ними фазы размера $\texttt{rem} \bmod (n-i)$. Иначе, выдадим ему все оставшиеся фазы этого «блока» и перейдем к рассмотрению фаз между c_{i+1} и c_{i+2}.

Действия, которые выполняются «долго» — взятие отрезка/суффикса и префикса фазы. Помимо этого, для одного разработчика могут быть рассмотрены несколько «блоков» фаз между соседними c_i и c_{i+1}. Однако, вычисление суффикса и префикса производится для каждого разработчика не больше одного раза, а перемещений к следующему «блоку» одинаковых фаз будет не больше n. Итого, суммарно мы потратили $\mathcal{O}(n)$ времени на обработку всех фаз, не считая вычисления суммы на отрезках, суффиксах и префиксах, а на них — $\mathcal{O}(n^2)$ времени, чего достаточно для прохождения

четвертой подгруппы.

Для пятой и шестой подгруппы требовалось соптимизировать нахождение сумм на отрезках, суффиксах и префиксах внутри фаз. Сразу сведем эту задачу к поиску суммы на префиксе с помощью префиксных сумм. Осталось научиться быстро находить сумму на префиксе в фазе быстрее, чем за $\mathcal{O}(n)$.

В **пятой подгруппе** подходил любой способ находить сумму за $\mathcal{O}(\log^2 n)$. Мы не будем на них фокусироваться, так как эти способы достигаются идеями, очень похожими на идеи полного решения, но в каких-то местах менее оптимальными. Для **полного решения** можно было воспользоваться, например, одной из двух следующих идей.

Первая идея основана на дереве отрезков. Будем хранить дерево отрезков, в котором i-й элемент равен w_i, если в i-й задаче еще остались нераспределенные элементы, и 0 иначе. При переходе между фазами достаточно просто обнулить «закончившиеся» задачи. Суммарно это займет $\mathcal{O}(n \log n)$ времени как n запросов к ДО. Чтобы получить сумму первых k задач в фазе, то есть сумму на префиксе, надо сначала получить позицию k-го ненулевого элемента t, а затем сделать запрос суммы на отрезке $[0, t]$. Чтобы находить t, надо было хранить количество ненулевых элементов в каждом отрезке ДО, и либо делать бинпоиск и запросы количества нулей на отрезке за $\mathcal{O}(\log^2 n)$ (чего хватало для пятой подгруппы), либо делая спуск по дереву за $\mathcal{O}(\log n)$.

Альтернативное решение использовало декартово дерево по ключу, равному номеру задачи. Будем хранить в вершинах ДД w_i и поддерживать их сумму на поддеревьях. Аналогично, переход между фазами — это удаление нескольких элементов из ДД по ключу, а запрос суммы на префиксе фазы заключается в поиске k-й порядковой статистики, и взятия суммы в левом поддереве после `split` по найденному ключу. Обе операции можно выполнять за $\mathcal{O}(\log n)$, если поддерживать также размеры поддеревьев.