Рейд на транспортер

Автор задачи и разработчик: Мария Жогова

Переформулируем: требуется выбрать последовательность (a_i, b_i) такую, чтобы a_i и b_i неубывали, и при этом соседние b_i отличались не более, чем на x. При этом можно добавить в множество произвольную пару (a, b), и требуется максимизировать размер итоговой последовательности.

Первая подгруппа решается произвольным перебором. Вторую и третью можно было решить, заметив, что если $x = 0$, то требование из условия превращается в требование на равенство всех b_i в итоговой последовательности. Таким образом, решение — сгруппировать все пары по b_i (например, с помощью `unordered_map`) и выбрать максимальную по размеру группу. Очевидно, что

- в любой группе можно взять все пары, просто упорядочив их по a_i;
- к любой группе можно добавить пару $(10^9, b_i)$, увеличив размер последовательности на 1.

Таким образом, ответ будет равен максимальному размеру группы плюс 1.

При ограничении, что все a_i равны, задача сводится к выбору максимальной последовательности, в которой выполняется условие на b_i. Отсортируем все пары по b_i, после чего заметим, что последовательность, заканчивающаяся в i-м элементе, может иметь в качестве предыдущего любой из отрезка $[j, i)$, где $b_j \geqslant b_i - x$. Таким образом, можно считать динамику $\texttt{dp}[i]$ как $\max\limits_{b_j \geqslant b_i - x} \texttt{dp}[j] + 1$. Это максимум на скользящем окне, его можно поддерживать с помощью очереди с максимумом.

Осталось только учесть, что произвольный (a, b) можно добавить в рассмотрение. Заметим, что такая пара может «склеить» максимальную последовательность на префиксе, заканчивающуюся в b_i, для которого $b_i \geqslant b - x$, и максимальную последовательность на суффиксе, начинающуюся в b_j, для которого $b_j \leqslant b + x$. Переберем i, заметим, что чтобы между b_i и b_j помещался b, необходимо и достаточно, чтобы выполнялось $b_j \leqslant b_i + 2x$, таким образом ответ можно найти как

$$\max_i \left(\texttt{dp}_{\text{pref}}[i] + \max_{j > i \wedge b_j \leqslant b_i + 2x} \texttt{dp}_{\text{suf}}[j] + 1 \right).$$

Опять же, данный максимум по динамике на суффиксах можно находить с помощью скользящего окна с очередью с максимумом.

Это подводит нас к полному решению. Будем также считать динамику на префиксах и суффиксах, но упорядочим пары по a_i. Ответ будет вычисляться по похожей формуле:

$$\max_i \left(\texttt{dp}_{\text{pref}}[i] + \max_{j > i \wedge b_i \leqslant b_j \leqslant b_i + 2x} \texttt{dp}_{\text{suf}}[j] + 1 \right).$$

Однако в этот раз необходимо находить максимум на суффиксе, с дополнительным условием на b_i. Это можно делать с помощью дерева отрезков или декартового дерева: будем поддерживать максимумы `dp` для соответствующих b_i, и, итерируясь по i справа-налево, считать $\max\limits_{j > i \wedge b_i \leqslant b_j \leqslant b_i + 2x} \texttt{dp}_{\text{suf}}[j]$ запросом в ДО на отрезке $[b_i, b_i + 2x]$.

Суммарное время работы такого решения — $\mathcal{O}(n \log n + n \log 10^9)$, если использовать динамическое (неявное) дерево отрезков, и $\mathcal{O}(n \log n)$, если использовать декартово дерево.