Смешивание напитков

Автор задачи: Даниил Орешников, разработчик: Егор Юлин

Основной идеей для решения данной задачи будет наблюдение, что крепость кофе непрерывна. То есть если у нас изначально был кофе крепости P_0, а после какого-то изменения крепость стала равна P, то мы могли получить любую крепость от P_0 до P, выпив из тех же слоев, но меньше.

Далее будут использоваться следующие обозначения: P_0 — изначальная крепость кофе в стакане, t_i — необходимая крепость кофе в запросе. Кроме того, будем отдельно рассматривать случаи, когда $t_i < P_0$ и случаи, когда $t_i > P_0$. Заметим, что заменой всех $p_i \leftarrow -p_i$ и всех $t_i \leftarrow -t_i$ можно привести один случай к другому, поэтому далее опишем только решение для $t_i < P_0$, а обратный случай будет полностью аналогичен.

Для решения **первой подзадачи** можно было перебирать конкретное множество слоев, которые будут целиком или частично выпиты. Для каждого множества затем можно проверять, могла ли в какой-то момент крепость кофе быть равной t_i. Поскольку мы рассматриваем $t_i < P_0$, достаточно проверить, что если выпить эти слои целиком, итоговая крепость будет $\leqslant t_i$.

Итоговую крепость можно посчитать за $\mathcal{O}(n)$ или за $\mathcal{O}(1)$, если по ходу рекурсивного перебора множества слоев поддерживать суммарные количество кофеина и высоту. Среди всех множеств, которые подошли, надо выбрать то, у которого самый глубокий слой находится как можно выше. Итоговое время работы — $\mathcal{O}(q \cdot 2^n)$.

Для решения **второй подзадачи** можно соптимизировать перебор и перебирать, сколько первых слоев будут задействованы (то есть по сути перебирать ответ на запрос). Опять же, пользуясь непрерывностью, заметим, что достаточно проверить, что можно получить крепость $\leqslant t_i$.

А для такой проверки можно заметить, что если зафиксировать доступные верхние x слоев, достаточно выпить целиком те из них, у которых крепость больше t_i. Если этого не хватит, то, выпивая еще и слои с крепостью $\leqslant t_i$, итоговой крепости t_i мы не добьемся. Соответственно, перебирая x, выпьем все слои крепости $> t_i$, после чего проверим, что итоговая крепость стала $\leqslant t_i$. Итоговое время работы — $\mathcal{O}(q \cdot n^2)$.

В **третьей подзадаче** можно было соптимизировать предыдущее решение и вместо обычного перебора высот искать ответ бинарным поиском. Это позволяет делать монотонность по ответу: чем более высокую трубочку мы возьмем, тем шире будет интервал достижимых уровней крепости. Такое решение работает за $\mathcal{O}(q \cdot n \cdot \log n)$.

Это же решение оптимизируется до асимптотики $\mathcal{O}(q \cdot n)$ для **четвертой подзадачи**, если совместить идеи второй и третьей подзадач. А именно, что чтобы получить крепость $t_i < P_0$, нет смысла трогать слои с крепостью $\leqslant t_i$.

Будем перебирать для каждого запроса затрагиваемые слои кофе сверху вниз. Тогда для каждого очередного слоя j, если $p_j \leqslant t_i$, мы его пропускаем, а если $> t_i$, выпиваем целиком. Как только общая крепость напитка достигнет значения меньше t_i, мы нашли наш ответ. Для быстрого вычисления общей крепости напитка достаточно поддерживать суммы по еще не выпитым слоям, фигурирующие в числителе и знаменателе формулы для P.

Пятую подзадачу можно было решать с помощью бинарного поиска по ответу. Используя рассуждения, описанные выше, поймем, что достаточно выпивать все слои, крепость которых больше t_i, а если слои отсортированы по крепости, то интересующие нас слои всегда будут образовывать некоторый отрезок.

Таким образом, можно было предподсчитать префиксные суммы $p_i \cdot h_i$ и p_i по слоям, после чего для ответа на запрос

- делать бинпоиск по ответу;
- внутри бинпоиском находить интересующий нас отрезок слоев;
- выпивать их и проверять, что итоговый уровень крепости нам подходит.

Вместо внутреннего бинпоиска можно было отсортировать запросы по t_i и использовать метод двух указателей: для меньших t_i нам всегда будет нужна не меньшая высота трубочки, и на со-

ответствующем суффиксе слоев нас будет интересовать не менее длинный отрезок. Время работы: $\mathcal{O}(q \cdot \log^2 n)$ или $\mathcal{O}(q \cdot \log n)$, причем решение с двумя указателями уже довольно близко к полному.

Шестая и седьмая подзадачи были расчитаны на некоторые частные решения. Условие на равенство высот всех слоев позволяет сократить высоты из формулы для P и просто считать его как среднее арифметическое всех p_i. Необходимость следить только за двумя значениями крепости позволяет легче проанализировать, что выгодно пить только из слоев крепости с определенной стороны от t_i по значению.

Для **восьмой подзадачи** необходима идея с двумя указателями из решения пятой. Пусть у нас есть две крепости t_i, t_j, при этом $t_i < t_j$, тогда ответ для крепкости t_i будет не меньше ответа для крепкости t_j.

Предподсчитаем ответы для всех $1 \leqslant t \leqslant P_0$ по убыванию и для всех $P_0 < t \leqslant 10^5$ по возрастанию симметрично. Дальше, как обычно, рассматриваем $t < P_0$. Пусть рассматриваемая сейчас крепость равна t, тогда

- будем перебирать выпиваемый слой сверху вниз;
- также будем поддерживать множество всех невыпитых слоев, которые остались выше, упорядоченное по p_i (например, с помощью `std::set`);
- как уже было отмечено выше, достаточно выпить все слои с крепостью $> t$, поэтому, встречая новый слой, добавим его в множество, а затем удалим из множества и выпьем все слои с $p_i > t$;
- если итоговая крепость оказывается $< t$, то мы нашли ответ для крепости t.

Предподсчитав ответ для всех значений крепости от 1 до 10^5, мы затем можем отвечать на запросы за $\mathcal{O}(1)$, получая время работы $\mathcal{O}(\max(p) \cdot \log n + q)$.

Для **полного решения** воспользуемся идеей предыдущей подзадачи, просто реализовав метод двух указателей не по всем возможным значениям крепости, а только по фигурирующим в запросах. То есть отсортируем все запросы по t_i и найдем ответ описанным алгоритмом, перебирая $t_i \leqslant P_0$ по убыванию и $t_i > P_0$ по возрастанию симметрично.