Домашнее задание по АиСД №2

Эмиль Гарипов M3138

2020-04-29

Задача №3

Предподсчет

Для начала необходимо сжать значения из входного массива. Далее опустим подробности того, какие значения были в массиве до этого, потому что получить это значение всегда можно за $\mathcal{O}(1)$, предварительно сохранив его.

По заданному массиву a построим массив b из нулей и единичек. На i-ой позиции будет стоять единичка, если i — первое вхождение a_i в массив a. По полученному массиву построим дерево отрезков на сумму. Так же будем хранить массив $next$, $next_i$ указывает на следующее вхождение a_i в массив a(такой массив можно посчитать за $\mathcal{O}(n)$ простым проходом по массиву a). Теперь мы умеем отвечать на запрос количества различных чисел на отрезке [1; r], для этого надо просто запросить сумму в дереве отрезков на этом отрезке(ведь кол-во единиц равно количству различных элементов на отрезке, так как единички стоят только в индексах первых вхождений каждого значение val, которое встречается на отрезке).

Чтобы отвечать на запросы на отрезке [2; r] преобразуем массив b. Просто поставим единичку в $next_1$ и поставим нолик в b_1. Построим по полученному массиву дерево отрезков на сумму. Тогда ответом на запрос кол-ва различных чисел на отрезке $[2; r]$ будет сумма на этом отрезке $[2; r]$ в полученном массиве b. Эту сумму посчитает дерево отрезков. Единички здесь так же стоят только в индексах первых вхождений(по построению).

Построение персистентного дерева отрезков

Но чтобы не строить второе дерево отрезков воспользуемся персистентным деревом отрезков. Как видно, при переходе от отрезков [1; r] к отрезкам [2; r] мы изменили в массиве b не более двух значений. Следовательно в дереве отрезков обновилось не более двух «веток»(путей от листов до корня). Поэтому выстроим эти новые две ветки, а ссылки на все оставшиеся элементы дерева отрезков возьмем из первого дерева отрезков(так как они не поменялись) при построении этих новых веток. И сохраним отдельно первую и вторую «версии» полученных деревьев.

Далее построим версии деревьев отрезков для всех $l \in [3; n]$ аналогичным способом и сохраним ссылки на эти версии в массив t. Таким образом на i-ой итерации в полученном массиве b единички будут стоять на индексах первых вхождений(считая с индекса i) элементов. Значит для ответа на запрос на отрезке $[i; r]$ надо в i-ой версии дерева отрезков посчитать сумму на отрезке $[i; r]$.

Итого, в предподсчете мы для каждого индекса i изменяем имеющийся массив b и соответственно получаем новую версию дерева отрезков. Это выполняется за $\mathcal{O}(n \log_2(n))$.

Кол-во различных элементов на отрезке

Для ответа на запрос на отрезке $[l; r]$ возьмем l-ю версию дерева отрезков($\mathcal{O}(1)$). В этом дереве запросим сумму элементов на отрезке $[l; r]$ ($\mathcal{O}(\log_2(n))$).

Время работы

Предподсчет выполняется за $\mathcal{O}(n \log_2 n)$, так построение новой ветки выполняется за ($\mathcal{O}(\log_2(n))$), а всего мы строим не более $2n$ новых веток. Ответы на запросы за $\mathcal{O}(\log_2 n)$.

Задача №4

Значение в вершине

Пусть в каждой вершине u изначально записано какое-то значение a_u. Переформулируем как получать значение в вершине. Будем считать, что актуальное значение переменной в вершине u равно сумме значений всех вершин в поддереве u(включая саму вершину u). То есть:

$$Value_u = \sum_{v \in T_u} Value_v$$

где T_u — множество вершин поддерева u.
По известным значениям a_u можно легко посчитать значения $Value_u$ простым обходом дерева.

Предподсчет

Выпишем Эйлеров обход дерева в массив, назовем его $path$, пар(первое значение — номер вершины u, второе — $Value_u$), запустив обход дерева из корня. Каждую вершину запишем в массив дважды, при спуске в нее и при выходе. Так же сохраним для каждой вершины u два значения : $last_u$ и $first_u$ — первое и последнее вхождения вершины u в этот массив обхода. Заметим, что вершины, лежащие между $first_u$ и $last_u$ — вершины поддерева u, причем каждая встречается там два раза. А уменьшенная вдвое сумма вторых значений элементов массива $path$ лежащих между $first_u$ и $last_u$ равна $value_u$:

$$Value_u = \frac{\sum_{i=first_u}^{last_u} path_i.second}{2} \quad (1)$$

Пользуясь этим и будем вычислять значения переменных в вершине. Построим дерево отрезков на массиве $path$ на сумму вторых значений. Ответ на запрос суммы на отрезке и изменения элемента будет выполняться за $\mathcal{O}(\log_2 n)$, так как размер массива $path - 2n$.
Так же предподсчитаем массив двочных подъемов за $\mathcal{O}(n \log_2 n)$ для вычисления lca, которое нам падобиться позже.

Прибавление на пути

Сведем прибавление значения x на пути между двумя вершинами u и v к прибавлению в точках. Чтобы прибавить на пути надо сделать всего лишь 4 прибавления: прибавить x в $Value_u$ и $Value_v$, вычесть x из $Value_{lca(u,v)}$ и $Value_{p(lca(u,v))}$, где $lca(u, v)$ — наименьший общий предок вершин u и v, а $p(lca(u, v))$ - родитель наименьшего предка вершин u и v. При таком прибавлении актуальные значения (которые считаются по описанному вначале «правилу») вершин, не лежащих на пути между u и v не поменялось, а значение вершин лежащих на этом пути увеличилось на x.

Ответы на запросы

Построим дерево отрезков на массиве суффиксных сумм.

I Запрос прибавления на пути

Воспользуемся сведением прибавления значения x на пути между двумя вершинами u и v к прибавлению в точках, просто запросив в дереве отрезков прибавление значения x в точки $first_u$, $last_u$, $first_v$, $last_v$ и прибавление $-x$ в точки $first_{lca(u,v)}$, $last_{lca(u,v)}$, $first_{p(lca(u,v))}$, $last_{p(lca(u,v))}$. Вычисление lca выполняется за $\mathcal{O}(\log_2 n)$, запросы в дереве отрезков на прибавление в сумме тоже работают за $\mathcal{O}(\log_2 n)$.

II Запрос вычисления значения переменной

Пусть нас просят найти значение вершины u. Тогда по формуле (1) надо просто запросить в дереве отрезков сумму на отрезке $[first_u; last_u]$ и уменишить ее вдвое. Значения $first_u$ и $last_u$ получаем за $\mathcal{O}(1)$, так как сохранили их заранее, а запрос суммы в дереве отрезков работает за $\mathcal{O}(\log_2 n)$.

Время работы

Предподсчет выполняется за $\mathcal{O}(n \log_2 n)$, ответы на запросы за $\mathcal{O}(\log_2 n)$.