Задача 1. Дан *отсортированный* массив длины n. Ответить в онлайн на q запросов за $\mathcal{O}((n+q)\log n)$, каждый из которых бывает двух типов:

1. Сделать $a_i = \min(a_i, x)$ для всех $i \le r$.
2. Найти сумму на отрезке $[l; r]$.

Разбор. Заметим, что при применении таких операций изменения массив остается отсортированным. Каждую такую операцию можно заменить на присвоение на некотором отрезке. А именно, можно с помощью спуска по дереву найти первую позицию j такую, что $a_j > x$, после чего сделать $a_t = x$ для $t = j \dots r$. Такие операции легко выполняются с помощью дерева отрезков.

Задача 2. Дан квадрат с крайними точками в $(0;0)$ и в $(C;C)$, а также n точек внутри него. Найти максимальный квадрат, находящийся внутри этого квадрата, в котором не лежит ни одна из данных точек.

a. $\mathcal{O}(n \log n \log C)$;

b. $\mathcal{O}(n \log n)$.

Разбор.

a. Сделаем бинпоиск по ответу. Пусть хотим проверить, что ответ не меньше, чем a. Сузим квадрат снизу и слева на a, а каждую из точек внутри квадрата превратим в квадрат $a \times a$ с левым нижним углом в исходной точке. Тогда понятно, что ответ существует, если найдется точка внутри нового квадрата, не покрытая ни одним квадратом (эта точка будет соответствовать правому верхнему углу квадрата $a \times a$, который можно разместить внутри исходного). Такая проверка легко реализуется с помощью scanline с деревом отрезков.

b. Сделаем scanline, обрабатывая точки по неубыванию из x-координат. Будем поддерживать Y — (отсортированное) мультимножество y-координат точек, у которых $x \in [x_1; x_2]$ (изначально $x_1 = x_2 = -\infty$). Тогда понятно, что можно прорелаксировать ответ величиной $\min(x_2 - x_1, \Delta_y)$, где Δ_y — минимальная разность между соседними точками в Y. Далее, если $\Delta_y > x_2 - x_1$, следует перейти к следующему по возрастанию значению x_2. Если же $\Delta_y \le x_2 - x_1$, следует перейти к следующему по возрастанию значению x_1.

Задача 3. Дан массив из n чисел. Требуется в оффлайне ответить на q запросов вида «количество чисел на отрезке с l_i по r_i, значения которых находятся в отрезке от a_i до b_i».

a. $\mathcal{O}((n+q)\log^2 n)$;

b. $\mathcal{O}((n+q)\log n)$.

Разбор. Научимся отвечать на запрос количества чисел на префиксе $[1; r]$, значения которых лежат на отрезке $[a; b]$. Понятно, что каждый исходный запрос сводится к двум таким запросам: на префиксе $[1; r_i]$ и на префиксе $[1; l_i - 1]$.

Будем обрабатывать массив scanline слева-направо, поддерживая структуру данных, которая позволяет совершать прибавление в точке и вычислять сумму на отрезке. Тогда, когда рассматриваемый очередной префикс, заканчивающийся на позиции r, достаточно прибавить 1 в точке a_r. Для ответа на запрос на соответствующем префиксе, достаточно просто взять сумму на отрезке $[a; b]$. В качестве такой структуры данных удобно использовать дерево отрезков или дерево Фенвика.

Задача 4. Дан массив a из n целых чисел. Требуется в оффлайне ответить на q запросов вида «даны l и k, найдите минимальное r, такое что на отрезке с l по r содержится хотя бы k различных чисел» за время:

a. $\mathcal{O}((n+q)\log^2 n)$.

b. $\mathcal{O}((n+q)\log n)$.

Разбор. Для начала поймем, как решать задачу поиска количества различных на отрезке, далее модифицируем решение для этой задачи.

Сделаем scanline, обрабатывающий массив слева-направо. Также будем поддерживать структуру данных, позволяющую совершать прибавление в точке и вычислять сумму на отрезке. Будем в ней хранить 1 в позициях, где в нашем массиве стоит число, которое на текущем префиксе встречается в последний раз, и 0 в остальных позициях. Например, если наш массив равен `1 4 1 3 5`, и мы сейчас рассматриваемый префикс длины 4: `1 4 1 3`, то наша структура будет выглядеть так: `0 1 1 1 0`. Тогда понятно, что количество различных чисел на отрезках, заканчивающихся в той же точке, что и текущий рассматриваемый префикс, — это просто сумма на таком отрезке этой структуры. При приписывании к нашему префиксу нового числа нужно поставить 0 на месте предыдущего вхождения такого значения (если оно существует), и поставить 1 на новую позицию.

Для решения исходной задачи достаточно сделать такой же scanline, только идти в нем справа-налево. Затем необходимо просто найти в структуре k-ую единицу. Если использовать в качестве структуры дерево отрезков, то это делается спуском по дереву.

Задача 5. Дано подвешенное дерево из n вершин, рядом с каждой вершиной записано какое-то число. Требуется в оффлайне за $\mathcal{O}((n+q)\log n)$ ответить на q запросов вида «на сколько вверх нужно подняться от вершины v, чтобы встретить на пути хотя бы k различных чисел, записанных рядом с вершинами».

Разбор. Достаточно применить идею из предыдущей задачи и делать scanline не по массиву, а по дереву: обходить дерево с помощью DFS. Спуск вниз по дереву обрабатывается точно так же, как в и случае массива. Возврат наверх обрабатывается также достаточно просто.

Задача 6. Дан массив из n целых неотрицательных чисел, не превосходящих C. Ответьте в онлайне на q запросов вида «найти gcd чисел на отрезке от l до r, значения которых лежат на отрезке от a до b».

a. $\mathcal{O}(n\log n\log C + q(\log^2 n + \log C))$.

b. $\mathcal{O}(n\log^2 n + n\log n\log C + q(\log n + \log C))$.

Разбор. В этой задаче нужно построить merge sort tree.

a. В этом пункте внутри merge sort tree необходимо построить деревья отрезков по операции gcd.

b. В этом пункте внутри merge sort tree необходимо построить разреженные таблицы по операции gcd. Также необходимо использовать частичное каскадирование.

Задача 7. Дан массив длины n. Ответьте в онлайне на q запросов двух типов:

1. Найти k-ую порядковую статистику на отрезке от l до r.
2. Изменить значение массива в точке c_i. Числа не превосходят n.

Время:

a. $\mathcal{O}(n\log n + q\log^3 n)$;

b. $\mathcal{O}(n\log n + q\log^2 n)$.

Разбор.

a. В этом пункте достаточно поддерживать merge sort tree, в вершинах которого хранятся декартовы деревья. Тогда изменение очевидно, а ответ на запрос можно делать с помощью бинпоиска по ответу.

b. В этом пункте можно внутри merge sort tree поддерживать динамические (неявные) деревья отрезков на сумму: в x-м элементе внутреннего ДО, соответствующего отрезку $[l;r]$ внешнего ДО, хранится количество вхождений числа x на отрезке $[l;r]$. Операции изменения тривиальны.

Чтобы найти k-ую порядковую статистику, необходимо выписать $\mathcal{O}(\log n)$ деревьев отрезков, соотствующих отрезкам из ДО, на которые разбивается отрезок из запроса. Мы умеем искать для одного такого ДО k-ую порядковую статистику за $\mathcal{O}(\log n)$ с помощью спуска по дереву. В данном случае нам нужно сделать то же самое для «объединения» этих ДО: но ведь ничто не мешает нам спускаться одновременно по $\mathcal{O}(\log n)$ ДО.

Задача 8. Дано n прямоугольников со сторонами, паралелльными осям координат. Прямоугольники не пересекаются, но могут вкладываться. Вы можете перемещаться по плоскости, но, пересекая границы прямоугольника i, необходимо заплатить налог c_i. Необходимо в онлайне ответить на q запросов: «Вы начинаете в точке $(x_1;y_1)$, хотите попасть в точку $(x_2;y_2)$. Сколько вам придётся заплатить?». $\mathcal{O}((n+q)\log n)$.

Разбор. Заметим, что мы можем представить прямоугольники и связи между ними в виде графа, в котором вершины — прямоугольники, а ребра соединяют прямоугольники, между которыми можно непосредственно пересечь границу, притом веса ребра равен размеру налога. Легко заметить, что этот граф является деревом. Когда дерево построено, отвечать на запросы легко: достаточно просто найти длину пути в дереве, это делается с помощью LCA.

Для построения дерева воспользуемся scanline. Будем поддерживать ДО с операциями присвоения на отрезке и запроса значения элемента в точке. Будет три типа событий в scanline:

1. Открыть прямоугольник (левая граница)
2. Закрыть прямоугольник (правая граница)
3. Запрос, к какой вершине принадлежит точка (на каждый запрос о стоимости перемещения будет два таких события)

Scanline делаем по координате x.

При открытии прямоугольника достаточно сначала узнать, кто является его предком в дереве. Это просто значение в любой точке, соответствующей его отрезку по y. После этого необходимо присвоить номер этого прямоугольника на его отрезке y.

При закрытии прямоугольника необходимо на его отрезке y присвоить номер его предка.

При запросе, в какой вершине принадлежит точка, необходимо просто взять значение в соответствующей точке по y.

Задача 9. Дан массив a длины n, числа целые и по модулю не превосходят C. Ответить в онлайне на q запросов вида «найти подотрезок отрезка $[l; r]$ с максимальным значением произведения суммы чисел на нем и побитового ИЛИ модулей чисел на нем». $\mathcal{O}(n \log n \log C + q \log n \log^2 C)$.

Разбор. Воспользуемся идеей, похожей на решение стандартной задачи о поиске подотрезка максимальной суммы внутри отрезка.

Сделаем ДО, в вершинах которого будем хранить следующие значения:

- Ответ на задачу.
- В каком порядке и на каких индексах появляются новые биты, а также префикс максимальной суммы, конец которого лежит между соседними вхождениями новых битов, если рассматривать числа на отрезке слева-направо.
- Аналогично предыдущему пункту, но справа-налево.

Объединить два таких отрезка легко за $\mathcal{O}(\log^2 C)$: порядки, индексы и лучшие суммы легко поддерживаются за $\mathcal{O}(\log C)$, а ответ пересчитывается как максимум из двух ответов, а также перебора, до какого бита мы берем суффикс из левого отрезка, до какого бита мы берем префикс у правого отрезка.

Задача 10. Дан массив a длины n. Требуется в оффлайне за $\mathcal{O}((n+q) \log n \log q)$ ответить на q запросов двух типов:

1. Найти сумму элементов, больших нуля, на отрезке $[l; r]$.
2. Дана тройка (p, x, y). Вычесть из каждого элемента a_i значение $\max(0, x - y \cdot |i - p|)$.

Разбор. Заметим, что достаточно для каждой позиции понять, после какого запроса второго типа значение в ней станет отрицательным. После этого задача решается тривиальным scanline.

Также заметим, что очень просто выполнять операции второго типа + операции вида «узнать значение в точке». Достаточно просто на двух отрезках прибавить арифметические прогрессии, что можно достаточно просто сделать разными способами.

После этого можно прийти к следующему решению: сделаем параллельные бинпоиски по ответу для каждого ответа и с помощью решения задачи из предыдущего абзаца определим, отрицательный ли элемент для данного префикса операций.

Задача 11. Дан массив a длины n, все числа целые неотрицательные и не превосходят C. За $\mathcal{O}((n + q) \log n \log C)$ ответить в онлайн на q запросов, каждый из которых бывает трёх типов:

1. Изменить какое-то a_i.
2. Все числа на отрезке $[l_i; r_i]$ взять по модулю x_i.
3. Найти сумму на отрезке $[l; r]$.

Разбор. Будем поддерживать ДО на сумму и максимум. Изменение и запрос суммы делаются тривиально. Для запроса типа 2 будем действовать так: если мы зашли в вершину ДО, и максимальное значение на соответствующем ей отрезке меньше, чем x_i, то, очевидно, никакие элементы там не изменятся, и мы можем сразу выйти. В противном случае мы спускаемся дальше. Так мы можем спуститься до листьев ДО. Однако заметим, что хотя бы для половины из листьев, до которых мы спустились, значение элементов в них возьмется по модулю x_i (и уменьшится, так как там были значения больше или равные x_i). Однако, когда число после взятия по модулю уменьшается, оно уменьшается хотя бы в 2 раза. Следовательно, суммарно таких операций не может быть больше $\mathcal{O}((n+q) \log n)$. И, так как каждый спуск до листа занимает $\mathcal{O}(\log n)$, получаем итоговую асимптотику $\mathcal{O}((n + q) \log n \log C)$.