1 Обязательная часть

1. ***НОД.***

   Строим дерево отрезков. В вершинах храним НОД детей. В листьях храним сами, собственно, значения массива. Изменять значение и считать НОД на отрезке умеем а логарифм, это ведь ДО все-таки. Как умножить. Заметим, что $gcd(ax, bx) = x \cdot gcd(a, b)$. Ну и огонь. Делаем теперь обычное для ДО обновление на отрезке.

2. ***Произведение.***

   Опять ДО. Только тут теперь проблема со степенями: мы не сможем их за единицу считать. Поэтому когда приходит запрос на обновление - в какой-нибудь хэштаблицу сначала за логарифм записываем все возможные степени x (отрезок [l; r] ведь наишими вершинами не больше, чем на логарифм отрезков разобьется), а потом уже делаем всякие обновления.

3. ***Ближайший большой.***

   Пусть ищем ближайший слева. Храним в вершинах максимум на отрезке. Искать начинаем снизу: встаем в лист, который соответствует pos. Теперь если значение в вершине меньше x, то поднимаемся вверх одним из двух способов. Если мы были правым сыном — просто поднимаемся в отца (ведь значит наш отец отвечает только за отрезок слева от нас), если же мы были левым сыном, то поднимаемся в вершину слева от нашего отца на одном с ним уровне (это по индексам несложно за единицу найти). Теперь пусть мы встали в вершину, где значение не меньше x. Теперь надо спускаться. Если в правом сыне значение не меньше x — спускаемся в него, если же меньше — спускаемся в левого сына. Все происходит за две высоты дерева, то есть логарифм.

4. ***Сумма кубов.***

   Тут опять проблема с обновлением. Ее решить несложно, кроме суммы кубов храним в вершине еще и сумму квадратов и линейную. Сумма квадратов за единицу пересчитается через линейную, а сумма кубов за единицу через квадратную и линейную. Теперь все спокойной за логарифм делается.

5. ***Скобки.***

   В листьях закрывающая скобка — -1, открывающая — +1. В вершинах храним балансики на префиксах. Теперь отрезок [l; r] правильный, если баланс на префиксе l равен балансу на префиксе r и ни в каком месте внутри отрезка [l; r] баланс не становился меньше, чем в l. Если скобку меняем, то ко всему ее суффиксу нужно, соответственно, прибавить или вычесть 2 (делаем это, естественно, не за линию, а деревом отрезков, как всегда). Ну и все за логарифм происходит.

6. ***Котики.***

   Оффлайн. Храним в вершине последний момент, когда из какой-то клетки соответствующего отрезка ушел котик, и количество котиков на этом отрезке оставшееся. А затем, когда отвечаем на запросы, если время запроса меньше того момента, когда ушел последний котик, то отрезок не пустой. Если же последний котик ушел раньше нашего запроса, но там все еще остались котики в конце, то отрезок опять не пустой. Иначе он пустой, учитываем его в подходящих запросах.

Онлайн. Представим каждый отрезок из набора как логарифм вершин дерева отрезков. Точнее: каждой вершине дерева отрезков поставим ссылки на те отрезки из набора, которые они представляют. Также для каждого отрезка храним специальную переменную cnt, показывающей количество непустых представляющих вершин отрезка. В каждой вершине же ДО храним кол-во котиков. Теперь, когда котик уходит, мы встаем в соответствующий ему лист, меняем в нем 1 на 0, и начинаем подниматься к корню, попутно обновляя ДО. Если какая-то его вершина вдруг становится нулевой — переходим к отрезкам, на которые она ссылается, и уменьшаем там cnt. Если cnt становится нулевой - отрезок пуст. Вот и все, в онлайне можем все узнавать.

7. ***Прямоугольники.***

Сканлайн. Сначала пары координат точек сортируем с приоритетом по x слева направо. Затем бежим тем самым сканлайном. Если встречается точка (x, y) с весом w, то на отрезке [y; y+a] в момент x мы прибавляем w, а в момент $x + b$ вычитаем w. Как же нам быстро находить максимальный вес на отрезке, да и прибавлять на нем? Опять дерево отрезков. В вершинах храним максимумы.

8. ***Объединение отрезков.***

Идея: каждую вершину дерева отрезков разбиваем на логарифм вершин, из которых она состоит. Как это делаем. Берем [0; n) и разбиваем на [pos; $\frac{n}{2}$) и [$\frac{n}{2}$; pos). И теперь рекурсивно вызываемся. В чем прикол: теперь любой отрезок, который содержит $\frac{n}{2}$ мы можем выразить через два непересекающихся. Ну вот таким образом и считаем за $n \log n$. Любой отрезок в итоге сможем выразить через два непересекающихся.