Алгоритмы и Структуры Данных ДЗ-1

Гарипов Роман М3138

19.04.2020

Задача №1

Решение 1, без массовых операций

- В каждой вершине будем хранить ровно ту функцию от отрезка, которую нас просят посчитать :

$$f[L;R) = \sum_{i=1}^{R-L} (a_{L+i-1} \cdot i) = a_L + 2 \cdot a_{L+1} + \ldots + (R-L) \cdot a_{R-1}$$

  А также сумму на отрезке - sum и длину отрезка - len.

- Тогда, чтобы пересчитать значение в вершине через значения в детях достаточно сделать так :

$$f[v] = f[ls] + f[rs] + sum[rs] \cdot len[ls] = \sum_{i=1}^{M-L} (a_{L+i-1} \cdot i) + \sum_{i=1}^{R-M} (a_{M+i-1} \cdot i) + (\sum_{i=M}^{R} (a_i)) \cdot len[ls] = \sum_{i=1}^{R-L} (a_{L+i-1} \cdot i)$$

  где ls и rs - левый и правый сын соответственно.

- Изменять значение в точке очень просто, спустимся по дереву, изменим значение и поднимемся вверх, пересчитывая значения в вершинах.

Решение 2, с массовыми операциями

- Построим дерево отрезков на массиве префиксных сумм. Когда нас попросят изменить значение элемента, достаточно лишь сделать прибавление на суффиксе после этого элемента. Это делается групповыми операциями с проталкиваниями.

- Нам нужно считать такую функцию на отрезке :

$$f[L;R) = \sum_{i=1}^{R-L} (a_{L+i-1} \cdot i) = a_L + 2 \cdot a_{L+1} + \ldots + (R-L) \cdot a_{R-1} \tag{1}$$

  Но мы будем считать такую :

$$g[L;R) = \sum_{i=1}^{R-L} (a_{R-i+1} \cdot i) = a_{R-1} + 2 \cdot a_{R-2} + \ldots + (R-L) \cdot a_L \tag{2}$$

  Чтобы вычислить такую функцию выполним запрос $get\_sum[L;R)$

$$get\_sum[L,R) = a_L \cdot (R-L) + \ldots + a_{R-1} + \sum_{i=0}^{L-1} (R-L) \cdot a_i$$

  Видно что $get\_sum[L;R)$ отличается от $g[L;R)$ на $(a_0 + \ldots + a_{L-1}) \cdot (R-L)$.

  Поэтому чтобы вычислить $g[L;r)$, вычтем из $get\_sum[L;R)$ сумму на префиксе $[0;L-1]$ домноженную на $(R-L)$.

  Сумма на произвольном префиксе $[0;x)$ вычисляется легко, это просто значение в точке x дерева отрезков.

- Осталось только свести запрос вычисления g к вычислению f.

  Для этого просто будем работать с нашим массивом как с развернутым, используя вместо позиции (i) - позицию $(N - i)$.

  Запрос $set\ (id)\ (val)$ - перейдёт в $set\ (N - id)\ (val)$, точно так же для остальных запросов.

Задача №2

- Заранее сохраним все прямоугольники. Сожмём координаты по оси OY. То есть перенумеруем их числами от 0 до m, где $m \leq 2 \cdot n$, сохранив отношение порядка на новых номерах.

  Переприсвоим всем прямоугольникам новые y-координаты, но ещё сохраним старые в, чтобы потом можно было понять какой точке соответствует новая координата.

- Рассмотрим проекции прямоугольников на OX, это будут отрезки. Разобьём их на события : 1) отрезок открылся 2) отрезок закрылся. Отсортируем события по следующему правилу : раньше идет событие с меньшей x-координатой, при равенстве координат раньше должно идти событие открытия отрезка.

- Заведём дерево отрезков с групповыми операциями по сжатой оси OY, в каждой вершине будем хранить максимум и позицию в которой достигается максимум, а так же вспомогательную информацию для групповых операций $(push[v])$.

- Идем по отсортированным событиям оси OX. Встретили событие типа:

  (1) : открытие отрезка - выполним операцию $add\_on\_segment(rect[id].d, rect[id].u, +1)$

  (2) : закрытие отрезка - выполним операцию $add\_on\_segment(rect[id].d, rect[id].u, -1)$.

  Где $rect$ - массив прямоугольников, $rect[id].d, rect[id].u$ - координата самой нижней и самой верхней точек прямоугольника соотвественно.

  Получается, что мы идем сканирующей прямой по оси OX, каждый раз когда начинается какой-то прямоугольник, к отрезку его проекции на сжатую ось OY добавляем $+1$, когда прямоугольник кончается по OX, нужно вычесть 1.

  И чтобы посчитать ответ, надо перед каждым вычитанием взять максимум в дереве отрезков, а так же его позицию, чтобы можно было понять в какой точке достигается максимум. Поскольку координаты по y были сжаты, надо понять чему соответствует эта точка на несжатой оси, для этого можно в дереве отрезков хранить ещё номер прямоугольника для позиции в которой достигается максимум, а в прямоугольнике мы уже храним старую и новую y-координату. Или можно воспользоваться хэш-мапом, но это уже детали реализации.

Сжатие координат	$\mathcal{O}(nlogn)$
Сортировка событий	$\mathcal{O}(nlogn)$
Обработка всех событий	$\mathcal{O}(nlogn)$
Итого :	$\mathcal{O}(nlogn)$

Теперь заметим, что можно обойтись без групповых операций. Так как нам требуется только прибавлять на отрезке и брать максимум на отрезке. Такая задача была рассмотрена на практике.

- Абстрагируемся от нашей задачи, пусть у нас есть массив a, на котором надо делать прибавление на отрезке и брать максимум на отрезке. Обозначим за a' следующий массив : $a'[i] = a[i] - a[i-1]$, $a'[0] = a[0]$. Построим на нем дерево отрезков, в каждой вершине будем хранить сумму.

- Заметим, что сумма на префиксе в таком массиве соответствует значению в точке в исходном массиве. Сумму на префиксе будем вычислять деревом отрезков : $get\_sum(l, r)$ за $\mathcal{O}(log(n))$.

- В таком случае, прибавление на отрезке $[l; r]$ можно делать двумя прибавлениями в точке :

  $a'[l] += x$,
  $a'[r + 1] += (-x)$.
  Это делается за $\mathcal{O}(log(n))$

Так к каждому элементу начиная с l-ого будет прибавлен x, а для всех после r-го ничего не поменяется, ведь мы берем сумму на префиксе чтобы вычислить значение в точке.

- Для того чтобы искать максимум на отрезке $[l; r]$, будем в каждой вершине хранить где кончается префикс отрезка соответствующего этой вершине такой, что сумма на нем максимальна и сообственно сумму на этом префиксе.

  Пусть хотим объединить два отрезка : $[l; s]$ и $[s + 1; r]$, для каждого из которых это посчитано : $mx[ls]$ и $mx[rs]$, $sum[ls]$, $sum[rs]$, где mx - максимальный префикс, sum - сумма на нём, ls и rs - левый и правый сын соответсвенно.

$$mx[v] = \begin{cases} mx[rs] & \text{если } sum[ls] + sum[rs] \geq sum[ls] \\ mx[ls] & \text{иначе} \end{cases}$$

  Соответственно, примерно так же надо пересчитывать sum.

  Таким образом, если мы можем увеличить значение максимального префикса , то мы его увеличиваем.

  Теперь, когда приходит запрос максимума на отрезке $[l; r]$, будем делать так же как в обычном дереве отрезков вычисляем функцию на отрезке, только будем пересчитывать функцию от левого и правого отрезка по указанной выше формуле. Вычислив такую функцию от отрезка, останется только прибавить сумму на префиксе $[0; l]$ к сумме на префиксе отрезка $[l; r]$ с максимальной суммой.

$$\sum_{i=0}^{l} a'[i] + \sum_{i=l+1}^{mx} a'[i] = \sum_{i=0}^{mx} a'[i] = a[mx] = \max_{[l;r]}(a[i])$$

Задача №4

- Заметим, что вычисление заданной функции от массива равносильно нахождению максимальной суммы на суффиксе, среди всех равных - самой правой.

  Понятно что ответом будет сумма на некотором суффиксе, причем на таком, что слева от него при вычислении нашей функции получился ноль(кроме вырожденного случая, когда все числа положительные).

  $(\Rightarrow)$

  Покажем сначала, что для позиции, в которой достигается максимум сумм на суффиксе, справа от нее не будет нуля при вычислении исходной функции. Если бы там был ноль, то мы бы могли перенести начало этого суффикса правее, сумма на нем не стала бы меньше, что противоречит тому что наш суффикс максимальный, а среди всех максимальных - самый правый.

  $(\Leftarrow)$

  Теперь в другую сторону, покажем, что для суффикса с максимальной суммой слева от него всегда получается ноль при вычислении функции. Если бы это было не так, то мы могли бы взять элемент справа и увеличить сумму на суффиксе, но наш суффикс уже максимальный, поэтому не может быть такого, что справа не будет получатся ноль.

  Теперь понятно, что для вычисления функции от массива, достаточно найти максимальную суффиксную сумму.

- Будем строить дерево отрезков на массиве суффиксных сумм. В каждой вершине храним сумму на отрезке такого массива.

- Чтобы ответить на запрос функции от массива, надо взять максимум во всём дереве отрезков.

- Когда приходит запрос изменения элемента, надо прибавлять значение на префиксе.

  Итого : свели это к задаче прибавления на отрезке и нахождения максимума на отрезке, это можно делать как в Задаче 2 без массовых операций. Строим дерево отрезков на массиве разности соседних элементов и в каждой вершине храним сумму на отрезке в таком массиве. И выполняем все операции так, как это описано в решении Задачи 2.

  Получаем решение с ассимптотикой $\mathcal{O}(nlogn)$ без массовых операций.