МІРТ, ЗКШ, февраль 2015
Занятие группы B по дереву отрезков

Бабанин Иван, Дмитриев Алексей, Останин Александр, Смирнов Иван

Задача 1.

Дан массив длины n из целых положительных чисел не больше k. Требуется отвечать на запросы двух видов (всего запросов q):

- Присвоить на отрезке всем одинаковое значение.
- Узнать на отрезке количество различных чисел.

Решение.

Заведем дерево отрезков, в каждой вершине которого будет в сжатом виде (в битсете) храниться информация о том, какие из чисел $1, 2 \dots, k$ встречаются в данном поддереве.

- Присвоить t на отрезке просто: нужно очистить битсет, а после этого выставить в нем t-ый бит.
- Для объединения информации в двух вершинах нужно использовать операцию OR (по-битового или).
- Для ответа на запрос второго типа осталось лишь научиться по битсету определять количество выставленных в нем бит.

  Для этого воспользуемся предподсчетом. Для каждой маски разумного размера (на практике удобнее всего $d = 16$ бит) запомним кол-во единиц в ней, затем разделим битсет на маски такого размера и сложим результат.

  Итоговая асимптотика: $\mathcal{O}(2^d)$ на предподсчет, $\mathcal{O}(\frac{nk}{2d})$ на построение дерева отрезков, $\mathcal{O}(\frac{qk \log n}{2d})$. Необходимая память: $\mathcal{O}(\frac{nk}{2d})$.

Задача 2.

Дан массив длины n изначально заполненный нулями. Требуется отвечать на запросы двух видов:

- Найти границы самого левого отрезка из k элементов, такого что все его элементы равны 0, и присвоить всем элементам этого отрезка значение 1.
- Отменить действие одного из запросов предыдущего типа.

Решение.

Заведем дерево отрезков, в каждой вершине которого будет храниться длина максимального отрезка из 0 в поддереве вершины, количество 0 на префиксе и на суффиксе отрезка, которому соответствует данная вершина.

- Нетрудно присвоить 0 на отрезке с помощью техники отложенных операций при реализации дерева отрезков сверху. Если на всем отрезке, которому соответствует вершина, нужно присвоить 0, то изменим соответствующим образом информацию хранимую в вершине, отметим в вершине необходимость обновить информацию в детях вершины, если она не является листом. Если присвоить 0 нужно не на всем отрезке, то вызовем операцию присвоения на отрезке рекурсивно от потомков, обновим информацию в вершине.
- Воспользуемся спуском по дереву отрезков, чтобы найти самый левый непрерывный отрезок состоящий из не менее, чем k нулей. Пусть мы находимся в некоторой вершине, в которой есть отрезок, состоящий из не менее, чем k нулей (в вершине хранится длина максимального отрезка из 0). Тогда возможно три случая: искомый отрезок находится полностью в отрезке, соответствующем левому потомку вершины, искомый отрезок находится полностью в отрезке, соответствующем правому потомку вершины, или искомый отрезок имеет непустое пересечение с отрезком соответствующем левому потомку и также с отрезком соответсвующем правому потомку. Эти случаи могут быть разобраны с помощью информации хранимой в детях текущей вершины.

Итоговая асимптотика выполнения каждой операции: $\mathcal{O}(\log n)$

Задача 3.

Дан массив a из элементов от 0 до 2009. Над массивом проделывают следующие операции:

- Выбрать числа l, r $(1 \le l \le r \le n)$ и для всех $i = l, l+1, \dots r$ выполнить $a_i = a_i^2 \mod 2010$.
- Посчитать сумму на отрезке массива от l до r.

Наблюдения (при первом прочтении можно ограничиться выводом, полученным в последнем предложении)

Посмотрим, что происходит с каждым элементом массива при наших операциях. А именно, рассмотрим последовательность

$$a_m = a^{2^m} \mod 2010.$$

Элементы последовательности принимают лишь конечное число значений, $a_{m+1} = a_m^2 \mod 2010 \Rightarrow$ последовательность $\{a_m\}$ зацикливается. Давайте найдём её период.

Чтобы найти период последовательности $\{a_m\}$, в которой элементы берутся по модулю 2010, достаточно найти периоды той же последовательности, взятой по модулям максимальных степеней простых, входящих в 2010, и после этого найти их наименьшее общее кратное.

$$2010 = 2 \cdot 3 \cdot 5 \cdot 67.$$

Все простые в разложении 2010 входят в первой степени $\Rightarrow$ нужно научиться искать период последовательности, взятой по модулю простого p.

Чтобы найти период $\{a_m \mod p\}$, нужно найти такое минимальное T, что начиная с какого-то m

$$a_{m+T} \equiv a_m \pmod p,$$

тогда такое m будет предпериодом, а T — периодом последовательности.

Перепишем это условие:

$$a^{2^{m+T}} - a^{2^m} \equiv 0 \pmod p \Leftrightarrow a^{2^m} \cdot (a^{2^m(2^T-1)}) \equiv 0 \pmod p.$$

Условие на период:

$$a^{2^m} \cdot (a^{2^m(2^T-1)}) \equiv 0 \pmod p.$$

Если $a \equiv 0 \pmod p$, то это сравнение выполняется.

Иначе нам достаточно, чтобы

$$2^m \cdot (2^T - 1) \equiv 0 \pmod{p-1},$$

так как по малой теореме Ферма при $(a, p) = 1$ верно

$$a^{p-1} \equiv 1 \pmod p.$$

Условие на период:

$$2^m \cdot (2^T - 1) \equiv 0 \pmod{p-1}.$$

Для 2010 $p = 2, 3, 5, 67$, поэтому $(p-1) = 1, 2, 4, 66$. В таком случае достаточно взять $m = 2, T = 10$, и условие будет выполняться для всех четырех простых чисел.

Значит, последовательность $\{a_m\}$ имеет период 10 и предпериод 2.

Решение

Теперь будем строить дерево отрезков. Что будем хранить в вершине?

- Величина add сигнализирует о том, что каждый элемент в отрезке, за который отвечает вершина, нужно возвести в степень 2^{add}. Поддерживается так же, как в дереве отрезков с операцией добавления на отрезке: выполняя запрос первого типа, нужно для каждого отрезка, на которые разбивается запрос, сделать инкремент add в соответствующей вершине.

- Массив $sum[12]$. В ячейке $sum[i]$ будет храниться сумма чисел в отрезке вершины, если при этом каждый элемент отрезка нужно возвести в степень 2^i, иными словами, если при проталкивании вниз величины add в текущей вершине накопился $add = i$.

Как реализовать функции $push()$ и $recalc()$?

Функция $push()$ будет проталкивать вниз величину add, как в дереве отрезков с добавлением на отрезке.

Функция $recalc()$ должна учитывать еще не учтенные в детях вершины величины add. Поэтому пересчет будет выглядеть так:

$$t[v][i] = t[2 \cdot v][getInd(i + add[2 \cdot v])] + t[2 \cdot v + 1][getInd(i + add[2 \cdot v + 1])],$$

где функция $getInd(i)$ по индексу i возвращает позицию в периоде или предпериоде последовательности $\{a_m\}$, в которой стоит число, равное a_i.

Если $i \leq 12$, то $getInd(i)$ возвращает i, иначе она возвращает $((i-2) \mod 10) + 2$.

Таким образом, мы описали структуру дерева отрезков и реализацию функций $push()$ и $recalc()$. Чтобы ответить на запрос, нужно сложить все величины $t[v][getInd(add[v])]$ для вершин v, которые соответствуют одному из отрезков, на которые разбивается запрос.

Итоговая асимптотика выполнения каждой операции: $\mathcal{O}(\log n)$

Задача 4.

На прямой расположены бомбы. Каждая бомба задается координатой и радиусом действия. При подрыве бомбы взрываются все бомбы в радиусе действия. Требуется для каждой бомбы определить, сколько бомб взорвется при ее прямом подрыве.

Решение

Заметим, что бомбы, взрывающиеся при подрыве данной, образуют отрезок. Проведем ребро между двумя бомбами, если вторая подрывается при взрыве первой, и построим конденсацию этого графа. После этого на ациклическом графе считаем динамику «самая левая и самая правая бомба, которая взорвется при подрыве этой компоненты связности».

Как строить конденсацию за $\mathcal{O}(n \log n)$? Построим фиктивное дерево отрезков на последовательности бомб. Вершины дерева отрезков также будут вершинами графа, который мы конденсируем. Проведем ребра между вершиной дерева отрезков и ее сыновьями. Пусть бомба x накрывает отрезок бомб $[l, r]$. Вместо $r - l + 1$ ребер добавим $\mathcal{O}(\log n)$ в те вершины дерева отрезков, которые соответствуют отрезку $[l, r]$. Можно заметить, что конденсация этого графа совпадает с требуемой конденсацией, если не учитывать добавленные фиктивные вершины.

Задача 5.

Дан выпуклый многоугольник из n вершин. Все вершины пронумерованы против часовой стрелки в порядке заданном во входном файле от 1 до n. После совершения запросов номера вершин не изменяются.

Требуется ответить на q запросов следующего вида:

- удалить вершину с номером i
- добавить ранее удаленную вершину с номером i
- посчитать площадь многоугольника состоящего из неудаленных вершин, которые будут получены при обходе исходного многоугольника с вершины i до вершины j против часовой стрелки.

Решение

Один из способов вычислить удвоенную площадь выпуклого многоугольника: сложить ориентированные площади трапеций, образованных вершинами i и $i+1$, считая, что вершина $n+1$ равна вершине с номером 1.

Формула: $S = \left|\sum_{i=1}^{n}(x_{i+1} - x_i) \cdot (y_i + y_{i+1})\right|$

Подобный подход применим и в данной задаче.

Построим дерево отрезков с операцией суммы. В элементе i, если вершина i не удалена, должна быть площадь ориентированной трапеции, образованной вершиной i и следующей неудаленной вершиной в порядке против часовой стрелки, иначе элемент i должен равняться 0.

Как ответить на запрос получения площади?

Пусть $i < j$. Тогда ответ это сумма элементов с i по j в дереве отрезков плюс площадь трапеции образованной вершинами j и i.

Если $i > j$, то получим ответ для пары вершин j и i и вычтем его из площади многоугольника, образованного всеми неудаленными вершинами.

Как поддерживать дерево отрезков после операций добавления и удаления?

Будем поддерживать два std::set с неудаленными вершинами. В одном будет порядок по увеличению номеров вершин, в другом - по уменьшению. Если вы пишете на языке *Pascal*, то вам потребуется использовать спуск по дереву в дополнительных деревьях отрезков, в которых будут храниться неудаленные вершины.

Запрос удаления вершины i

Присвоим элементу i в дереве отрезков 0. Найдем с помощью хранимых setов предыдущую(j_1) и следующую(j_2) неудаленные вершины после вершины i в порядке против часовой стрелки. Обновим значения элемента j_1.

Запрос добавления вершины i

Найдем предыдущую(j_1) и следующую (j_2) неудаленные вершины после вершины i в порядке против часовой стрелки. Обновим значения элементов j_1 и i.

Итоговая асимптотика решения $\mathcal{O}((q+n)\log n)$.