Древовидные структуры

В этом разделе мы представим две древовидные структуры, позволяющие обрабатывать запросы по диапазону и изменять значения элементов массива за логарифмическое время. Сначала обсудим двоичные индексные деревья, поддерживающие запросы о сумме, а затем - деревья отрезков, поддерживающие также запросы других видов.

Двоичные индексные деревья

Двоичное индексное дерево (или дерево Фенвика) можно рассматривать как динамический вариант массива префиксных сумм. Оно предоставляет две операции с временной сложностью O(log n): обработка запроса о сумме по диапазону и изменение значения. Хотя в названии этой структуры данных фигурирует слово «дерево», обычно она представляется в виде массива. При обсуждении двоичных индексных деревьев мы будем предполагать, что все массивы индексируются, начиная с 1, потому что это упрощает реализацию структуры.

Обозначим p(k) наибольшую степень двойки, делящую k. Двоичное индексное дерево хранится в массиве tree таким образом, что



т. е. элемент в позиции k содержит сумму по заканчивающемуся в этой позиции диапазону длины p(k). Например, поскольку р(6) = 2, tree[6] содержит значение sumq(5,6). На рис. 1 показаны массив и соответствующее ему двоичное индексное дерево. На рис. 2 показано соответствие между значениями двоичного индексного дерева и диапазонами исходного массива.



Рис. 1. Массив и его двоичное индексное дерево

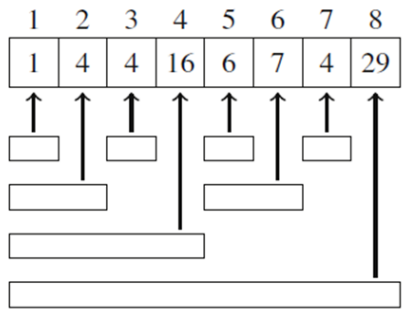


Рис. 2. Соответствие между двоичным индексным деревом и диапазонами

А чтобы вычислить значение sumq(a, b), где а > 1, мы применим тот же прием, что в случае массивов префиксных сумм:

sumq(a, b) = sumq(1, b) - sumq(1, a - 1).

И sumq(1, b), и sumq( 1, a - 1) можно вычислить за время O(log n), поэтому полная временная сложность равна O(log n).

После изменения любого элемента массива необходимо обновить двоичное индексное дерево. Например, если изменяется элемент 3, то следует обновить суммы по диапазонам [3, 3], [1,4] и [1, 8] (рис. 4). Поскольку каждый элемент массива принадлежит O(log n) таким диапазонам, то достаточно обновить O(log n) элементов дерева.

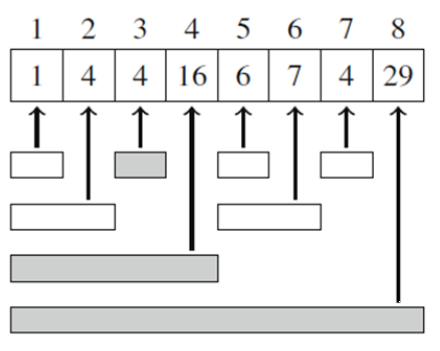


Рис. 4. Обновление значения в двоичном индексном дереве

Реализация. Операции двоичного индексного дерева эффективно реализуются с помощью поразрядных операций. Нам понадобится следующий факт: значение р(к) можно вычислить по формуле:

p(k) = k & -k,

которая выделяет самый младший единичный бит k.

Следующая функция вычисляет sumq(1, k):

int sum(int k) {

int s = 0;

while (k >= 1) {

s += tree[k];

k -= k & -k;

}

return s;

}

А эта функция увеличивает значение k-го элемента массива на х (x может иметь любой знак):

void add(int k, int x) {

while (k <= n) {

tree[k] += x; k += k& -k;

}

}

Временная сложность обеих функций равна O(log n), потому что они обращаются к O(log n) элементам двоичного индексного дерева, а переход к следующей позиции занимает время O(1).