Конспект лекции про дерево отрезков

Матвей Кориненко

Октябрь 2021

1 Решения статичной задачи

Давайте рассмотрим такую задачу. Имеется массив a, состоящий из n целых чисел. Нам поступают запросы вида L, R, на которые нам нужно выводить сумму элементов на отрезке массива a от L до R. Как мы можем решать эту задачу?

- Считать ответ для каждого запроса с нуля (сложность $O(qn)$)
- Предпосчитать массив префиксных суммм, а потом отвечать на запросы за $O(1)$. (общая сложность алгоритма $O(n+q)$)

  Массив префиксных суммм — массив, в котором на i-том месте стоит число, равное сумме чисел обычного массива, стоящих на индексах от 0 до $i-1$.

  Если массивом a является массив

1	2	3	4	-4

  то его массивом префиксных сумм будет массив

0	1	3	6	10	6

  Посчитать этот массив можно за линейное время.

2 Проблема динамической задачи

А теперь нам еще будут поступать запросы вида i x, что означает — на i-тое место массива a поставить элемент x. Как мы можем изменить наши предыдущие решения

- Просто за $O(1)$ будем менять число в массиве. Общая сложность не изменится — $O(qn)$
- При каждом изменении, нужно будет пересчитывать массив префиксных сумм за $O(n)$. Общая сложность будет $O(qn)$.

Можно ли эту задачу решить оптимальнее? Оказывается, да.

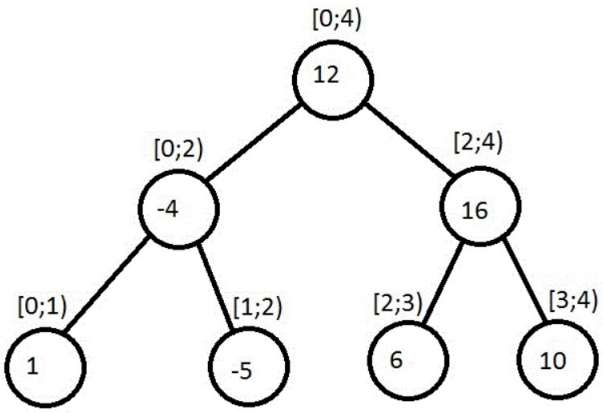
3 Дерево отрезков

Эту структуру можно представить в виде $2*n$ отрезков над массивом размера n. За каждый из этих отрезков отвечает число, которое равно функции на этом отрезке (в нашем случае сумме). И получилось это число благодаря дочерним отрезкам как сумма значений этих отрезков.

Иными словами, значение в вершине дерева отрезков равно значению функции от ее дочерних вершин. А значениями в вершинах самого нижнего уровня (у которых нет дочерних вершин) являются значения начального массива.

Построение этой структуры по готовому массиву занимает $O(n)$ времени (через каждую вершину мы спускаемся не более двух раз), а изменение одного элемента или получение значения функции на отрезке — $O(\log n)$. Потому что всего уровней в дереве $\log_2 n$ и для выполнения запроса на отрезке достаточно спуститься на несколько уровней.

Каждая вершина этого дерева отвечает не за отрезок, а за полуинтервал. Поверьте, в дальнейшем использование полуинтервалов Вам еще не раз встретится и пригодится, хоть сейчас это и покажется неудобным.



Дерево отрезков реализуется на обычном массиве. Функционально оно занимает $2 * n$ вершин, но в некоторых случаях $2 * n$ индексов может не хватить, например, когда размер изначального массива нечетный. Чтобы с этим постоянно не заморачиваться, достаточно просто поставить $4n$, в стандартные ограничения по памяти это помещается.

Первая вершина дерева стоит под индексом 1 (для удобства), тогда две дочерних будут под индексами $2 * 1$ и $2 * 1 + 1$ (чтобы оптимально использовать память). Аналогично, любая вершина с номером i имеет две дочерние вершины, отвечающие за левую часть отрезка и за правую, с номерами $2i$ и $2i + 1$.

Может так получиться, как в случае с $n = 5$, что на одном уровне будут отрезки разных длин — 2 и 3, но это лишь изменяет высоту поддеревьев двух этих вершин и никак негативно на выполнение запросов не влияет.

4 Реализация ДО для поиска суммы на отрезке

```
a, tree, n = ...

void build(int v = 1, int l = 0, int r = n) {
    if (r - l == 1) {
        tree[v] = a[l];
        return;
    }
    build(v * 2, l, (l + r) / 2);
    build(v * 2 + 1, (l + r) / 2, r);
    tree[v] = tree[v * 2] + tree[v * 2 + 1];
}

void update(int pos, long long val, int v = 1, int l = 0, int r = n) {
    if (r - l == 1) {
        tree[v] = val;
        return;
    }
    int m = (l + r) / 2;
    if (pos < m)
        update(pos, val, v * 2, l, m);
     else
        update(pos, val, v * 2 + 1, m, r);
    tree[v] = tree[v * 2] + tree[v * 2 + 1];
}

long long get(int ql, int qr, int v = 1, int l = 0, int r = n) {
    if ((qr <= l) || (ql >= r))
        return 0;
    if (ql <= l && r <= qr)
        return tree[v];
    return get(ql, qr,v * 2, l, (l + r) / 2) +
           get(ql, qr, v * 2 + 1, (l + r) / 2, r);
}
```