# День 1. Запросы на отрезке.

4 января, 2024

## 1 Введение

Сегодня мы поговорим о довольно большом и расплывчатом классе задач, который я буду называть “запросы на отрезке”. Вообще говоря, общего у алгоритмов что мы сегодня обсудим будет только название, но они решают очень похожие, на первый взгляд задачи.
План теоретической части.

(i) Преффиксные суммы.
(ii) Оффлайн алгоритмы над преффискными суммами.
(iii) Дерево отрезков
(iv) Оффлайновые модификации дерева отрезков.

## 2 Преффиксные суммы.

**Definition 2.1.** (Префиксные суммы): *Префиксными суммами массива* $A = [A_0, A_1, A_2, ..., A_{n-1}]$ *называется массив* $A = [S_0, S_1, S_2, ..., S_n]$, *определенный следующим образом:*

$$S_0 = 0$$

$$S_1 = A_0$$

$$S_2 = A_0 + A_1$$

$$S_3 = A_0 + A_1 + A_2$$

$$...$$

$$S_n = A_0 + A_1 + ... + A_{n-1}$$

*(i) Обратите внимание, что в такой индексации*

*(i)* $S_k$ *равен сумме первых* $k$ *элементов массива* $aa$ *не включая* $A_k$,
*(ii) длина* $S$ *на единицу больше длины* $aa$,
*(iii)* $S_0$ *всегда равен нулю.*

*Задача 1.* Дан массив целых чисел (0-индексация), и приходят запросы вида «найти сумму на отрезке $[l; r]$. Нужно отвечать на запросы за $O(1)$.
*Решение.* Возьмём разность $S_{r+1} - S_l$
*Задача 2.* Найти подотрезок нулевой/максимальной суммы. Ассимптотика решение $O(n \log n)$ и $O(n)$ соответственно.

(i) Сумма 0. По задаче 1, поиск отрезка $[l; r]$ с суммой эквивалентен задаче поиска двух одинаковых чисел $S_l, S_{r+1}$ в массиве префиксных сумм.

(ii) Максимальная сумма. Давайте решать задачу методом двух указателей и хранить минимальный элемент массива префиксных сумм на префиксе, ведь чтобы получить максимум, нужно отнять минимум!

*Задача 3.* Дана таблица $n * n$ заполненная числами. Научиться отвечать на запросы суммы на подматрице за $O(1)$ с $O(n^2)$ предподсчёта.
Построим аналогичные одномерным двумерные преффиксные функции.

## 2.1 Обобщение преффиксных сумм для любой операции.

Понятно, что операция "+" хорошая и позволяет без проблем создавать и использовать на ней массив префиксных сумм, но что если захотелось использовать другие операции? Тогда, чтобы отвечать на запросы на отрезке наша операция должна быть обратима и ассоциативна (вообще тяжело придумать пример неассоциативной функции над которой хочется считать префиксные суммы (деление?)).

$\left((1:2):2=\frac{1}{4}, 1:(2:2)=1\right)$
Примеры.

(i) Префиксные суммы по операции $\oplus$ можно без ограничений, обратная к нему операция это сам $\oplus$

(ii) Для минимума или максимума возможно считать только префиксные или суфиксные функции, так как у этих операций нет обратных.

(iii) С умножением та же история, если мы позволяем себе домножать на 0, то мы больше не имеем обратной операции к умножению.

.

## 2.2 Офлайновые алгоритмы с префиксными суммами

**Definition 2.2.** (Разностный массив.): *Разностным массивом массива* $[B_0, B_1, ..., B_{n-1}]$ *называется массив* $[A_0, A_1, ..., A_{n-2}]$, *определяющийся следующим образом:*

$$A_0 = B_1 - B_0$$

$$A_1 = B_2 - B_1$$

$$A_2 = B_3 - B_2$$

$$...$$

$$A_{n-3} = B_{n-2} - B_{n-3}$$

$$A_{n-2} = B_{n-1} - B_{n-2}$$

Очевидно, что если $B$ массив префиксных сумм массива a, то массив $A$ разност- ный массив массива b, потому что формула $A_i = B_{i+1} - B_i$ это просто преобразованная рекуррентная формула для поиска префиксных сумм: $B_{i+1} = B_i + A_i$. Однако разностный массив может помочь нам даже там, где массивом префиксных сумм и не пахнет! Также обратите внимание, что если для подсчета массива префиксных сумм была нужна рекуррентная формула, то каждый член разностного массива зависит всего от двух элементов исходного, так что можно пользоваться формулами из определения для подсчета разностного массива за $O(n)$.
*Задача 4* Даны запросы прибавления константы на отрезке, нужно вывести массив после всех преобразований. $O(n+q)$
На разностном массиве прибавление на отрезке вырождается в прибалвение в двух точках.

А раз уж у нас есть формула по которой мы можем однозначно восстановить сам массив из разностного, ничего не мешает сделать все преобразования на разностном.
*Задача 5* То же самое, но прибавлять будем арифметическую прогрессию.
Теперь давайте дважды насчитаем разностный массив и будем делать прибавления уже на нем, когда мы восстановим исходный массив получится то что нужно.

# 3 Дерево отрезков.

Дерево отрезков — очень мощная и гибкая структура данных, позволяющая быстро отвечать на самые разные запросы на отрезках.

Рассмотрим конкретную задачу: дан массив aa из nn целых чисел, и требуется отвечать на запросы двух типов:

(i) Изменить значение в ячейке (т. е. реагировать на присвоение $a[k] = x$).
(ii) Вывести сумму элементов $a_i$ на отрезке с $l$ по $r$.

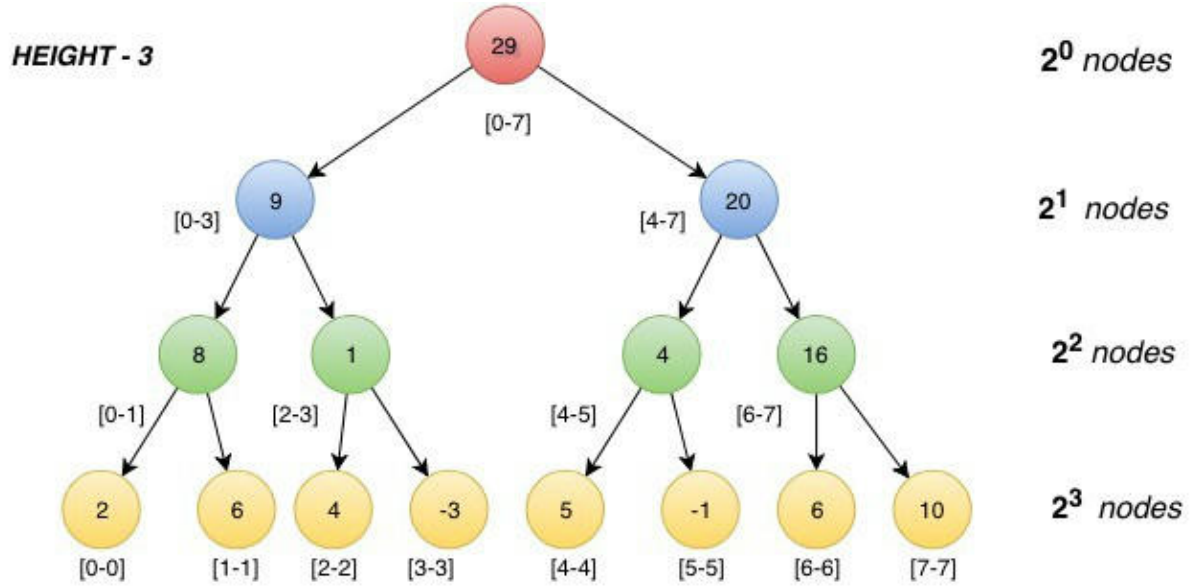
Оба запроса нужно обрабатывать за время $O(\log n)$.

## 3.1 Структура дерева отрезков

Чтобы решить задачу, сделаем с исходным массивом следующие манипуляции.

Посчитаем сумму всего массива и где-нибудь запишем. Потом разделим его пополам, посчитаем сумму на половинах и тоже где-нибудь запишем. Каждую половину потом разделим пополам ещё раз, и так далее, пока не придём к отрезкам длины 1.

Эту последовательность разбиений можно представить в виде дерева.




==Разные полезные свойства

Высота дерева отрезков равна $\Theta(\log n)$: на каждом новом уровне длина отрезка уменьшается вдвое. Этот факт будет ключевым для оценки асимптотики операций.

Любой полуинтервал разбивается на $O(\log n)$ неперекрывающихся полуинтервалов, соответствующих в вершинам дерева: с каждого уровня нам достаточно не более двух отрезков.

Дерево содержит менее 2n2n вершин: первый уровень дерева отрезков содержит одну вершину (корень), второй уровень — в худшем случае две вершины, на третьем уровне в худшем случае будет четыре вершины, и так далее, пока число вершин не достигнет nn. Таким образом, число вершин в худшем случае оценивается суммой $n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + ... + 1 < 2n$. Значит, оно линейное по памяти.

При $n$, отличных от степеней двойки, не все уровни дерева отрезков будут полностью заполнены. Например, при $n = 3$ левый сын корня есть отрезок $[0, 2)$, имеющий двух потомков, в то время как правый сын корня — отрезок $[2, 3)$, являющийся листом.

## 3.2 Пробуем отвечать на запросы

### 3.2.1 Запрос обновления. Нам нужно обновить значения в вершинах таким образом, чтобы они соответствовали новому значению $a[k] = x$.

Изменим все вершины, в суммах которых участвует $k$-тый элемент. Их будет $\Theta(\log n)$ — по одной с каждого уровня.

Это можно реализовать как рекурсивную функцию: ей передаётся текущая вершина дерева отрезков, и функция выполняет рекурсивный вызов от одного из двух своих сыновей (от того, который содержит $k$-ый элемент в своём отрезке), а после этого — пересчитывает значение суммы в текущей вершине точно таким же образом, как мы это делали при построении дерева отрезков.

### 3.2.2 Запрос суммы. Мы знаем, что во всех вершинах лежат корректные значения, и нам с помощью них посчитать сумму на отрезке.

Сделаем тоже рекурсивную функцию, рассмотрев три случая:

Если отрезок вершины лежит целиком в отрезке запроса, то вернуть записанную в ней сумму. Если отрезки вершины и запроса не пересекаются, то вернуть 0. Иначе разделиться рекурсивно на 2 половины и вернуть сумму этой функции от обоих детей.

## 3.3 Решаем опорные задачи.

1) Спуск в ДО
2) $k$-тый ноль
3) Количество элементов со значениями из отрезка на отрезке массива

## 3.4 Реализация

1) Почему полуинтервалы?

2) Почему дополняем до степени двойки

3) Построение за $O(n)$

4) Обновление снизу

## 3.5 Решаем опорные задачи 2.

1) Ленивое проталкивание
2) Объединение прямоугольников