Дерево отрезков

Лесников Юрий, ceagest

1 Дерево отрезков

- Мы уже умеем отвечать на запросы вида: «найти сумму на отрезке», например, с помощью префиксных сумм.
- Хотим решать задачу «обновить значение элемента массива».
- Нам может помочь структура «Дерево отрезков».

Лемма 1.1. *Пусть $\mathbb{X}$ – абстрактное множество. Дерево отрезков способно за $\mathcal{O}(\log n)$ получать на подотрезке результат любой операции $\mathcal{B}\colon \mathbb{X} \times \mathbb{X} \to \mathbb{X}$, которая:*

1. *Ассоциативна:* $\forall a, b, c \in \mathbb{X} \quad \hookrightarrow \quad \mathcal{B}(\mathcal{B}(a, b), c) = \mathcal{B}(a, \mathcal{B}(b, c))$
2. *Коммутативна:* $\forall a, b \in \mathbb{X} \quad \hookrightarrow \quad \mathcal{B}(a, b) = \mathcal{B}(b, a)$
3. *Имеет нейтральный элемент:* $\exists 0 \in \mathbb{X} : \forall a \in \mathbb{X} \quad \hookrightarrow \quad \mathcal{B}(a, 0) = \mathcal{B}(0, a) = a$

Замечание 1.1. *Задачи **RSQ**/**RMQ**(Range sum query/range minitimum query), как правило, решаются с использованием этой структуры данных.*

1.1 Структура

Рассмотрим для суммы:

- Корень содержит сумму всего массива.
- Левый ребенок – сумму первой половины элементов.
- Правый ребенок – сумму второй половины.
- Далее аналогично бьем пополам детей.

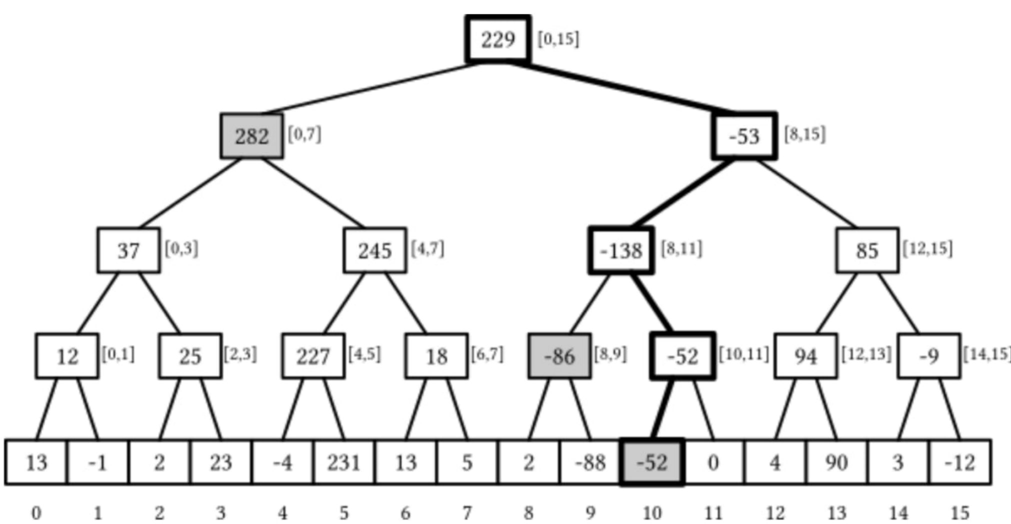

Рис. 1: Пример ДО на сумму

1.2 Построение

Пусть $\mathbb{X}$ – абстрактное множество, $\mathcal{B}\colon \mathbb{X} \times \mathbb{X} \to \mathbb{X}$, операция $\mathcal{B}$ удовлетворяет всем трём условиям из леммы 1.1. Опишем нерекурсивное построение дерева на массиве $\mathcal{A}$ длины n:

1. Будем считать, что $\log_2 n \in \mathbb{N}$, иначе дозаполним массив $\mathcal{A}$ до степени двойки нейтральными элементами.
2. Заведем массив $\mathcal{T}$ длины $2n - 1$, последние n элементов которого будут элементами исходного массива.
3. Первые $n-1$ элементов массива $\mathcal{T}$ заполним следующим образом: $\mathcal{T}[i] = \mathcal{B}(\mathcal{T}[2i+1], \mathcal{T}[2i+2])$ (заполнение от элемента с номером $n - 2$ и до 0 в цикле).

1.3 Ответ на запрос

Наша цель — получить результат операции на отрезке. Пусть мы находимся в узле, отвечающего за подотрезок $[l, r]$, а изначальный запрос был $[L, R]$.

1. Если $[l, r] \cap [L, R] = \varnothing$, то вернём нейтральный элемент.
2. Если $[l, r] \subseteq [L, R]$, то вернём значение в узле.
3. Иначе вернём результат операции с детей.

То есть исходный отрезок разбивается на дизъюнктное объединение подотрезков.

1.4 Время работы

Теорема 1.1. *Время ответа на запрос составляет $\mathcal{O}(\log n)$.*

Доказательство. Заметим, что на каждом уровне раскрываться вниз могут не более двух узлов, так как только крайние могут порождать дочерние запросы. Следовательно, время работы $\mathcal{O}(\log n)$. □

Algorithm 1 Query Function

function $query(int\ node,\ int\ a,\ int\ b)$
  $l \leftarrow tree[node].left$
  $r \leftarrow tree[node].right$
if $[l, r] \cap [a, b] = \varnothing$ **then**
  return $neutral$
end if
if $[l, r] \subseteq [a, b]$ **then**
  return $tree[node].res$
end if
return $\mathcal{B}(query(node \cdot 2 + 1, a, b),\ query(node \cdot 2 + 2, a, b))$

1.5 Обновление по индексу

Пусть необходимо обновить элемент с индексом i. Тогда:

1. Найдем в дереве лист, отвечающий за i-й элемент. Это $(n - 1 + i)$-й элемент.
2. Индекс родителя: $\lfloor \frac{i-1}{2} \rfloor$
3. Обновляем значение в родителе через результаты на деревьях.
4. Повторяем, пока не обновим корень.

2 Групповые операции

Определение 2.1. *Пусть $\mathbb{X}$ – абстрактное множество, $\mathcal{B}\colon \mathbb{X} \times \mathbb{X} \to \mathbb{X}$ – операция запроса на отрезке. Операцией группового обновления будем называть операцию $\mathcal{G}\colon \mathbb{X} \times \mathbb{X} \to \mathbb{X}$, которая применяется ко всем элементам на подотрезке. Условия на операцию $\mathcal{G}$:*

1. *Существование нейтрального элемента:* $\exists 0 \in \mathbb{X} : \forall a \in \mathbb{X} \quad \hookrightarrow \quad \mathcal{G}(a, 0) = \mathcal{G}(0, a) = a$
2. *Ассоциативность:* $\forall a, b, c \in \mathbb{X} \quad \hookrightarrow \quad \mathcal{G}(\mathcal{G}(a, b), c) = \mathcal{G}(a, \mathcal{G}(b, c))$
3. *Дистрибутивность:* $\forall a, b, c \in \mathbb{X} \quad \hookrightarrow \quad \mathcal{G}(\mathcal{B}(a, b), c) = \mathcal{B}(\mathcal{G}(a, c), \mathcal{G}(b, c))$

В каждом нижеприведенном псевдокоде в узлах дерева хранятся структуры из четырех полей:

- $left$ – левая граница полуинтервала, за который "отвечает"текущая вершина.
- $right$ – правая граница этого полуинтервала.
- ans – результат на отрезке по операции $\mathcal{B}$.
- d – несогласованность.

2.1 Проталкивание несогласованности

"Проталкивание" несогласованности детям. Необходимо выполнять как только идет рекурсивный запуск от текущей вершины к её детям. Нужно это для того, чтобы в детях в момент обработки были корректные данные.

Algorithm 2 Push Function

function $push(int\ node)$
  $tree[2 \cdot node + 1].d \leftarrow \mathcal{G}(tree[2 \cdot node + 1].d,\ tree[node].d)$
  $tree[2 \cdot node + 2].d \leftarrow \mathcal{G}(tree[2 \cdot node + 2].d,\ tree[node].d)$
  $tree[node].d \leftarrow \mathcal{G}\_neutral$

2.2 Групповое обновление

Процедура обновления на отрезке. Данная процедура выполняет разбиение текущего отрезка на подотрезки и обновление в них несогласованности. Очень важно выполнить push как только идет рекурсивный вызов от детей, чтобы избежать некорректной обработки в детях. И так как значение в детях могло измениться, то необходимо выполнить обновление ответа по операции $\mathcal{B}$ на текущем отрезке.

Algorithm 3 Update Function

function $update(int\ node,\ int\ a,\ int\ b,\ T\ val)$
{val - значение, которое поступило в качестве параметра на запрос, a и b - границы запроса}
$l \leftarrow tree[node].left$
$r \leftarrow tree[node].right$
if $[l, r) \cap [a, b) = \varnothing$ **then**
  return
end if
if $[l, r) \subseteq [a, b)$ **then**
    $tree[node].d \leftarrow \mathcal{G}(tree[node].d,\ val)$
  return
end if
  $push(node)$
  $update(2 \cdot node + 1,\ a,\ b,\ val)$
  $update(2 \cdot node + 2,\ a,\ b,\ val)$
  $tree[node].ans \leftarrow \mathcal{B}(\mathcal{G}(tree[2 \cdot node + 1].ans,\ tree[2 \cdot node + 1].d),$
      $\mathcal{G}(tree[2 \cdot node + 2].ans,\ tree[2 \cdot node + 2].d))$

2.3 Ответ на запрос

Получение ответа по операции $\mathcal{B}$. Отличие от операции обновления лишь в том, что для каждого отрезка разбиения необходимо не обновить несогласованность, а сложить по операции $\mathcal{B}$ с текущим ответом истинное значение на отрезке (то есть результат сложения по операции $\mathcal{G}$ значения в вершине с несогласованностью).

Algorithm 4 Query Function

function $query(int\ node,\ int\ a,\ int\ b)$
$l \leftarrow tree[node].left$
$r \leftarrow tree[node].right$
if $[l, r) \cap [a, b) = \varnothing$ **then**
    return $neutral$
end if
if $[l, r) \subseteq [a, b)$ **then**
    return $\mathcal{G}(tree[node].ans,\ tree[node].d)$
end if
    $push(node)$
    $T\ ans \leftarrow \mathcal{B}(query(2 \cdot node + 1,\ a,\ b),\ query(2 \cdot node + 2,\ a,\ b))$
    $tree[node].ans \leftarrow \mathcal{B}(\mathcal{G}(tree[2 \cdot node + 1].ans,\ tree[2 \cdot node + 1].d),$
                        $\mathcal{G}(tree[2 \cdot node + 2].ans,\ tree[2 \cdot node + 2].d))$
return ans