Билет 27. Дерево отрезков

Курс "Алгоритмы и структуры данных"

Билет 27. Дерево отрезков

Основная идея: Структура данных для быстрого вычисления операций на отрезках за O(log n)!

Что такое дерево отрезков?

Дерево отрезков — структура данных, которая позволяет за O(log n) выполнять операции на любом отрезке массива.

Требования к операции

Операция должна быть:

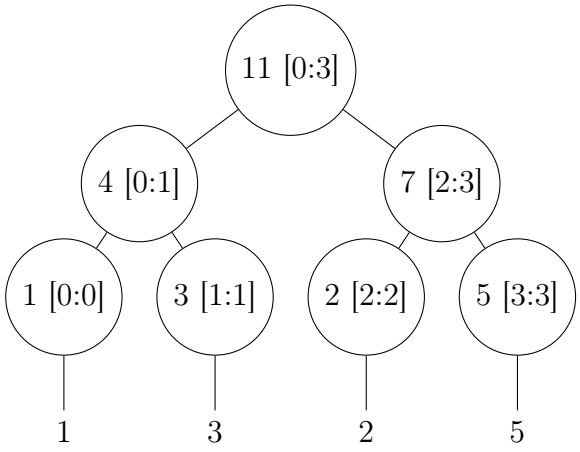
- **Ассоциативна:** $op(op(a, b), c) = op(a, op(b, c))$
- **Коммутативна:** $op(a, b) = op(b, a)$
- **Иметь нейтральный элемент:** $op(a, 0) = a$

Подходящие операции

- **Сумма:** $a + b$, нейтральный: 0
- **Произведение:** $a \times b$, нейтральный: 1
- **Минимум:** $\min(a, b)$, нейтральный: $+\infty$
- **Максимум:** $\max(a, b)$, нейтральный: $-\infty$
- **XOR:** $a \oplus b$, нейтральный: 0

Структура дерева отрезков

Пример для суммы на массиве [1, 3, 2, 5]



Нерекурсивное построение

Шаг 1: Подготовка массива

- Находим наименьшую степень двойки $\geq n$
- Дополняем массив нейтральными элементами

Пример: Исходный массив [1, 3, 2, 5], n = 4

1	3	2	5

Исходный массив:

Шаг 2: Создаём массив дерева размером 2N

Дерево (размер 8):

0	1	2	3	4	5	6	7
?	?	?	?	1	3	2	5

Шаг 3: Заполняем родительские узлы

Формула: $t[i] = t[2i] + t[2i+1]$

0	1	2	3	4	5	6
11	4	7	1	3	2	5

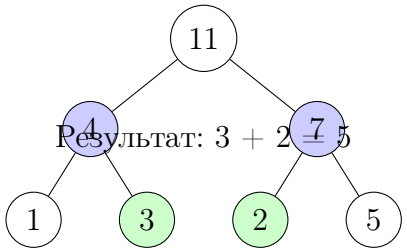
Операции на дереве отрезков

Запрос на отрезке [L, R]

1. Начинаем с листьев: $l = L + N$, $r = R + N$
2. Поднимаемся к корню, складывая нужные узлы
3. Если l - правый ребёнок: добавляем $t[l]$, $l++$
4. Если r - левый ребёнок: добавляем $t[r]$, $r--$

Пример: Сумма на [1, 2]

Запрос: сумма на [1,2]



Результат: 3 + 2 = 5

Обновление элемента

1. Находим лист: $pos = index + N$
2. Обновляем значение в листе
3. Поднимаемся к корню: $pos = pos/2$, обновляем $t[pos]$

Реализация на C++

```
class SegmentTree {
private:
    int n;
    vector<int> tree;

public:
    SegmentTree(vector<int>& nums) {
        n = nums.size();
        tree.resize(2 * n);

        // Инициализация листьев
        for (int i = 0; i < n; i++)
            tree[i + n] = nums[i];

        // Построение дерева
        for (int i = n - 1; i > 0; i--)
            tree[i] = tree[2 * i] + tree[2 * i + 1];
    }

    void update(int index, int value) {
        int pos = index + n;
        tree[pos] = value;

        while (pos > 1) {
            pos /= 2;
            tree[pos] = tree[2 * pos] + tree[2 * pos + 1];
        }
    }

    int query(int l, int r) {
        l += n;
        r += n;
        int sum = 0;

        while (l <= r) {
            if (l % 2 == 1) {
```

```
                sum += tree[l];
                l++;
            }
            if (r % 2 == 0) {
                sum += tree[r];
                r--;
            }
            l /= 2;
            r /= 2;
        }
        return sum;
    }
};
```

Пример работы

Исходный массив: [1, 3, 2, 5]

Операция	Параметры	Результат	Объяснение
query	[0, 2]	6	$1 + 3 + 2 = 6$
query	[1, 3]	10	$3 + 2 + 5 = 10$
update	index=1, value=4	-	Меняем $3 \rightarrow 4$
query	[0, 2]	7	$1 + 4 + 2 = 7$
query	[1, 1]	4	Просто элемент 4

Анализ сложности

- **Построение:** O(n)
- **Запрос:** O(log n)
- **Обновление:** O(log n)
- **Память:** O(n)