

# Лекция 8. Разреженная таблица. Дерево отрезков.

#вшпи #аисд #теория #дерево\_отрезков #структуры\_данных

Автор конспекта: Гридчин Михаил

## RMQ / RSQ

**Def.** RMQ - запрос поиска минимума на подотрезке

**Def.** RSQ - запрос поиска суммы на отрезке

**Def.** Online режим - если запросы поступают в режиме онлайн после ответов на предыдущие запросы

**Def.** Offline режим - если запросы известны заранее

**Def.** Static задача - если нет запросов изменений

**Def.** Dynamic задача - если есть запросы изменений.

Мы уже умеем решать static RSQ в online/offline с помощью префиксных сумм.

Также мы умеем решать static offline RMQ с помощью очереди на минимум и алгоритма Mo.

**Def.** Разреженная таблица (Sparse table) - двумерный массив st размера  $N$  на  $\log N$ , где

$$st[i][j] = f(a[i], a[i + 1], \dots, a[i + 2^j - 1]), \quad j = \{0, \dots, \log_2 N\}$$

$$st[i][j] = \begin{cases} f(st[i][j - 1], st[i + 2^{j-1}][j - 1]), & j > 0, \\ a[i], & j = 0 \end{cases}$$

Заметим, что мы знаем ответ для всех подотрезков длины  $2^j$ . Тогда пусть

$$fl_{log}[j] := \text{floor}(\log_2 j)$$

Тогда  $f(a[l], a[l + 1], \dots, a[r]) = f(st[l][j], st[r - 2^j + 1][j]), j = fl_{log}[r - l + 1]$

Потребляемые ресурсы  $O(N \log N)$  по памяти и  $O(N \log N)$  по времени построения и  $O(1)$  на запрос.

Sparse Table может решать задачу static online RMQ.

**Def.** Идемпотентность - свойство операции, согласно которому её повторное применение к одному и тому же объекту не меняет конечный результат после первого выполнения.  $\forall a \implies f(a, a) = a$ .

## Дерево отрезков

**Def.** Дерево отрезков за  $O(\log N)$  способна считать операцию на отрезке для ассоциативной, коммутативной операции с нейтральным элементом.

### Построение

- будем считать, что  $\log_2 N \in \mathbb{N}$  иначе дополним до степени двойки нейтральными элементами
- Заведём массив длины  $2N - 1$ , последние  $N$  элементов будут элементами исходного массива
- Первые  $N - 1$  элементов заполняем по формуле  $t[i] = F(t[2i + 1], t[2i + 2])$ .  
Наша цель получить результат на подотрезке  $[L, R]$ . Пусть мы находимся в вершине  $[l, r]$ . Возможны три случая
  - $[l, r] \cap [L, R] = \emptyset \implies \text{optional}$
  - $[l, r] \subseteq [L, R] \implies s[v]$
  - Возвращаем результат  $F(query(2v + 1, L, R), query(2v + 2, L, R))$ .To есть разбиваем наш подотрезок на дизъюнктные подотрезки длины степени двойки.

### Время работы.

Заметим, что на каждом уровне раскрываются вниз могут не более двух узлов, так как только крайние могут порождать дочерние. Время работы  $O(\log N)$ .

*Обновление в точке.* Пусть необходимо обновить элемент с индексом  $i$ .

- присвоим  $t[i + N] = new\_value$
- поднимаемся вверх по  $\log N$  узлов и пересчитываем корректное значение функции на отрезке. Индекс родителя -  $\frac{i-1}{2}$ .

**Def.** Операция группового обновления  $ch$  будем называть запрос, в ходе которого применяется на подотрезке операция  $ch(\circ, val)$ .

Пусть  $ch$  - операция изменения,  $op$  - операция запроса. Тогда

- $\exists e : ch(a, e) = e$
- ассоциативность
- дистрибутивность  $ch(op(a, b), c) = op(ch(a, c), b)$

**Def.** Несогласованность - величина, с которой надо выполнить операцию  $op$ , чтобы получить корректный результат. В каждый момент времени в узле необязательно лежит

истинное значение *op* на отрезке. Но к моменту запроса будем осуществлять *проталкивание несогласованности*.