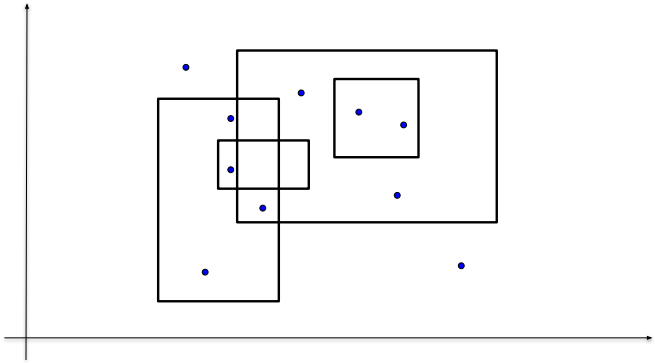
Решение двумерных задач. Персистентные и двумерные структуры данных

Россия, Санкт-Петербург

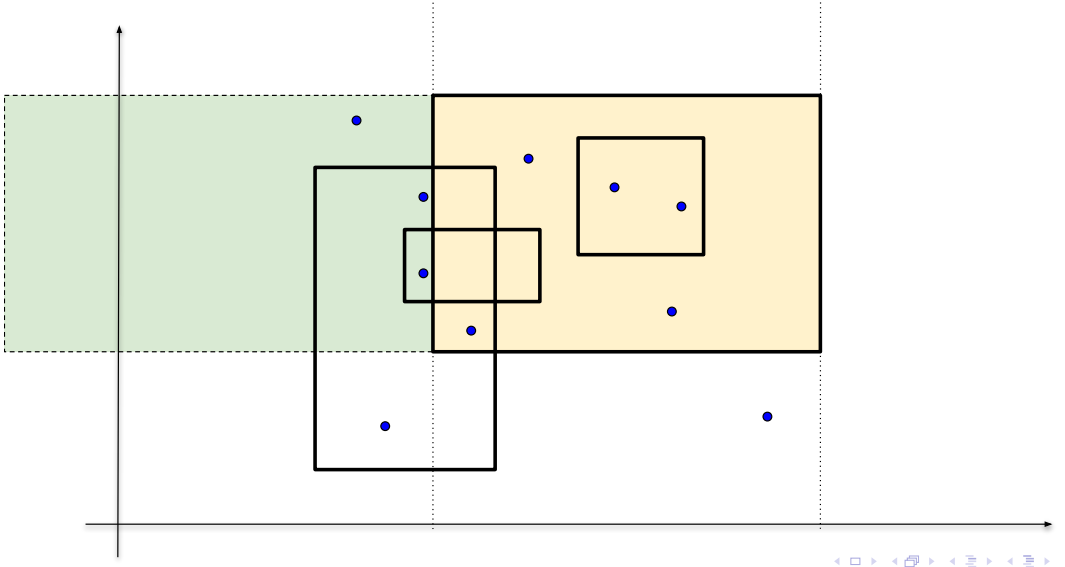
13 мая 2020

Пример задачи

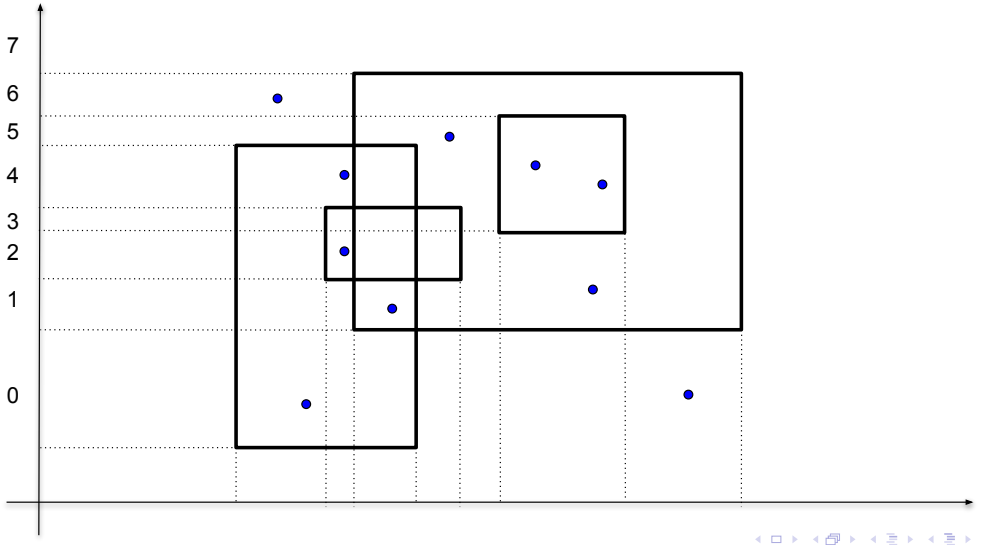
- Заданы n точек на плоскости (x_i, y_i)
- Запросы: прямоугольники $[l_x, r_x] \times [l_y, r_y]$ — сколько точек внутри
- Ответ на запрос за $\mathcal{O}(\log n)$.



Offline: сканирующая прямая



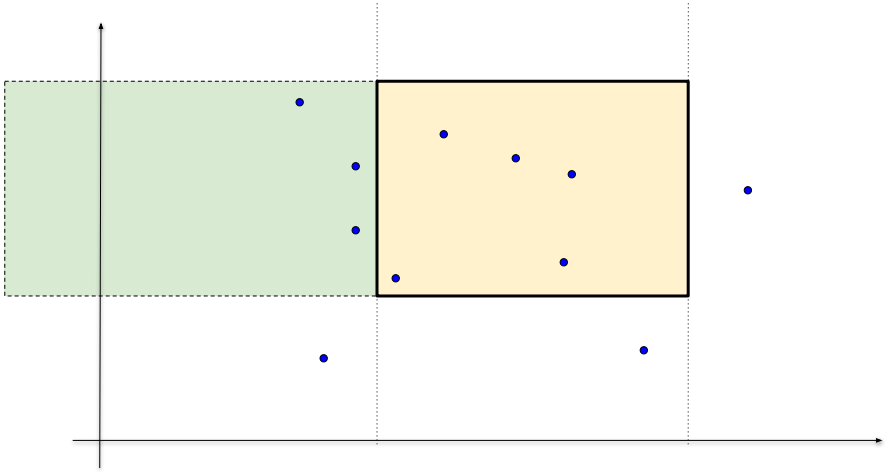
Offline: сканирующая прямая



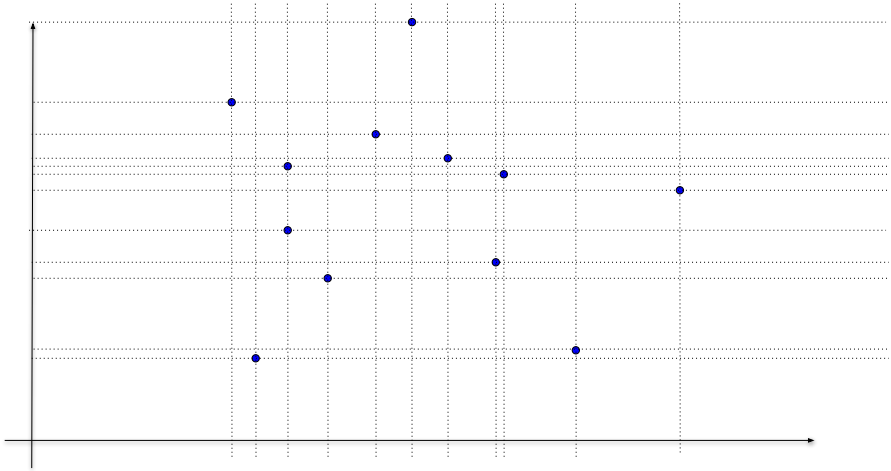
Offline: сканирующая прямая

- Отсортировать все вертикальные стороны и точки по x.
- Сжать координаты по y.
- Каждый прямоугольник это разность:
  1. $(-\infty, r_x] \times [l_y, r_y]$;
  2. $(-\infty, l_x) \times [l_y, r_y]$.
- Дерево отрезков:
  - для каждого y сохраним сколько точек встретили.
- Ответ на запрос: сумма на отрезке.
- Время работы: $\mathcal{O}((n+m)\log(n+m))$.

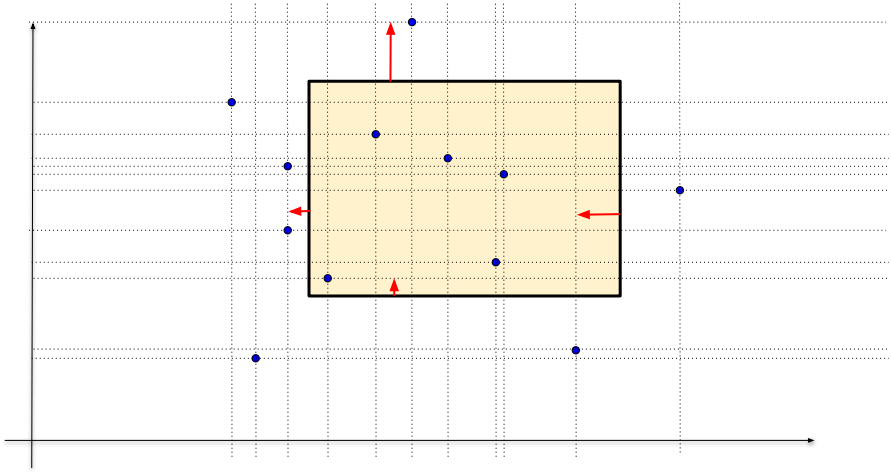
Персистентное дерево



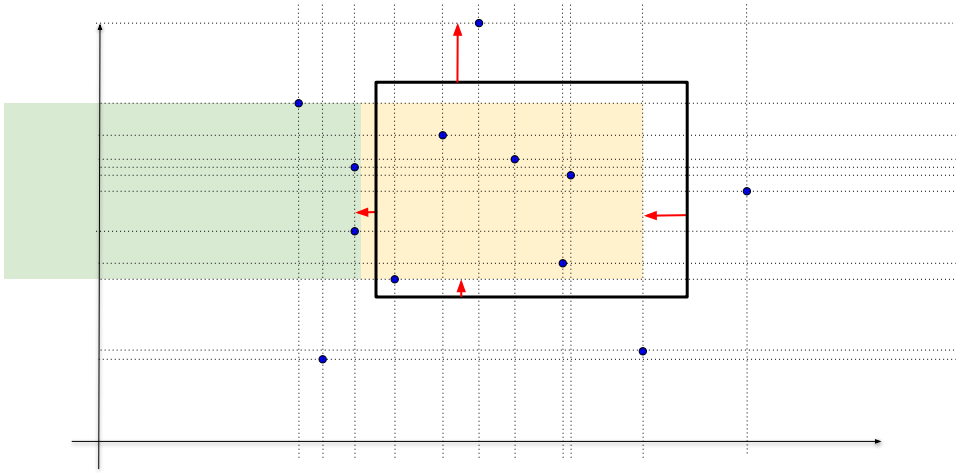
Персистентное дерево



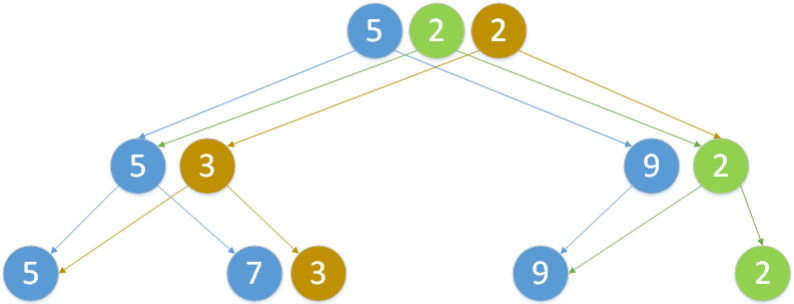
Персистентное дерево



Персистентное дерево



Персистентное дерево



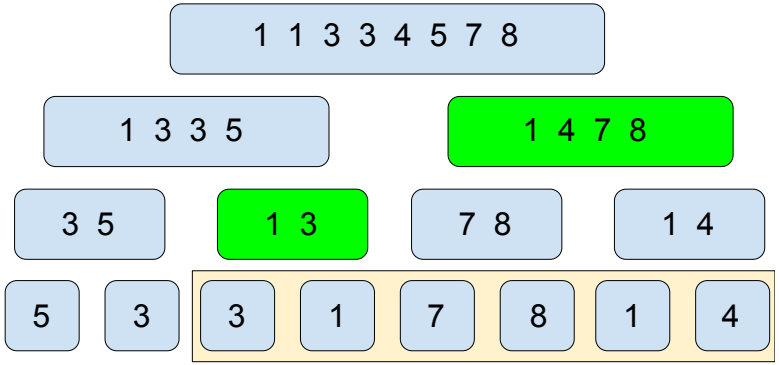
Персистентное дерево

- Пользуемся той же идеей, но точек не знаем.
- Сортируем только точки.
- Сжимаем координаты только точек.
- Прямоугольник делим на два прямоугольника:
  - двоичным поиском ищем, когда добавлена последняя точка;
  - берем версию дерева отрезков;
  - ищем в ней сумму;
  - отрезок тоже ищем двоичным поиском.
- Ответ на запрос: $\mathcal{O}(\log n)$.

Двумерное дерево

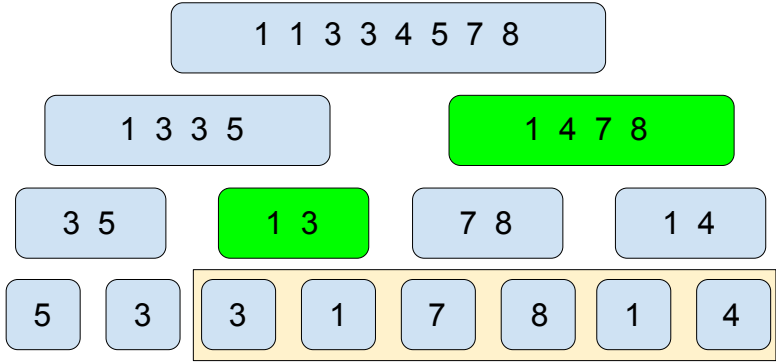
- Возьмем точки (x_i, y_i).
- Посортируем по x_i.
- Двумерное дерево для $\{y_i\}$.

- На картинке дерево для: $(1,5)$, $(2,3)$, $(3,3)$, $(4,1)$, $(4,7)$, $(5,8)$, $(7,1)$, $(7,4)$.



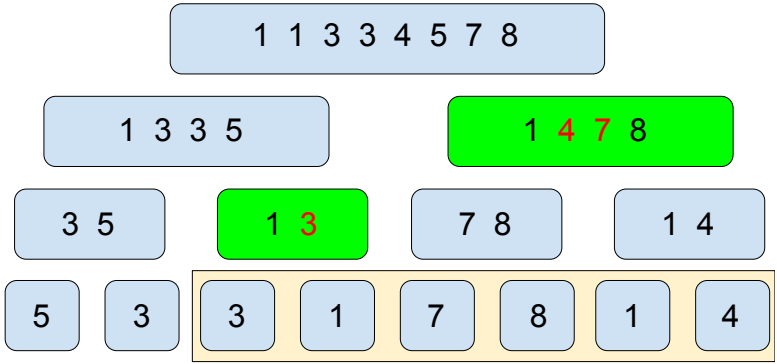
Двумерное дерево

- $[l_x, r_x] \times [l_y, r_y]$ выглядит так:
- $[l_x, r_x]$ — отрезок дерева;
- $[l_y, r_y]$ — значения в вершине, которые отсортированы.



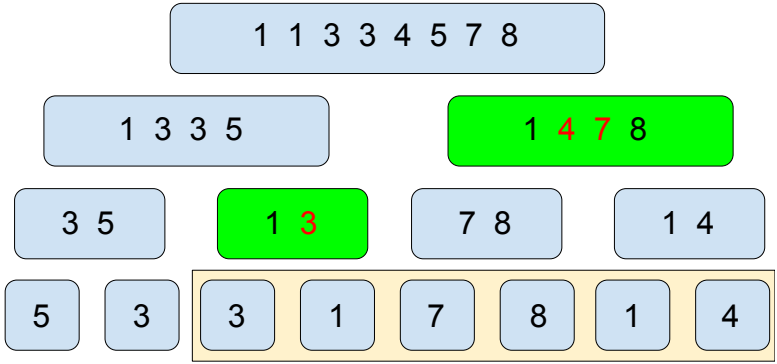
Двумерное дерево

- На картинке дерево для: $(1,5)$, $(2,3)$, $(3,3)$, $(4,1)$, $(4,7)$, $(5,8)$, $(7,1)$, $(7,4)$.
- Прямоугольник: $[3,10] \times [3,7]$



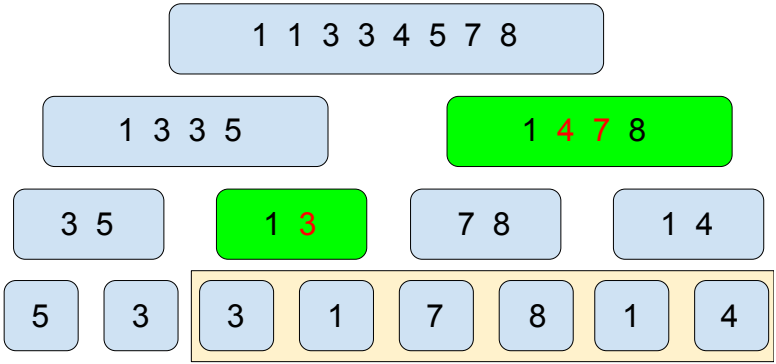
Двумерное дерево

- Построение: MERGE детей.
- Запрос: выделить вершины + двоичный поиск в каждой вершине.
- Время построения: $\mathcal{O}(n \log n)$, время на запрос: $\mathcal{O}(\log^2 n)$

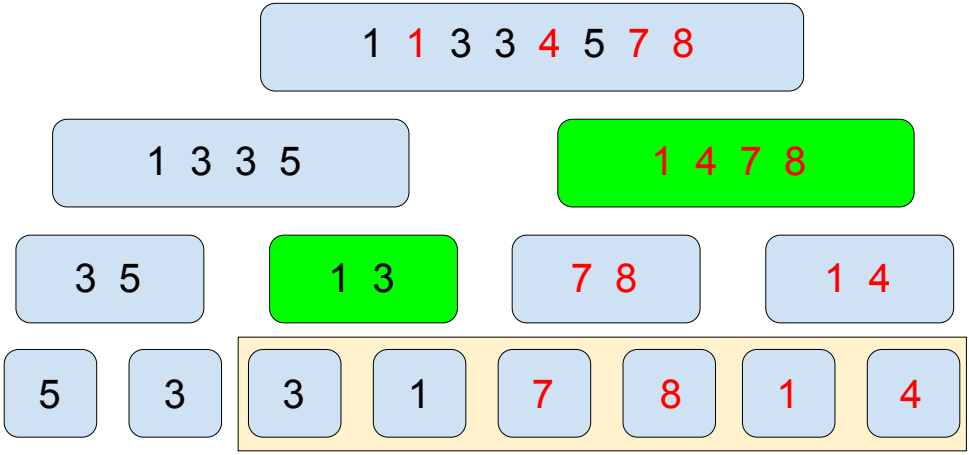


Двумерное дерево

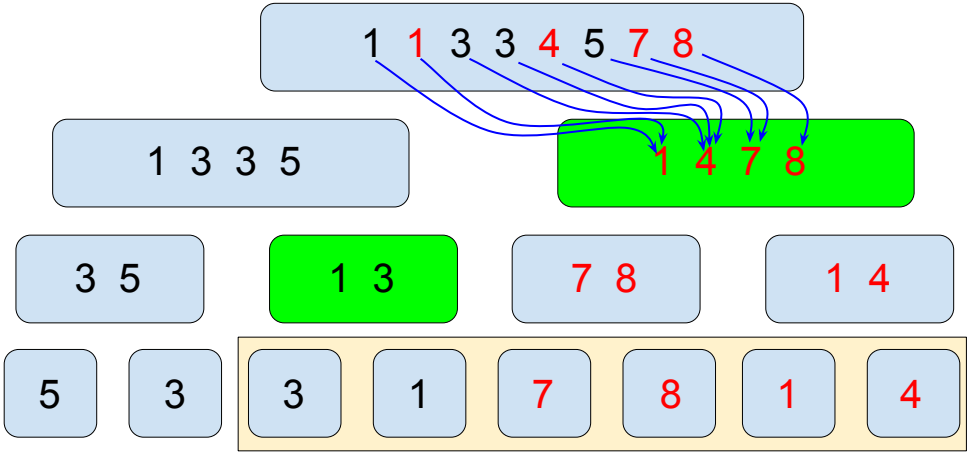
- English: Layered range tree
- Gabow, Bentley, Tarjan (1984)



Fractional cascading



Fractional cascading



Fractional cascading

- Для каждого элемента сохраним два указателя:
  - по одному в каждого из детей;
  - для элемента x указатель в минимальное y, что $y \geqslant x$;
- Это можно считать по ходу операции $\textsc{Merge}$.
- Двоичный поиск делаем в корне:
  - из найденного элемента по указателям переходим в детей;
  - в детях двоичный поиск не нужен.
- Построение указателей: $\mathcal{O}(n \log n)$.
- Ответ на запрос: $\mathcal{O}(\log n)$ на бинпоиск в корне $+$ $\mathcal{O}(\log n)$ на поиск вершин дерева отрезков.

Описание задачи

- Задан массив чисел: $a_1, a_2, \ldots, a_n$.
- Запросы:
  - Задан отрезок $[L, R)$ и число k;
  - Найти k-й по возрастанию элемент среди $a_L, a_{L+1}, \ldots, a_{R-1}$.

Сведение к числу точек в прямоугольнике

- k-й по возрастанию элемент $\leqslant x$,
  - если число элементов $\leqslant x$ среди $a_L, a_{L+1}, \ldots, a_{R-1}$ хотя бы k;
  - $|\{i \mid a_i \leqslant x \wedge L \leqslant i < R\}| \geqslant k$.
- Сделаем двоичный поиск по x:
  - требуется проверить, сколько есть точек (i, a_i) в прямоугольнике $[L, R) \times (-\infty, x]$.
- Минимальное такое x и есть k-й элемент.
- Персистентное дерево или Двумерное дерево с fractional cascading
  - $\log n$ на бинпоиск, и $\log n$ на подсчет числа точек;
  - ответ на запрос за время $\mathcal{O}(\log^2 n)$.

Персистентное дерево

i	1	2	3	4	5	6	7	8
a_i	3	5	6	1	7	4	2	8

- Структура данных для подсчета числа точек в прямоугольнике:
  - $c_i[x]$ — сколько чисел равных x на префиксе $[1 \dots i]$;
  - увеличение префикса — $+1$ число.
- k-й по возрастанию элемент $\leqslant x$,
  - если число элементов $\leqslant x$ среди $a_L, a_{L+1}, \dots, a_{R-1}$ хотя бы k;
  - $|\{i \mid a_i \leqslant x \wedge L \leqslant i < R\}| \geqslant k$;
  - $|\{i \mid a_i \leqslant x \wedge L \leqslant i < R\}| = c_{R-1}[-\infty \dots x] - c_{L-1}[-\infty \dots x]$.
- $L = 3$, $R = 7$

x	1	2	3	4	5	6	7	8
$c_2[x]$	0	0	1	0	1	0	0	0
$c_6[x]$	1	0	1	1	1	1	1	0

Персистентное дерево

- $L = 3$, $R = 7$, $k = 3$

x	1	2	3	4	5	6	7	8	$3 - 1 = 2$
$c_2[x]$	0	0	1	0	1	0	0	0	1
$c_6[x]$	1	0	1	1	1	1	1	0	3

  - элементов $\leqslant 4$ на отрезке 2, что меньше $k = 3$;
  - уменьшаем k на 2, теперь $k = 1$.

- $L = 3$, $R = 7$, $k = 1$

x	1	2	3	4	5	6	7	8	$2 - 1 = 1$
$c_2[x]$	0	0	1	0	1	0	0	0	1
$c_6[x]$	1	0	1	1	1	1	1	0	2

  - элементов $\leqslant 6$ и > 4 на отрезке 1, что $\leqslant k = 1$;
  - сдвигаем влево.

Персистентное дерево

- $L = 3$, $R = 7$, $k = 1$

x	1	2	3	4	5	6	7	8	$1 - 1 = 0$
$c_2[x]$	0	0	1	0	1	0	0	0	1
$c_6[x]$	1	0	1	1	1	1	1	0	1

  - элементов $\leqslant 5$ и > 4 на отрезке 0, что меньше $k = 1$;
  - сдвигаем вправо;
  - Ответ: 6.
- Строим $c_i[x]$ для всех i: персистентное дерево.
- Делаем спуск по двум деревьям параллельно:
  - в обоих деревьях идем налево;
  - если разность $\geqslant k$, то остаемся слева;
  - если разность $< k$, то идем вправо уменьшая k.
- Время работы: $\mathcal{O}(n \log n)$ препроцессинг, $\mathcal{O}(\log n)$ на запрос.

Двумерное дерево, другие координаты

i	1	2	3	4	5	6	7	8
a_i	3	5	6	1	7	4	2	8

Отсортируем:

i	4	7	1	6	2	3	5	8
a_i	1	2	3	4	5	6	7	8

Построим двумерное дерево для (a_i, i):

<table>
<tr><td colspan="8">1 2 3 4 5 6 7 8</td></tr>
<tr><td colspan="4">1 4 6 7</td><td colspan="4">2 3 5 8</td></tr>
<tr><td colspan="2">4 7</td><td colspan="2">1 6</td><td colspan="2">2 3</td><td colspan="2">5 8</td></tr>
<tr><td>4</td><td>7</td><td>1</td><td>6</td><td>2</td><td>3</td><td>5</td><td>8</td></tr>
</table>