1 Range-деревья

Range-дерево (дерево параллелепипедов) — структура данных, позволяющая хранить n точек $p_1, p_2, \ldots, p_n \in \mathbb{R}^d$ и отвечать на запросы вида «сколько точек p_i находится в параллелепипеде $[x_1^1, x_2^1] \times \ldots \times [x_1^d, x_2^d]$» за время $O(\log^{d-1} n)$; при этом построение структуры данных для n точек занимает время $O(n \cdot \log^{d-1} n)$.

При $d \geq 2$ d-мерное дерево параллелепипедов устроено следующим образом: это дерево отрезков для множества всех точек, упорядоченного по первой координате, в каждой вершине которого хранится, для множества всех точек p_i в листьях–потомках данной вершины,

- их количество;
- наибольшее и наименьшее значение их первой координаты;
- $d-1$-мерное дерево параллелепипедов по оставшимся $d-1$ координатам, кроме первой.

Дерево параллелепипедов размерности $d-1$, построенное в данной вершине d-мерного дерева для множества точек в её поддереве, называется *ассоциированной структурой*, см. рисунок 1.

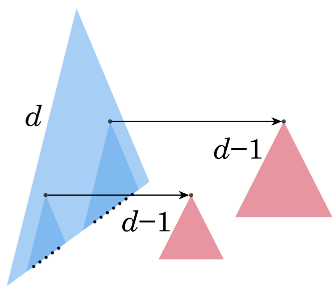

Рис. 1: Ассоциированные структуры: в каждом внутреннем узле d-мерного дерева параллелепипедов — $d-1$-мерное дерево, построенное для множества всех точек p_i под данным узлом

Так как d-мерное дерево по сути *состоит* из $d-1$-мерных, то понимать, какое время требуется на построение и на запрос, мы будем индукцией по размерности. Только *базой индукции* будет случай $d = 2$: для двумерного дерева мы покажем, как его можно нетривиально ускорить.

Теорема 1. *Пусть $d \geq 3$; запрос к d-мерному дереву параллелепипедов сводится к $\log n$ запросам к $d-1$-мерным деревьям.*

Доказательство. Сначала вспомним, как устроен запрос к одномерному дереву отрезков — а устроен он следующим образом. Будем рекурсивно спускаться из корня, и в текущей вершине для каждого из двух детей:

- если отрезок, накрываемый им, не пересекает диапазон из запроса, — не вызываемся от этого ребёнка,
- если отрезок, накрываемый им, полностью содержится в диапазоне из запроса, — прибавляем к ответу количество точек, находящихся в листьях под этим ребёнком (это количество хранится непосредственно в нём);
- если отрезок, накрываемый им, пересекается с диапазоном из запроса, — вызываемся от него рекурсивно.

В общем, дерево отрезков в точности накрывает диапазон из запроса поддеревьями $\log n$ своих узлов, и ответ на запрос — сумма хранящихся в этих узлах количеств точек в листьях их поддерева.

Пусть теперь мы отвечаем на d-мерный запрос $[x_1^1, x_2^1] \times \ldots \times [x_1^d, x_2^d]$. Наше дерево параллелепипедов — оно же дерево отрезков по первой координате. Потому мы можем, как описано выше, накрыть множество точек p_i, первая координата которых лежит в отрезке $[x_1^1, x_2^1]$, $\log n$ вершинами дерева.

После этого останется из множеств точек p_i под каждой из этих вершин отфильтровать те, оставшиеся координаты которых лежат в интересующих нас диапазонах. Это можно сделать как раз запросами к $d-1$-мерным деревьям. $\square$

Теорема 2. *Пусть $d \geq 3$. Если $d-1$-мерное дерево отрезков для n точек строится за время $O(n \cdot \log^{d-2} n)$, то d-мерное дерево отрезков для n точек строится за время $O(n \cdot \log^{d-1} n)$.*

Доказательство.

$$T(n) \;=\; 2 \cdot T\left(\frac{n}{2}\right) + O(n \cdot \log^{d-2} n) \;=\; \sum_{k=0}^{\log n} 2^k \cdot \frac{n}{2^k} \cdot \log^{d-2}\left(\frac{n}{2^k}\right) \;=$$

$$=\; n \cdot \left(\log^{d-2} n + \left(\log^{d-2} n\right) + \ldots + 1^{d-2}\right) \;=\; n \cdot \log^{d-1} n.$$

$\square$

Теорема 3. *Двумерное дерево параллелепипедов можно построить за время $O(n \cdot \log n)$, а отвечать на запросы к нему — за время $O(\log n)$.*

Доказательство. То, что описано далее, гуглить/искать в [de Berg, van Kreveld et al.] по ключевым словам *“fractional cascading”*. Для быстрого построения сначала упорядочим все точки по *второй* координате, и в каждой вершине будем хранить *массив* из всех точек (вместо одномерного дерева отрезков на них).

Заметим, что массивы в потомках любой вершины являются подмножествами массива в ней. Потому, чтобы быстро отвечать на запрос, мы сначала найдём границы диапазона из запроса (по второй координате) в массиве в корне (двоичным поиском за $O(\log n)$). А затем будем получать границы этого диапазона в массиве в каждой из вершин, посещённых нами при запросе к дереву отрезков, без необходимости осуществлять там поиск.

Чтобы иметь возможность так сделать, строим дерево следующим образом (см. рисунок 2): упорядоченный по второй координате массив в каждой из вершин будем разбивать на две половины по первой координате, найдя по ней медиану, за линейное время. После чего каждый элемент соединим ссылкой с *первым элементом с не меньшей второй координатой* в каждом из массивов в детях, также за линию. Время на построение — $T(n) = 2T(\frac{n}{2}) + O(n) = O(n \cdot \log n)$.

При ответе на запрос будем переходить по ссылкам из крайних элементов диапазона по второй координате в вершины, которые посещает запрос к дереву отрезков по первой координате. Если ссылка из правого края пришла в элемент, не входящий в диапазон запроса по второй координате, отступим в массиве на один элемент влево. Время на запрос — $O(\log n)$ на двоичный поиск в массиве в корне и $O(\log n)$ переходов по ссылкам. $\square$

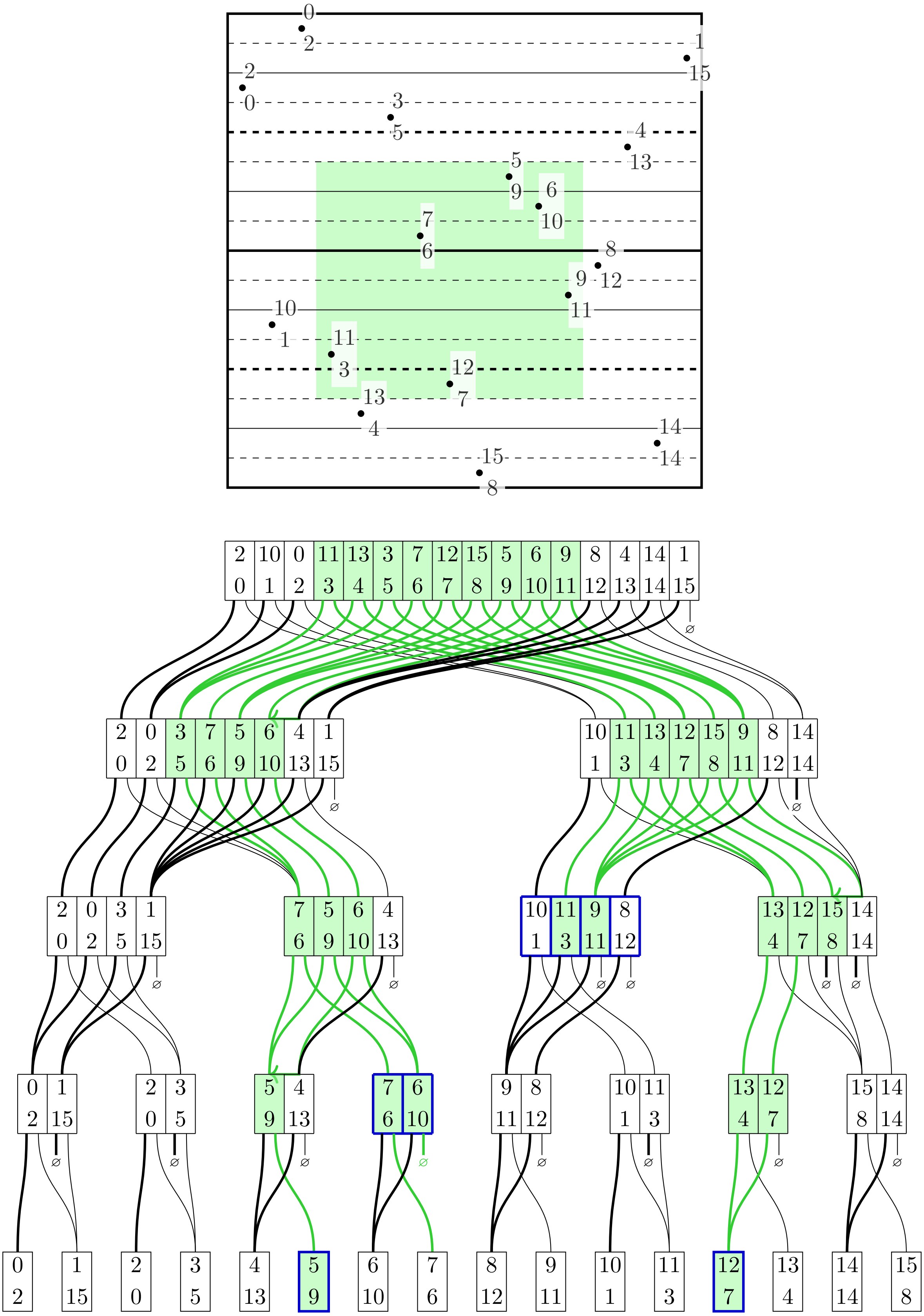

Рис. 2: Множество точек в $\mathbb{R}^2$, границы дерева отрезков по первой координате и запрос $[5, 12] \times [3, 9]$ (вверху, координаты «матричные»); двумерное дерево параллелепипедов и переходы по ссылкам, соответствующие обработке этого запроса (внизу). Синим обведены вершины дерева отрезков, накрывающие диапазон запроса по первой координате. Зелёная заливка — точки, принадлежащие диапазону по второй координате.