

1 Range-деревья

Range-дерево (*дерево параллелепипедов*) — структура данных, позволяющая хранить n точек $p_1, p_2, \dots, p_n \in \mathbb{R}^d$ и отвечать на запросы вида «сколько точек p_i находится в параллелепипеде $[x_1^1, x_2^1] \times \dots \times [x_1^d, x_2^d]$ » 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.

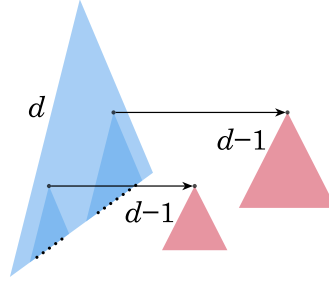


Рис. 1: Ассоциированные структуры: в каждом внутреннем узле d -мерного дерева параллелепипедов — $d - 1$ -мерное дерево, построенное для множества всех точек p_i под данным узлом

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 поддерев.

Пусть теперь мы отвечаем на d -мерный запрос $[x_1^1, x_2^1] \times \dots \times [x_1^d, x_2^d]$. 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.

$$\begin{aligned} T(n) &= 2 \cdot T\left(\frac{n}{2}\right) + O(n \cdot \log^{d-2} n) = \sum_{k=0}^{\log n} 2^k \cdot \frac{n}{2^k} \cdot \log^{d-2}\left(\frac{n}{2^k}\right) = \\ &= n \cdot \left(\log^{d-2} n + \left(\log^{d-2} \frac{n}{2} \right) + \dots + 1^{d-2} \right) = n \cdot \log^{d-1} n. \end{aligned}$$

\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 корне будем хранить массив из всех точек (вместо одномерного дерева отрезков на них).

Напомним, что в упорядоченном массиве тоже можно отвечать на запросы о количестве точек в диапазоне за $\log n$: два двоичных поиска по краям диапазона, а затем разность индексов. А также в каждой вершине разбить множество точек на две половины по первой координате, найдя медиану по этой координате, можно за линейное время.

(ссылки в массивы в детях)

\square