Лекция по курсу «Алгоритмы и структуры данных» / «Технологии и методы программирования»

Сбалансированные деревья. АВЛ-деревья. Префиксные деревья. Поиск многомерных данных: kd-, vp-, ball-деревья

Мясников Е.В.

Сбалансированные деревья

Сбалансированные деревья

Бинарное дерево называется ***идеально сбалансированным***, если для каждого его узла количество узлов в левом и правом поддеревьях отличается не более чем на единицу.

Идеально сбалансированное дерево из n узлов имеет высоту

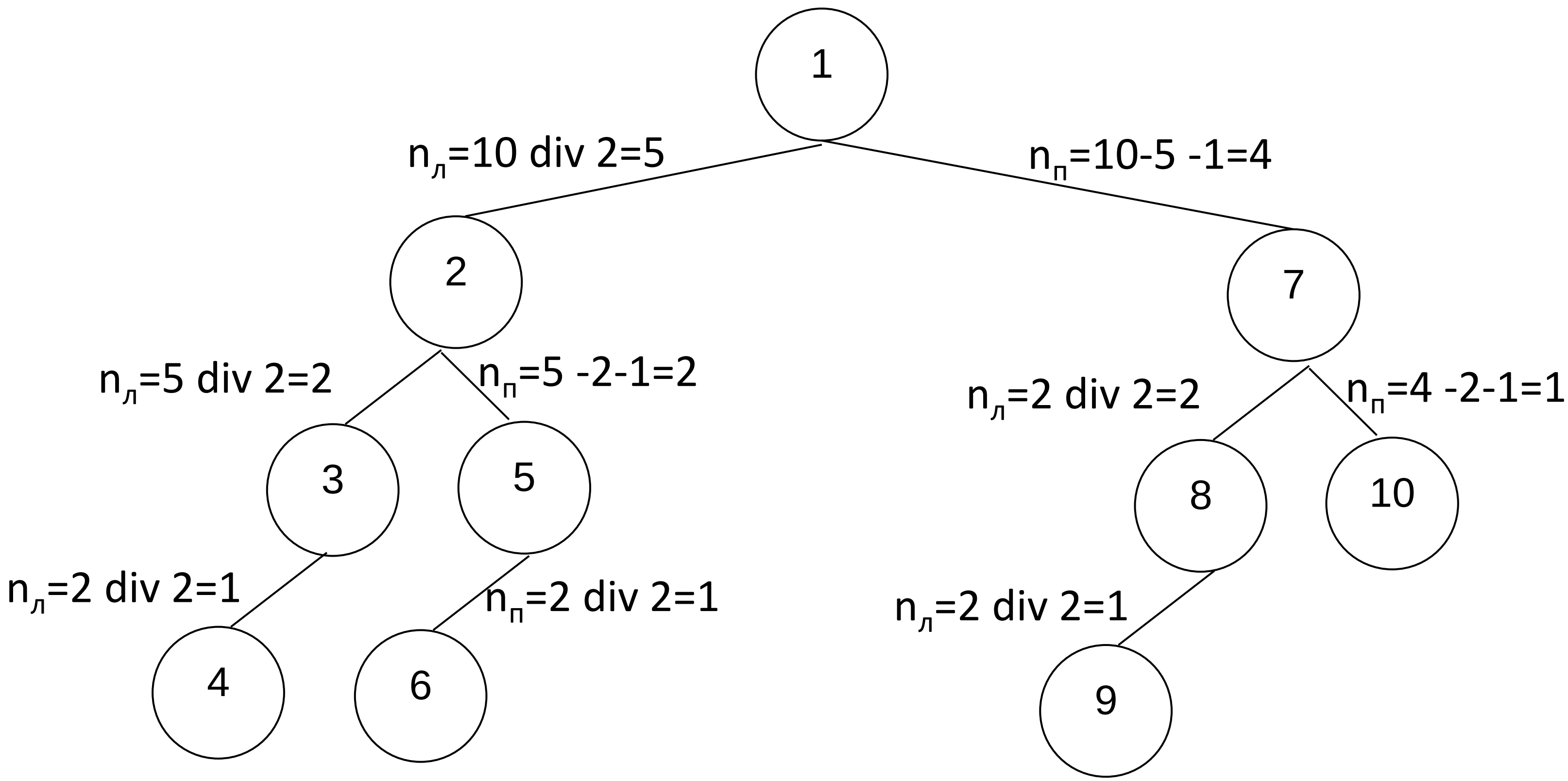
$$[\log_2 n] + 1.$$

Такая высота является минимальной для бинарных деревьев, состоящих из **n** узлов.

Алгоритм построения сбалансированного дерева из n узлов является рекурсивным и состоит из следующих шагов:

1. Взять один узел в качестве корня.
2. Построить по данному алгоритму левое поддерево с числом узлов $n_л = n \text{ div } 2$.
3. Построить по данному алгоритму правое поддерево с числом узлов $n_п = n - n \text{ div } 2 - 1$

Сбалансированные деревья: пример построения



Сбалансированные деревья поиска

Поиск в идеально сбалансированном дереве осуществляется за логарифмическое время. Построение идеально сбалансированного дерева поиска для упорядоченной последовательности значений не представляет большой трудности, однако поддержание сбалансированности при удалении и добавлении узлов весьма трудоемко. По этой причине для построения сбалансированных деревьев поиска используют иную формулировку сбалансированности.

Дерево называют ***сбалансированным*** тогда и только тогда, когда высоты двух поддеревьев каждой из его вершин отличается не более чем на единицу. Такие деревья называют также ***АВЛ-деревьями*** по первым буквам имен их создателей - советских ученых Г.М. Адельсона-Вельского и Е.М. Ландиса.

В АВЛ деревьях для каждого узла дерева вводят показатель сбалансированности и выполняют балансировку (так называемые вращения дерева) в том случае, если показатель сбалансированности нарушается.

АВЛ-деревья

АВЛ-деревья были предложены советскими учеными в работе [Адельсон-Вельский Г.М., Ландис Е.М. Один алгоритм организации информации // Доклады АН СССР. – 1962. Т. 146, № 2. – С. 263–266].

Основными операциями, как и при использовании других структур данных, являются:
- поиск узла по ключу,
- добавление нового узла,
- удаление узла.

Поиск узла выполняется тривиальным образом, как и в других бинарных деревьях поиска.

Добавление и удаление вершин начинаются также как и в рассмотренном ранее простом бинарном дереве поиска.
При **добавлении** производится поиск по ключу того места, в котором должен находиться вставляемый узел с последующей вставкой узла.
При **удалении**, как и ранее, удаляемый узел заменяется на самый левый (правый) узел правого (левого) поддерева.

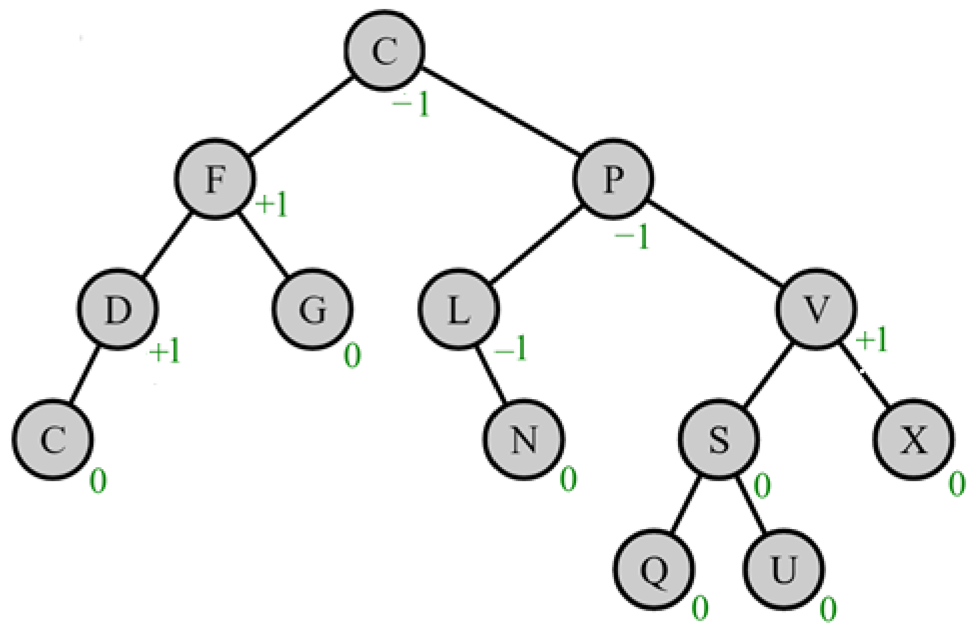
АВЛ-деревья

Так как операции вставки и удаления могут нарушить сбалансированность дерева, после проведения таких операций выполняется оценка сбалансированности с последующей балансировкой при необходимости.
В оценке сбалансированности участвует так называемый ***коэффициент сбалансированности узла*** – разность высот левого и правого поддеревьев данного узла:

$d(a) = h(L_a)-h(R_a)$,

где **$h(L_a)$** и **$h(R_a)$** – высота левого и правого поддеревьев узла *a*, соответственно.

В АВЛ-дереве допустимыми значениями коэффициента сбалансированности являются –1, 0 и 1.

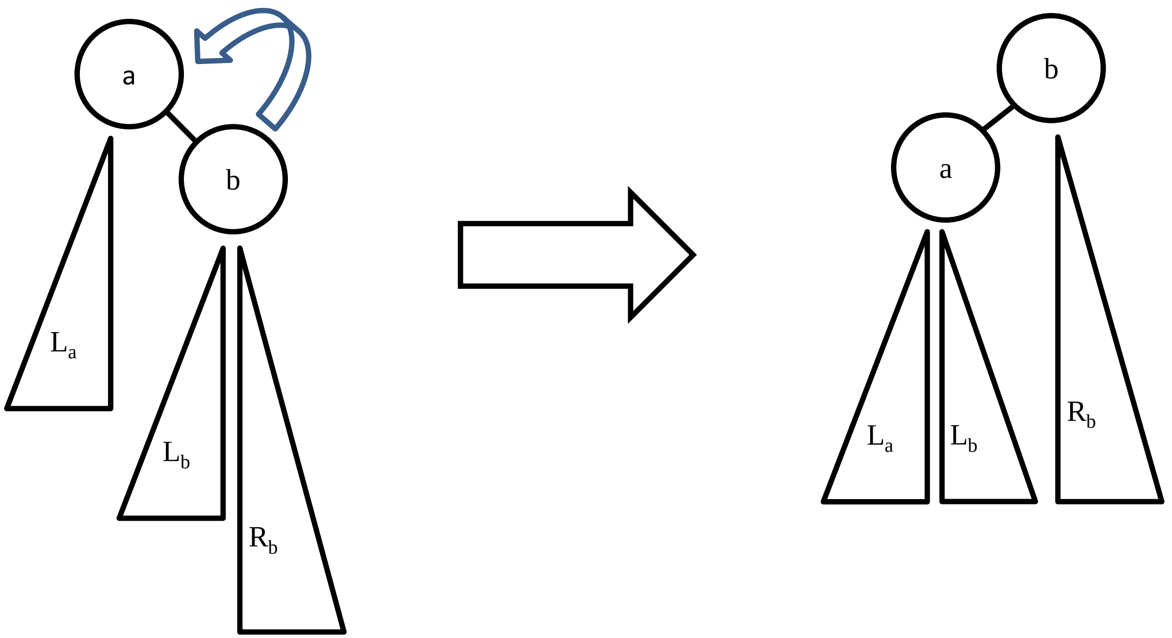


Таким образом, если значение коэффициента сбалансированности для некоторого узла оказывается равным 2 (-2), выполняется операция балансировки, приводящая коэффициент к допустимому значению.

АВЛ-деревья: Малое левое вращение

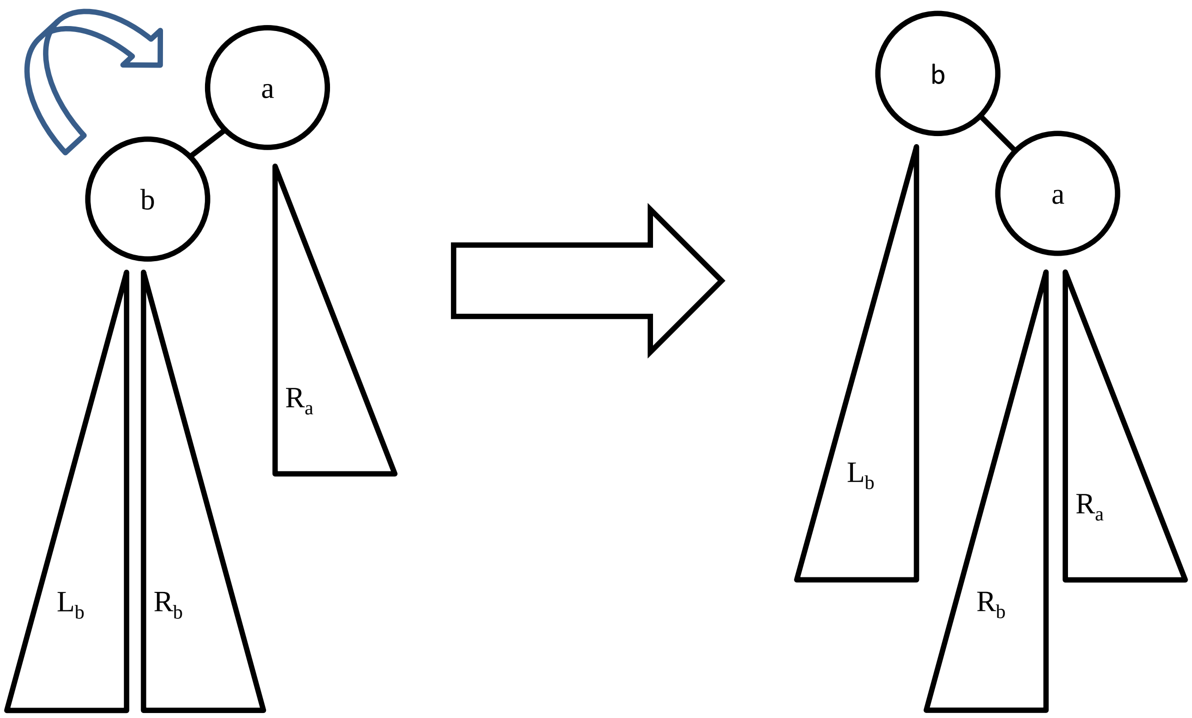
Предположим, что в АВЛ-дереве имеются два узла: узел-предок **a**, его потомок **b**, и для узла **a** в результате некоторой модификации нарушено условие сбалансированности, то есть **d(a)=-2** или **d(a) = 2**. Рассматривается несколько случаев:

- если ***(d(a) = -2) и (d(b) = -1 или d(b) = 0)***, то выполняется малое левое вращение. Пример такого вращения для ***($h(L_a)$-h(b) = -2) и ($h(L_b)$-$h(R_b)$ = -1)*** показан на рис.

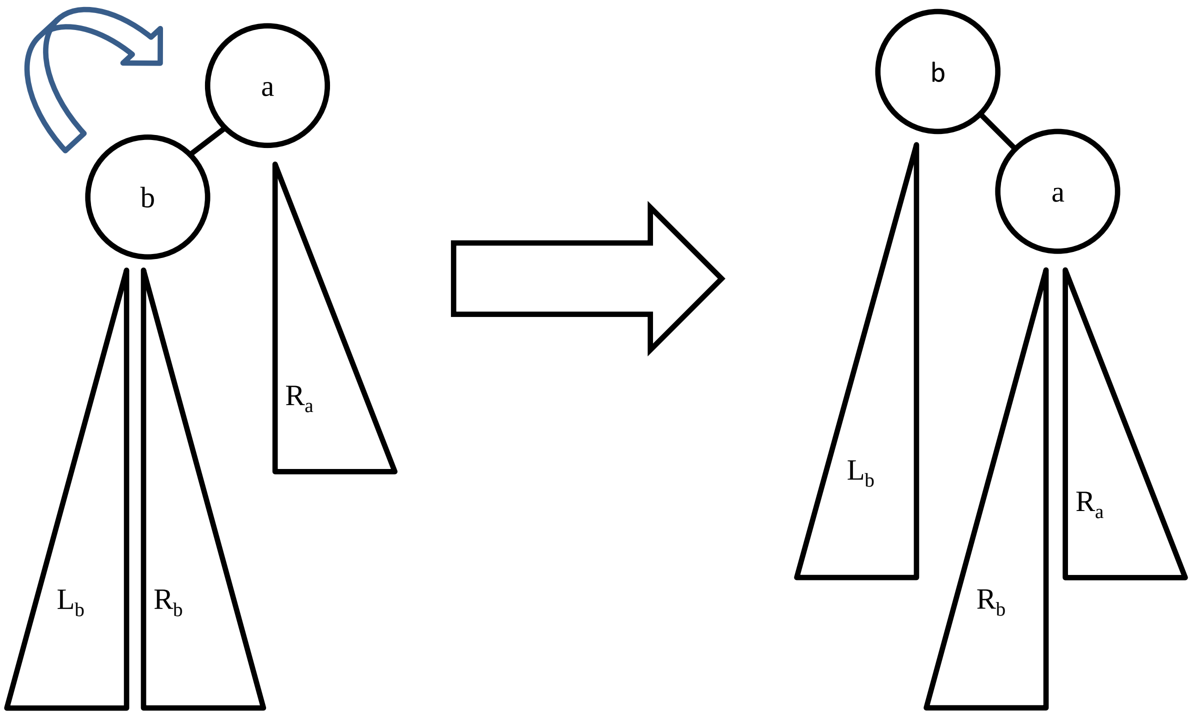


АВЛ-деревья: Малое правое вращение

- если **(d(a) = 2) и (d(b) = 1 или d(b) = 0)**, то выполняется малое правое вращение. Пример для случая **($h(L_a)$-h(b) = 2) и ($h(L_b)$-$h(R_b)$) = 0)** показан на рис.



АВЛ-деревья: Малое правое вращение



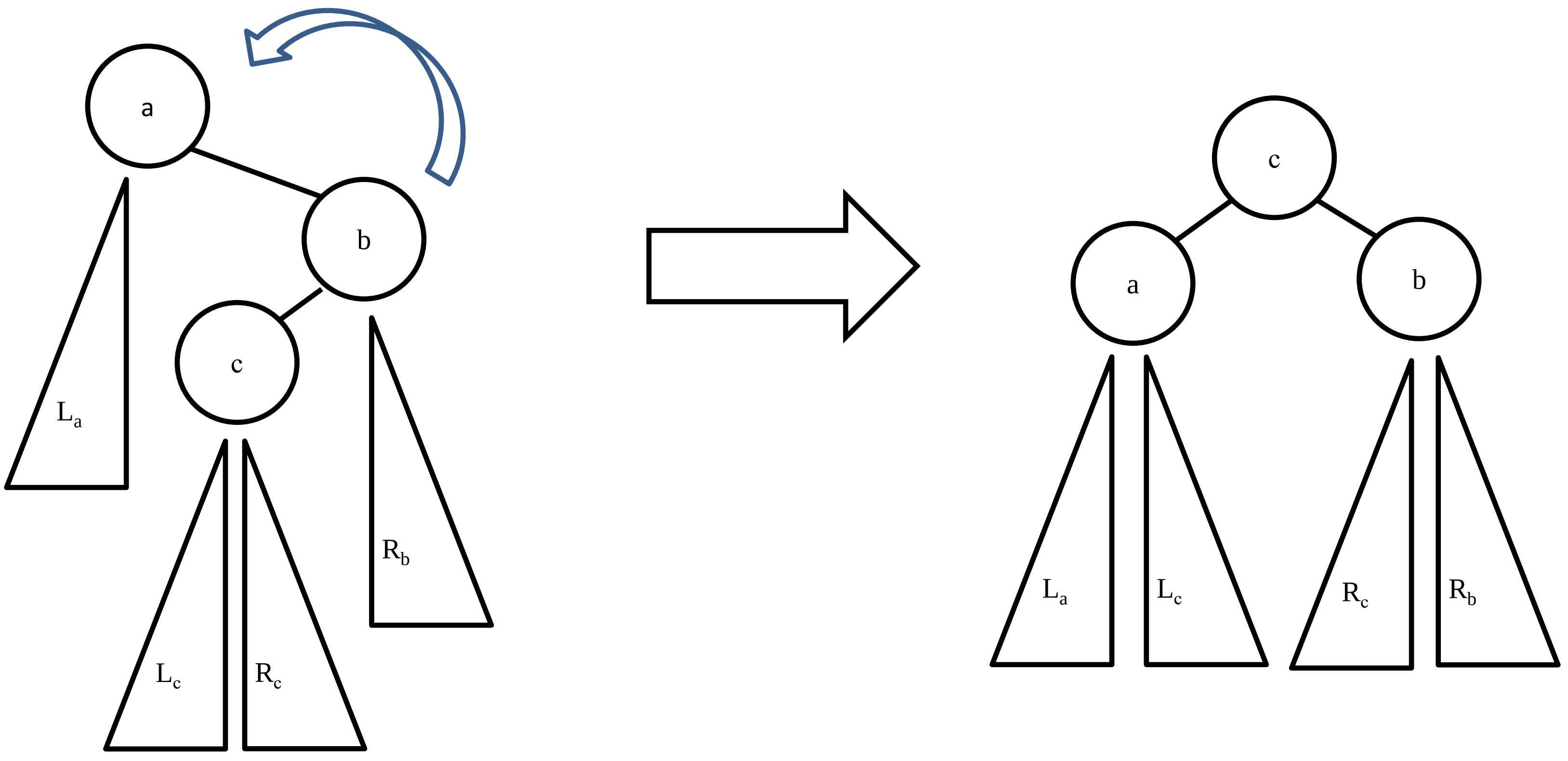
На примере, показанном на рис. видно, что малое правое вращение заключается в повороте относительно вершины *b*, при котором сама вершина *b* становится корнем, ее правым потомком становится вершина *a*, а бывшее правое поддерево R_b отходит вершине *a* и становится ее левым поддеревом. Таким образом, требования к отношениям между ключами узлов для бинарного дерева поиска сохраняются:

$\mathbf{L_b<b<R_b<a<R_a}$.

Малое левое вращение выполнятся аналогично.

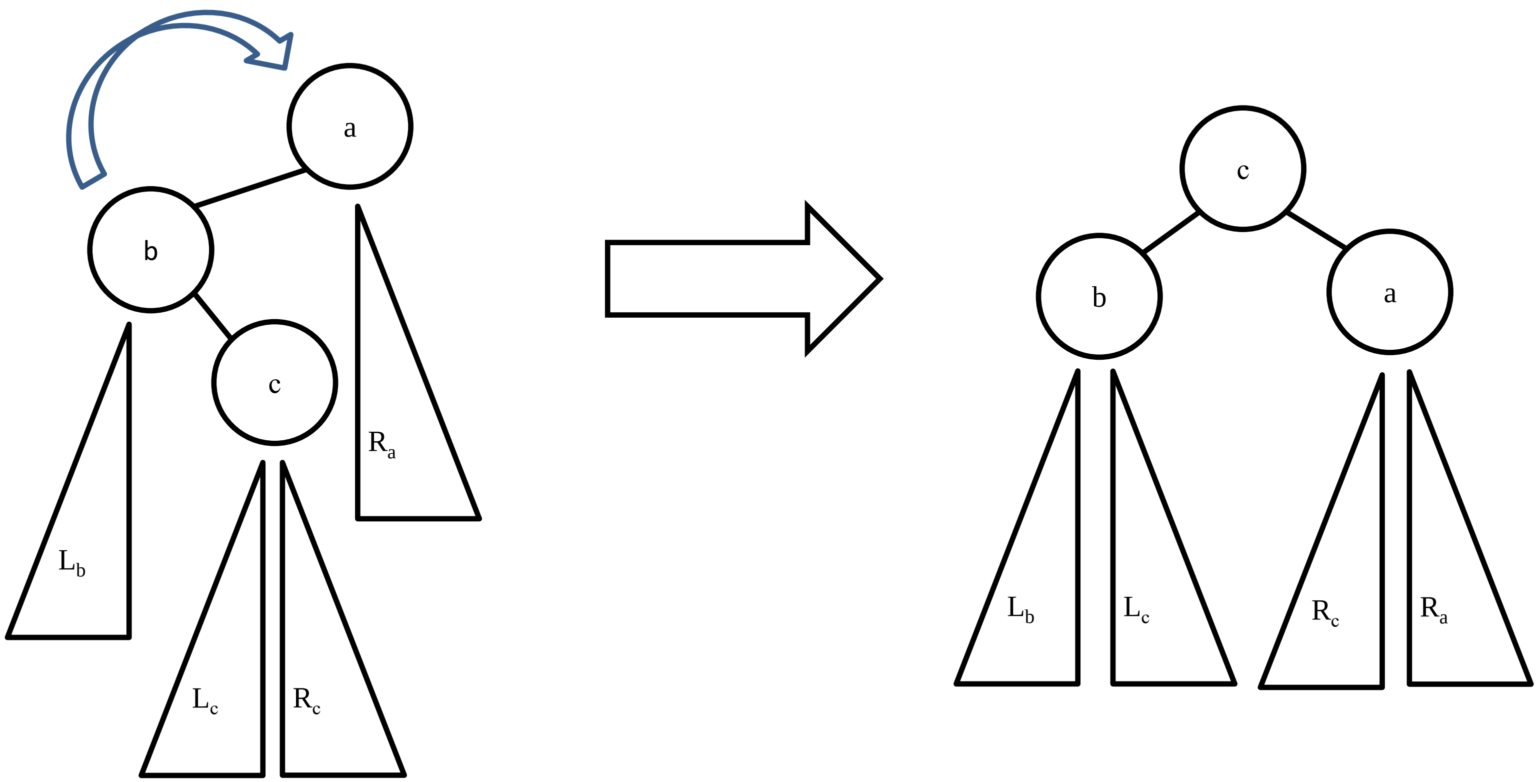
АВЛ-деревья: Большое левое вращение

если **(d(a) = -2) и (d(b) = 1)**, то выполняется большое левое вращение, показанное на рис. 3 (при этом коэффициент сбалансированности для узла *c* может принимать любые допустимые значения, на рисунке **d(c)=0**)



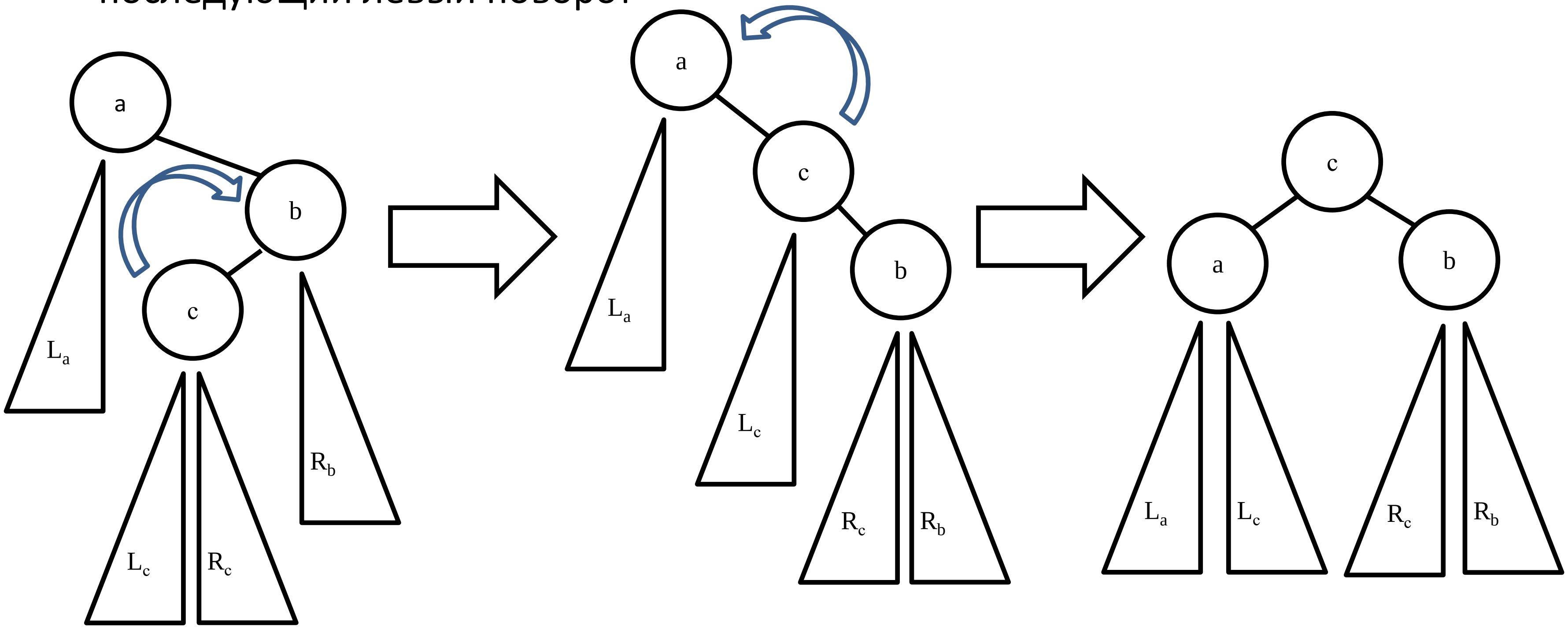
АВЛ-деревья: Большое правое вращение

если **(d(a) = 2) и (d(b) = -1)**, то выполняется большое правое вращение, показанное на рис. 4 (аналогично, коэффициент сбалансированности для узла *c* может принимать любые допустимые значения от -1 до 1)



АВЛ-деревья: Реализация больших поворотов

Технически большой левый поворот может быть выполнен, как два последовательно выполненных вращения: малый правый поворот и последующий левый поворот



При таком вращении отношения между ключами в дереве также сохраняются: **$L_a<a<L_c<c<R_c<b<R_b$**.

АВЛ-деревья

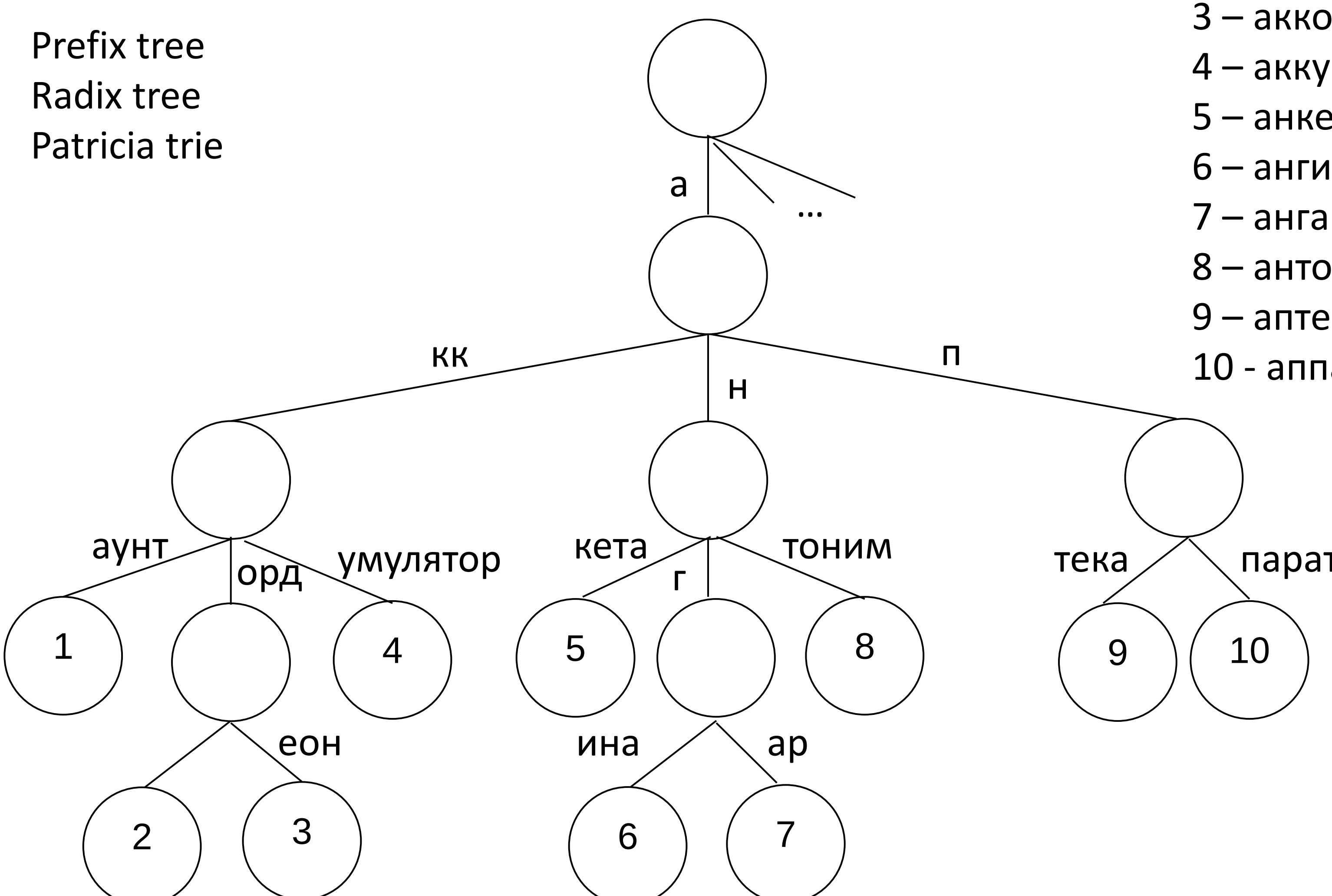
В целом для сохранения сбалансированности после вставки или удаления вершины выполняется подъем от вставленной или удаленной вершины к корню дерева с пересчетом показателей сбалансированности посещаемых вершин. Если показатель стал равным 2 или −2, то выполняется соответствующий поворот и, при необходимости, подъём продолжается. Если высота соответствующего поддерева не изменилась, то подъём можно остановить.
Отметим, что любой поворот выполняется за константное время O(1) и сохраняет свойства АВЛ-дерева. Выполнение основных операций требует $O(\log_2 n)$ сравнений.

Префиксные деревья

Префиксные деревья

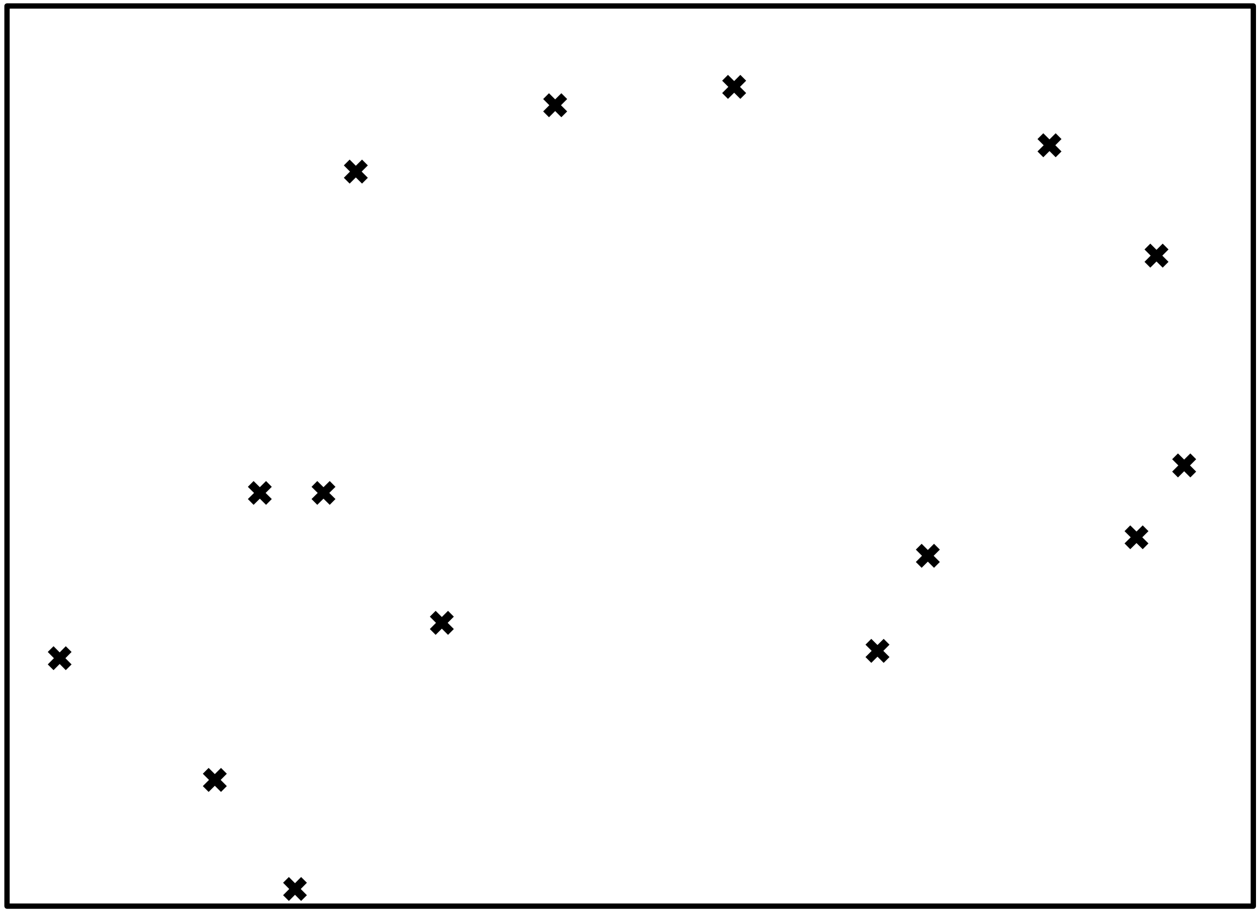
Prefix tree
Radix tree
Patricia trie

1 – аккаунт
2 – аккорд
3 – аккордеон
4 – аккумулятор
5 – анкета
6 – ангина
7 – ангар
8 – антоним
9 – аптека
10 - аппарат



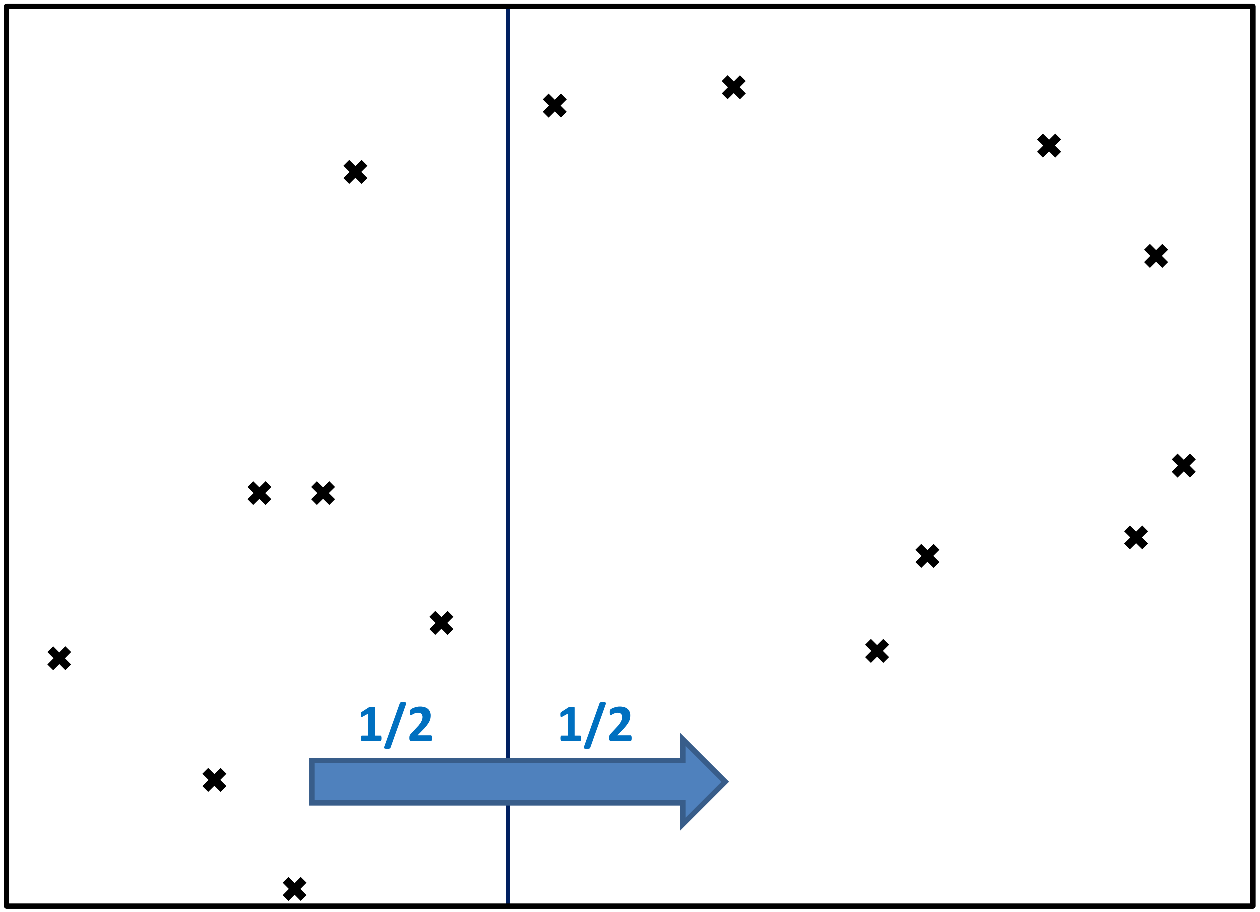
Поиск многомерных данных
kd-tree

kd-tree: исходные данные



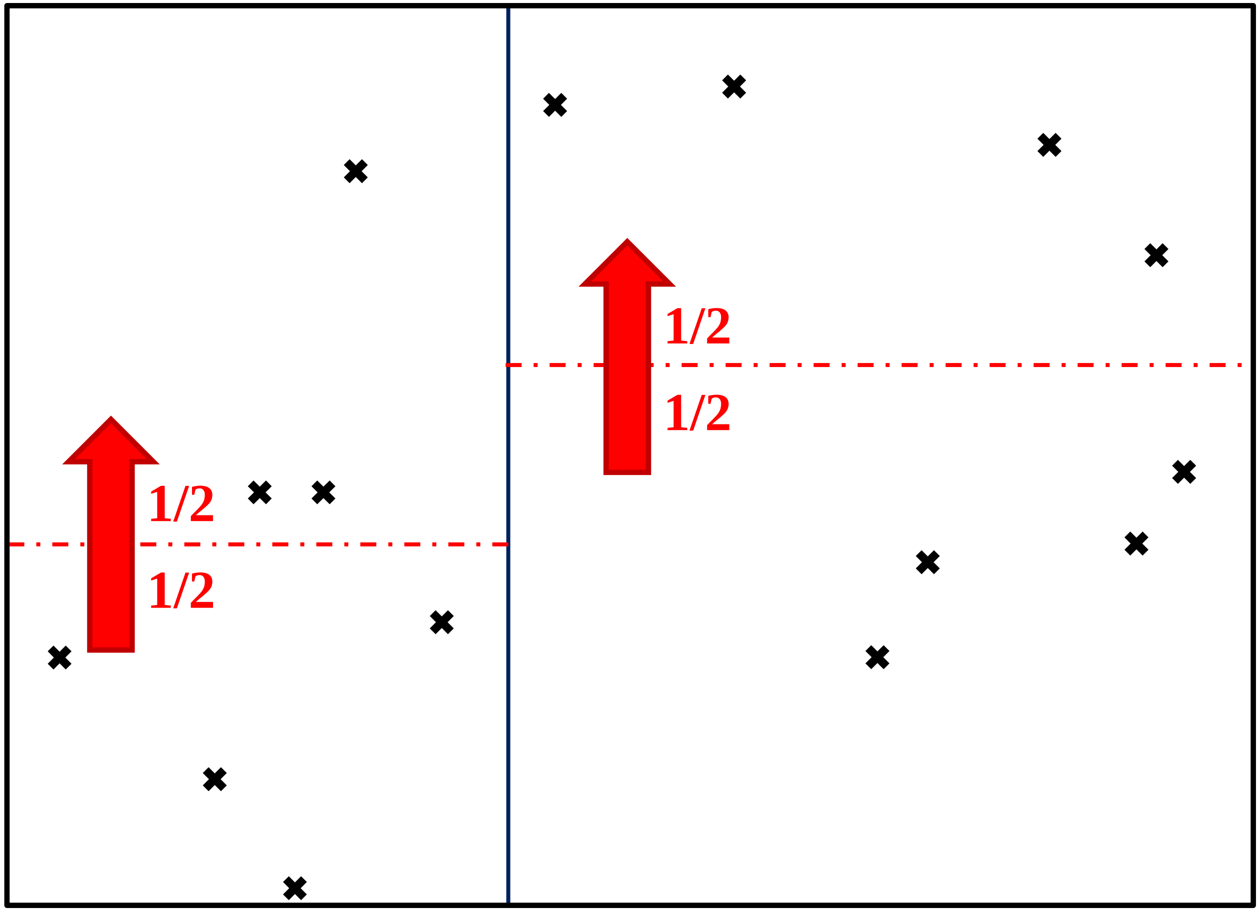
Различные уровни декомпозиции данных:
сплошная линия (темно-синий) – первый уровень
штрих-пунктирные линии (красный) – второй уровень
пунктирные линии (зеленый) – третий уровень

kd-tree: шаг 1



Различные уровни декомпозиции данных:
сплошная линия (темно-синий) – первый уровень
штрих-пунктирные линии (красный) – второй уровень
пунктирные линии (зеленый) – третий уровень

kd-tree: шаг 2



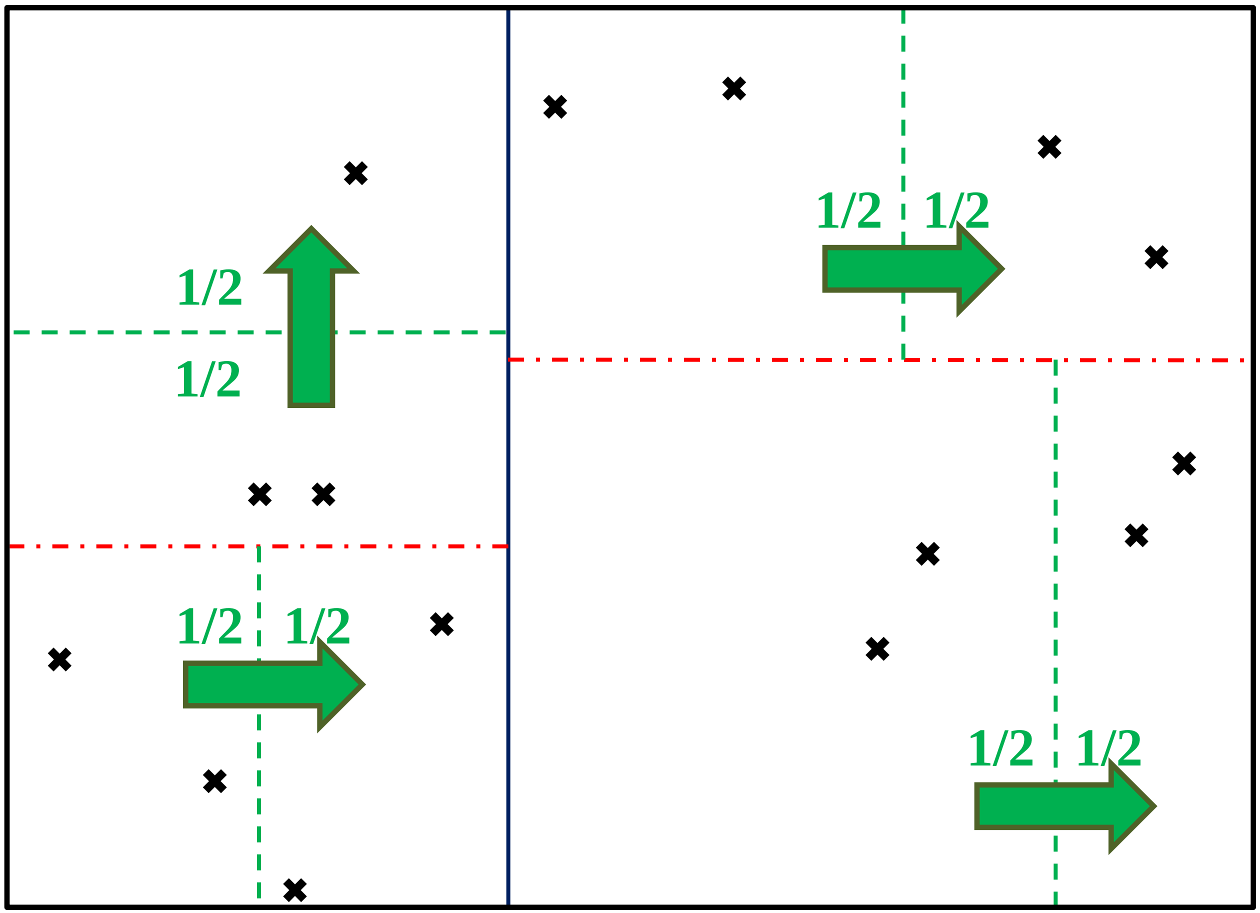
Различные уровни декомпозиции данных:

сплошная линия (темно-синий) – первый уровень

штрих-пунктирные линии (красный) – второй уровень

пунктирные линии (зеленый) – третий уровень

kd-tree: шаг 3



Различные уровни декомпозиции данных:

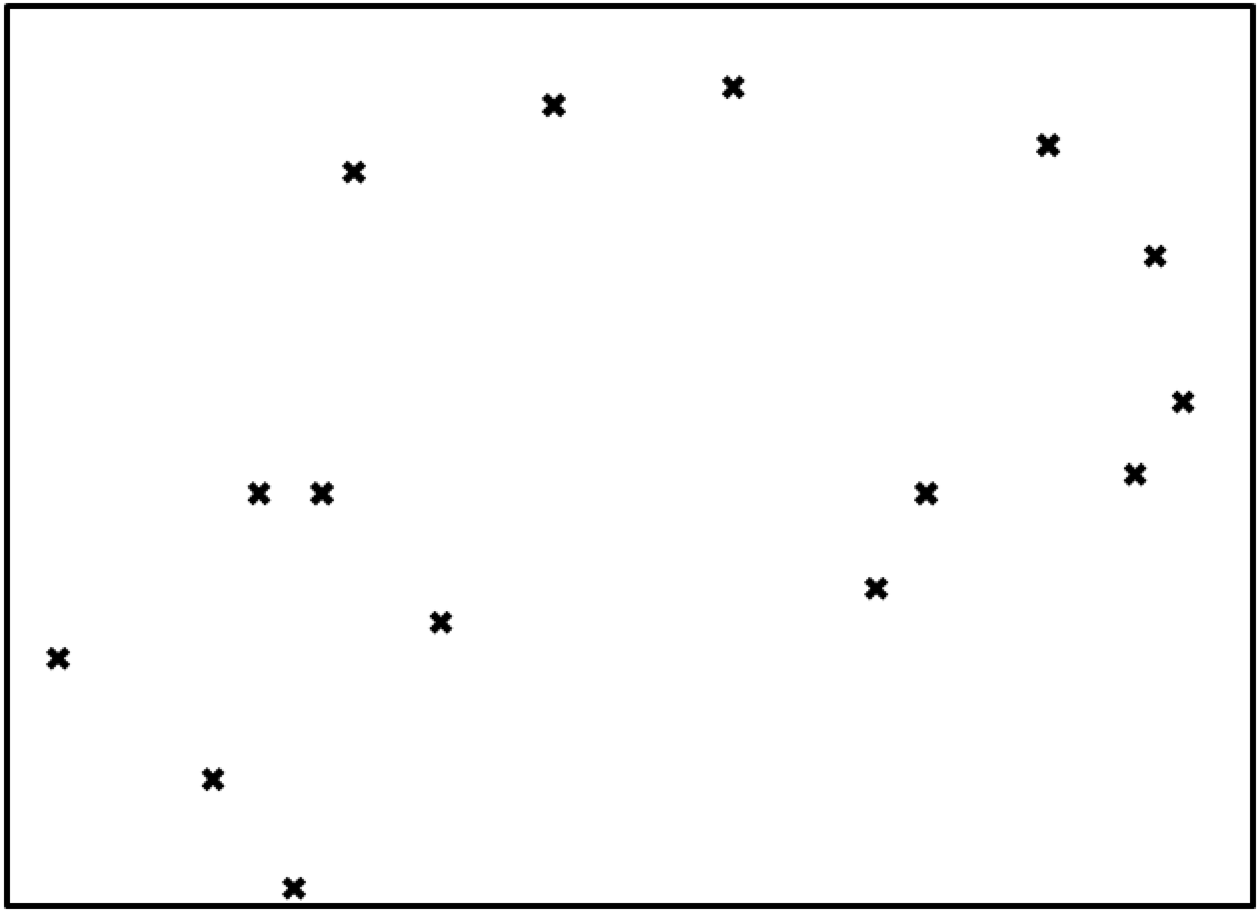
сплошная линия (темно-синий) – первый уровень

штрих-пунктирные линии (красный) – второй уровень

пунктирные линии (зеленый) – третий уровень

Поиск многомерных данных
vp-tree

vp-tree: исходные данные



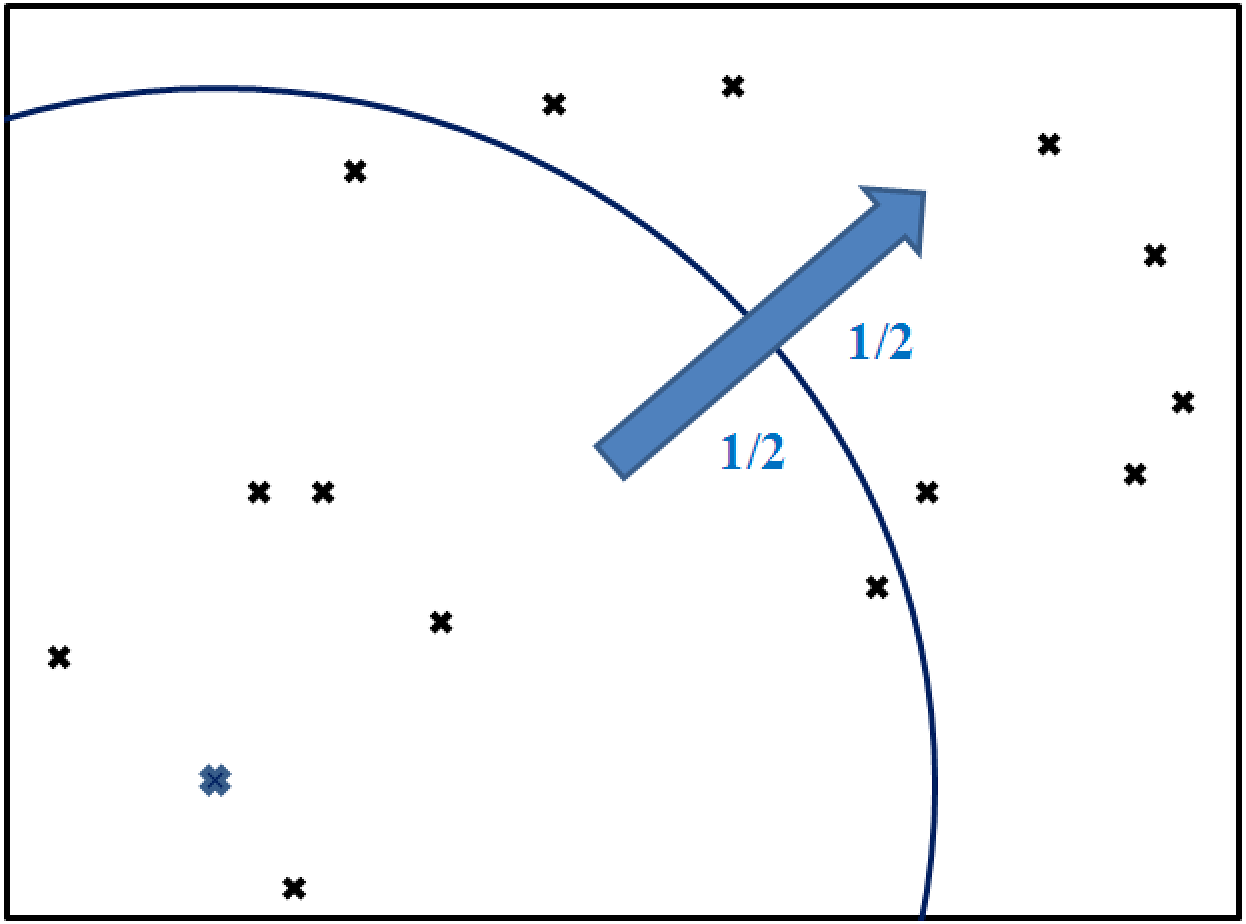
Различные уровни декомпозиции данных:

сплошная линия (темно-синий) – первый уровень

штрих-пунктирные линии (красный) – второй уровень

пунктирные линии (зеленый) – третий уровень

vp-tree: шаг 1



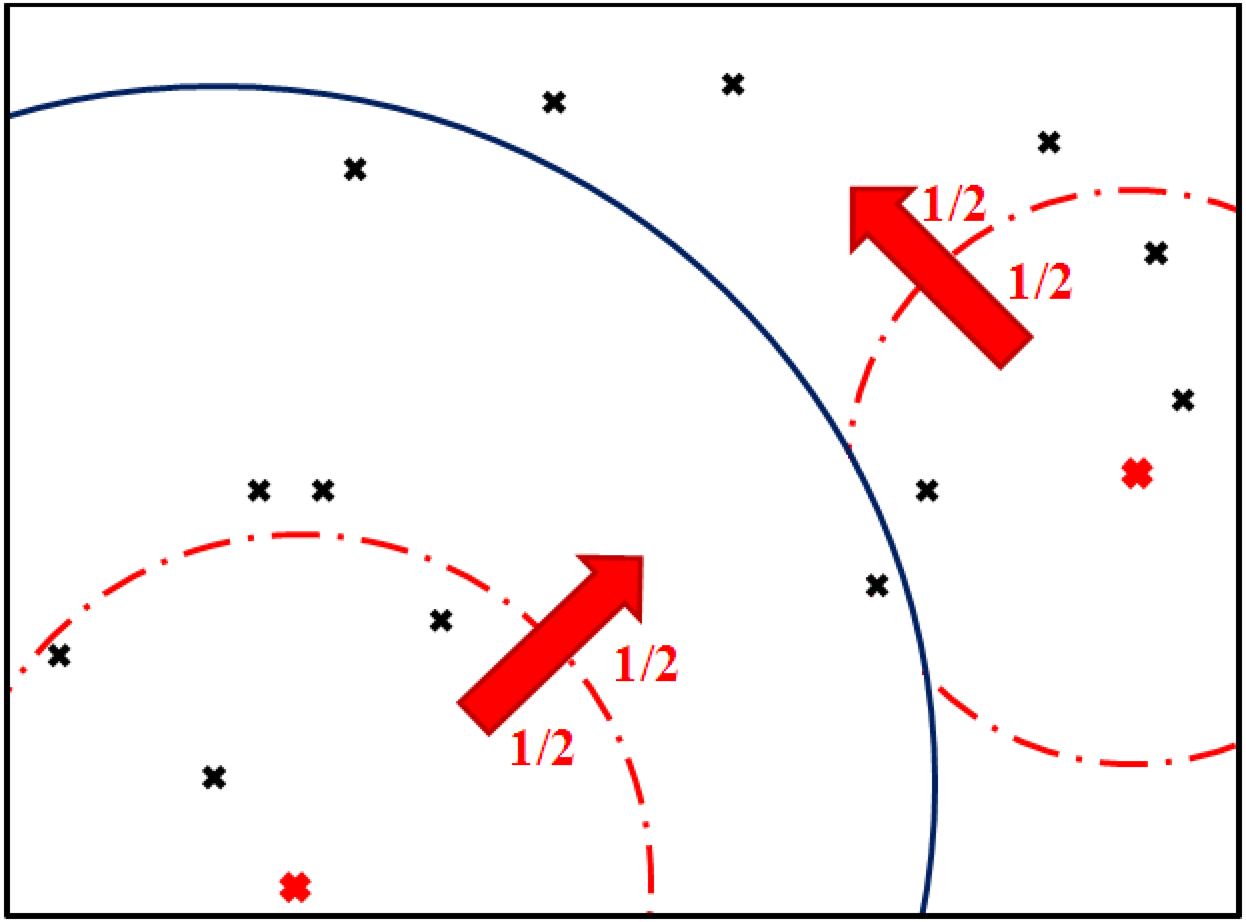
Различные уровни декомпозиции данных:

сплошная линия (темно-синий) – первый уровень

штрих-пунктирные линии (красный) – второй уровень

пунктирные линии (зеленый) – третий уровень

vp-tree: шаг 2



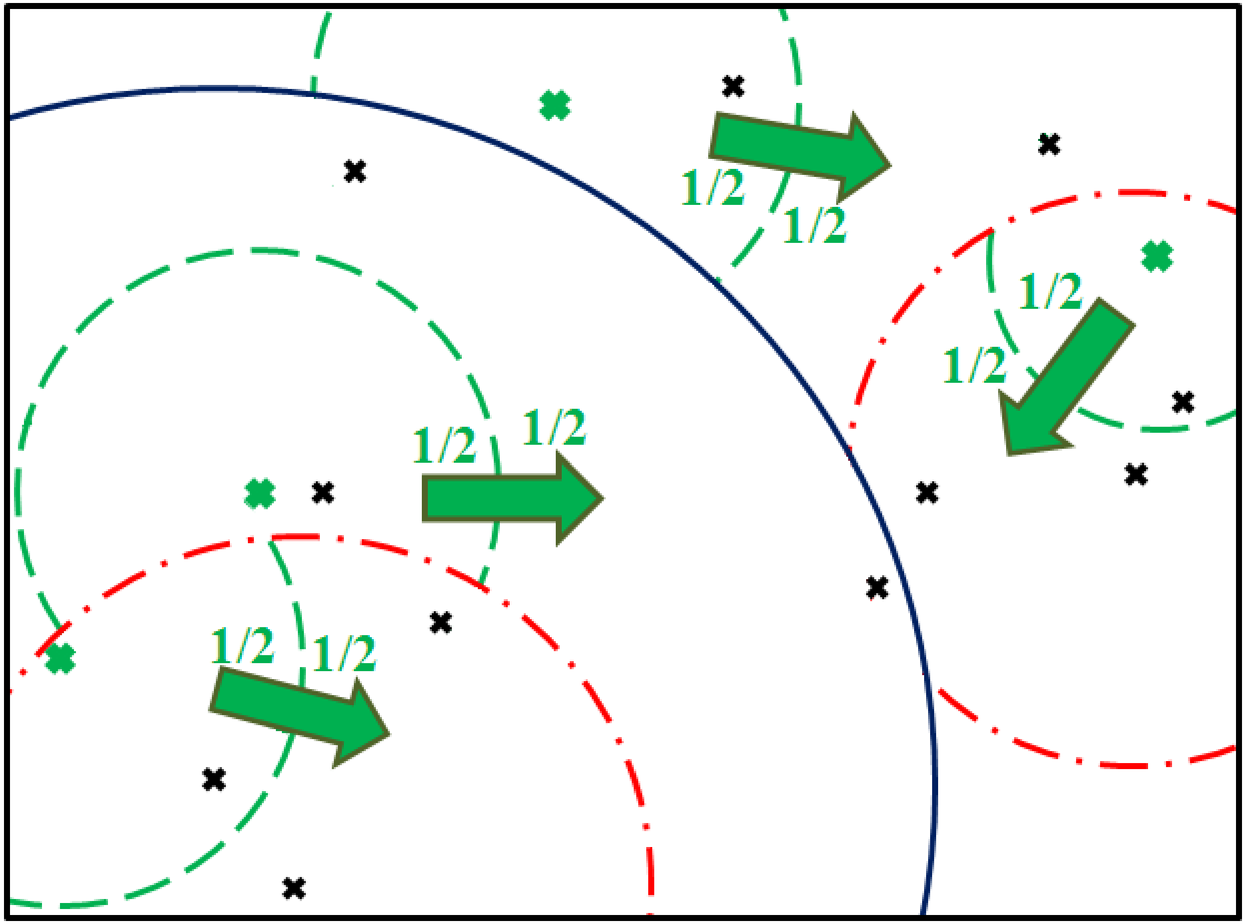
Различные уровни декомпозиции данных:

сплошная линия (темно-синий) – первый уровень

штрих-пунктирные линии (красный) – второй уровень

пунктирные линии (зеленый) – третий уровень

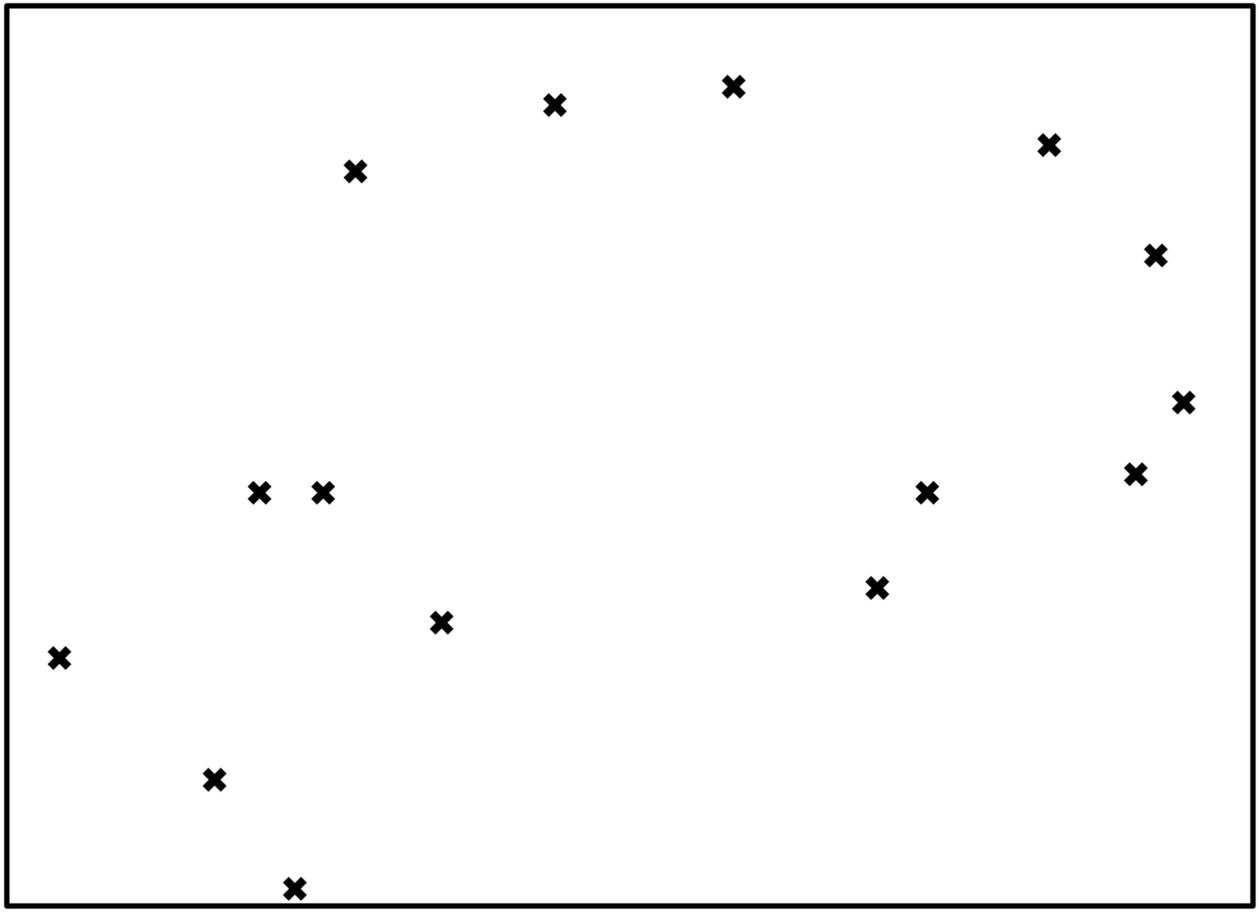
vp-tree: шаг 3



Различные уровни декомпозиции данных:
сплошная линия (темно-синий) – первый уровень
штрих-пунктирные линии (красный) – второй уровень
пунктирные линии (зеленый) – третий уровень

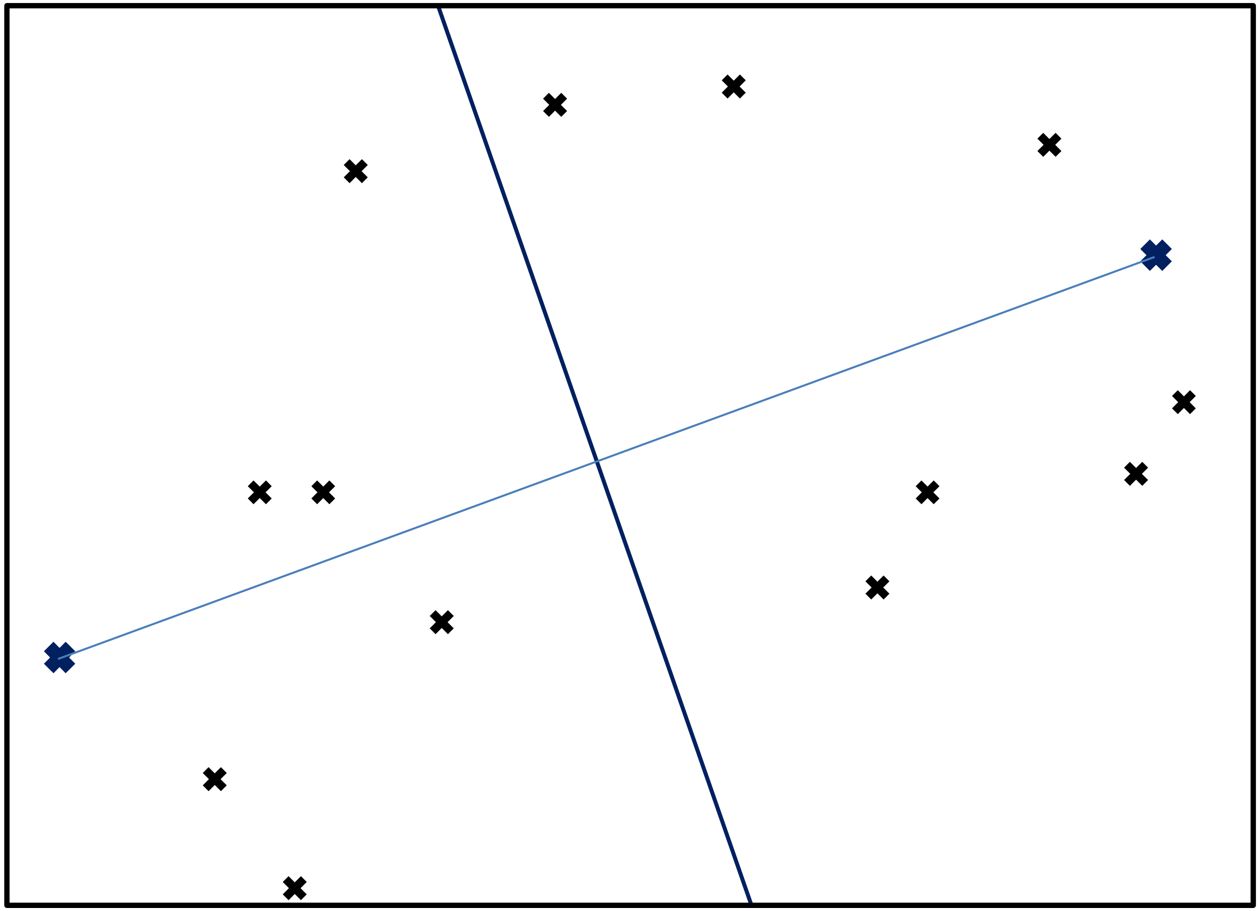
Поиск многомерных данных
ball-tree

ball-tree: исходные данные



Различные уровни декомпозиции данных:
сплошная линия (темно-синий) – первый уровень
штрих-пунктирные линии (красный) – второй уровень
пунктирные линии (зеленый) – третий уровень

ball-tree: шаг 1



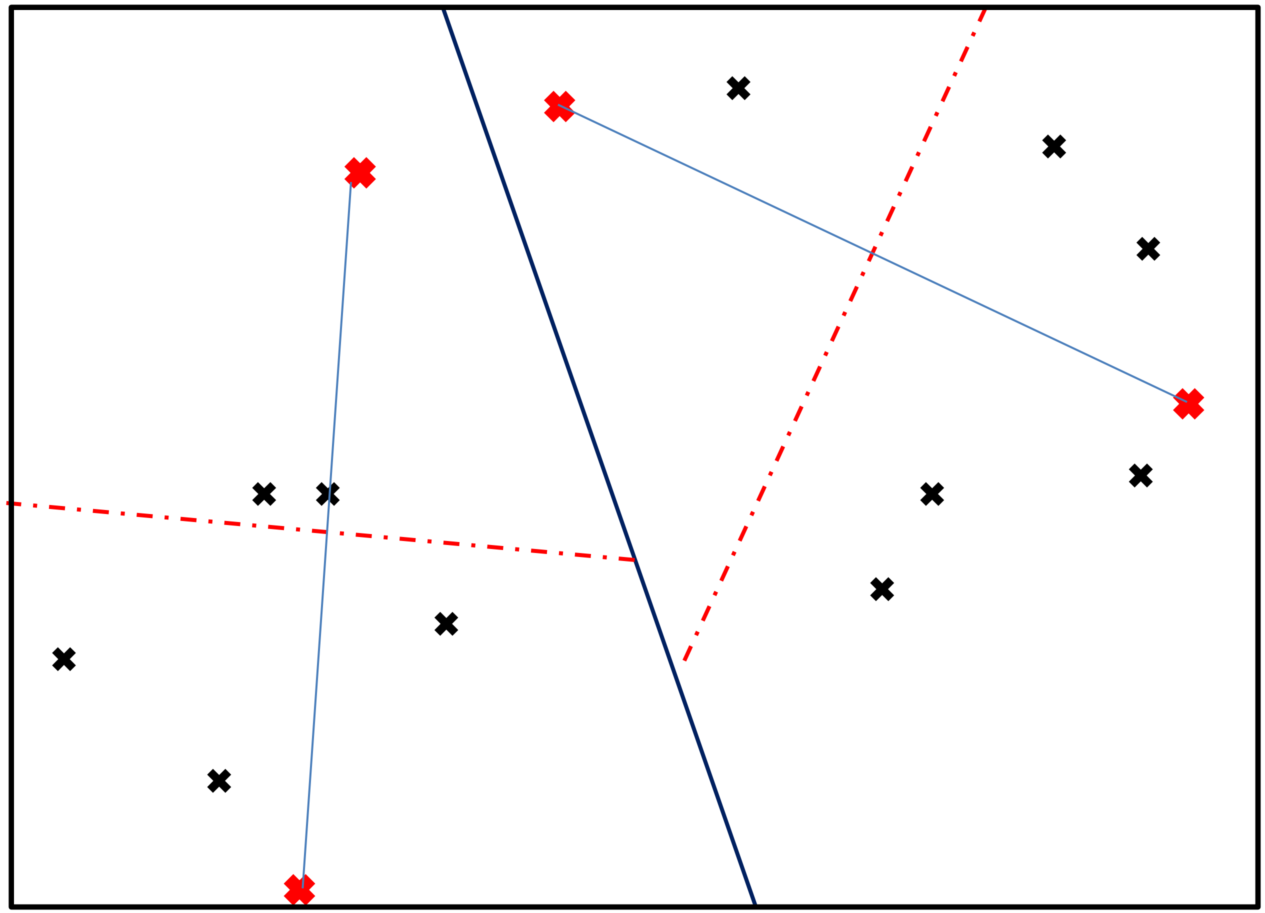
Различные уровни декомпозиции данных:

сплошная линия (темно-синий) – первый уровень

штрих-пунктирные линии (красный) – второй уровень

пунктирные линии (зеленый) – третий уровень

ball-tree: шаг 2



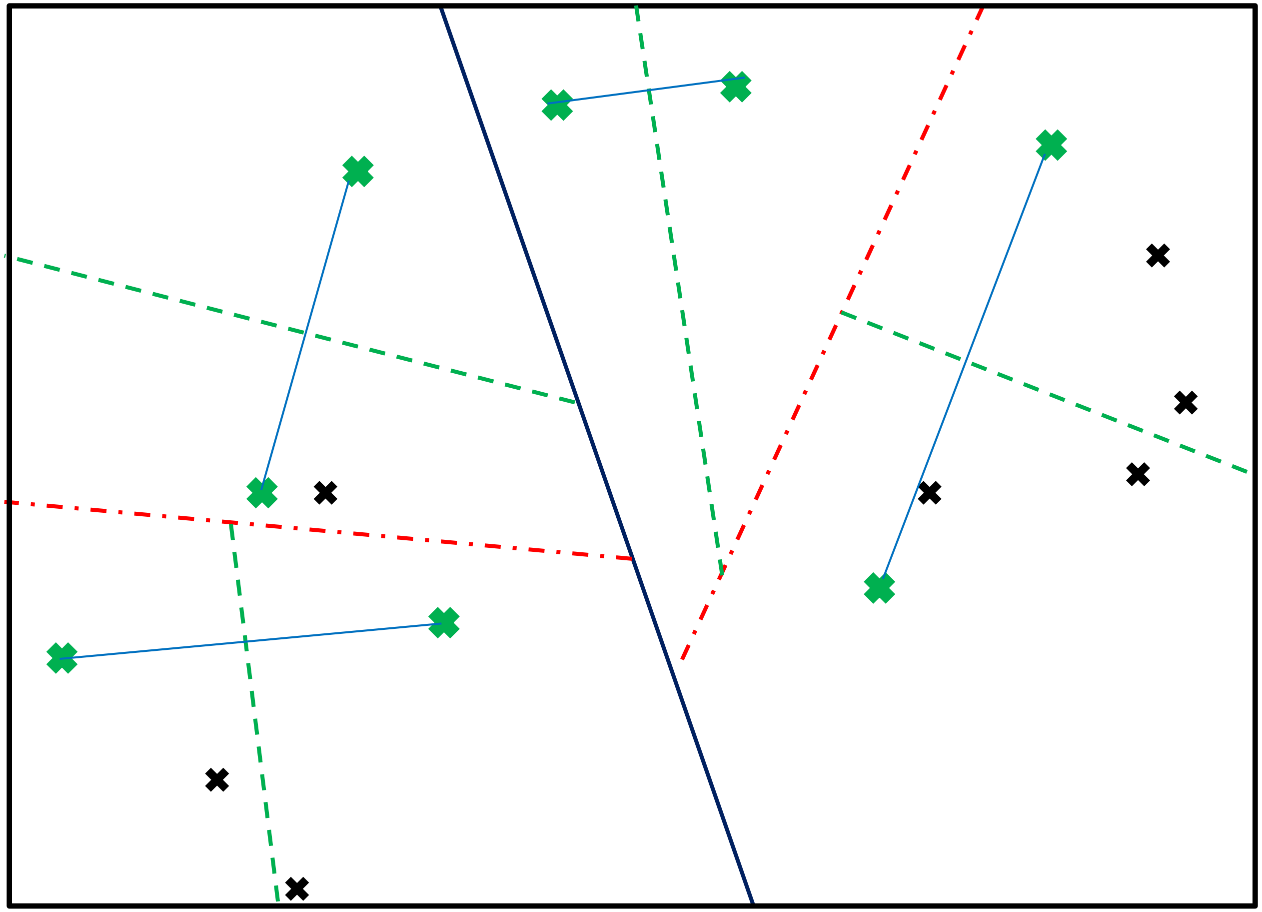
Различные уровни декомпозиции данных:

сплошная линия (темно-синий) – первый уровень

штрих-пунктирные линии (красный) – второй уровень

пунктирные линии (зеленый) – третий уровень

ball-tree: шаг 3



Различные уровни декомпозиции данных:

сплошная линия (темно-синий) – первый уровень

штрих-пунктирные линии (красный) – второй уровень

пунктирные линии (зеленый) – третий уровень