# Красно-чёрные деревья

ЛЕКЦИЯ 10

# Для чего нужны красно-чёрные деревья?

Может случится так, что узлы выстраиваются в линию без ветвления. Поскольку каждый узел больше узла, вставленного перед ним, каждый узел является правым потомком, поэтому все узлы располагаются по одну сторону от корня. Дерево получается предельно несбалансированным. Если вставить элементы, упорядоченные по убыванию, то каждый узел будет левым потомком своего родителя, а дерево окажется несбалансированным с другой стороны. Происходит вырождение до O(N). При отсутствии ветвей дерево фактически превращается в связанный список. Двумерная структура данных становится одномерной. К сожалению, как и в случае со связанным списком, для поиска нужного элемента теперь придется просмотреть (в среднем) половину элементов.

# Особенности красно-чёрного дерева

- Каждый узел окрашен в определенный цвет.
- В процессе вставки и удаления выполняются различные правила взаимного расположения этих цветов.

В красно-черном дереве каждый узел окрашен либо в красный, либо в черный цвет. Цвета выбраны произвольно; например, их можно было бы заменить синим и желтым. Более того, сама концепция «цветов» выбрана произвольно: с таким же успехом можно было разделить узлы на тяжелые или легкие, или на узлы-ин и узлы-янь. Тем не менее цвета достаточно удобны в качестве меток. Для представления информации о цвете в класс узла включается поле данных, которое может относиться к логическому типу

# Особенности красно-чёрного дерева

При вставке (или удалении) узла должны выполняться некоторые правила, которые мы будем называть красно-черными правилами. Если эти правила выполняются, дерево остается сбалансированным. Краткий перечень этих правил:

1. Каждый узел окрашен в красный или черный цвет.
2. Корень всегда окрашен в черный цвет.
3. Если узел красный, то его потомки должны быть черными (хотя обратное не всегда истинно).
4. Все пути от корня к узлу или пустому потомку должны содержать одинаковое количество черных узлов.

# Особенности красно-чёрного дерева

Пустым потомком, упомянутым в правиле 4, называется место возможного присоединения потомка к не-листовому узлу. Другими словами, это отсутствующий левый потомок узла, имеющего правого потомка, или отсутствующий правый потомок узла с левым потомком. Не беспокойтесь, скоро все станет более понятным. Количество черных узлов на пути от корня к листу называется черной высотой. В другой формулировке правило 4 может гласить, что все пути от корня к узлу должны иметь одинаковую черную высоту

Предположим, вы видите (или приложение сообщает вам) о нарушении красночерных правил. Как исправить ситуацию и привести дерево к виду, удовлетворяющему правилам? Возможны два (и только два!) действия:

Изменение цвета узлов ИЛИ Выполнение поворотов.

В приложении под изменением цвета узла понимается изменение цвета его контура (а не заливки). Поворотом называется такая перегруппировка узлов, в результате которой дерево становится более сбалансированным

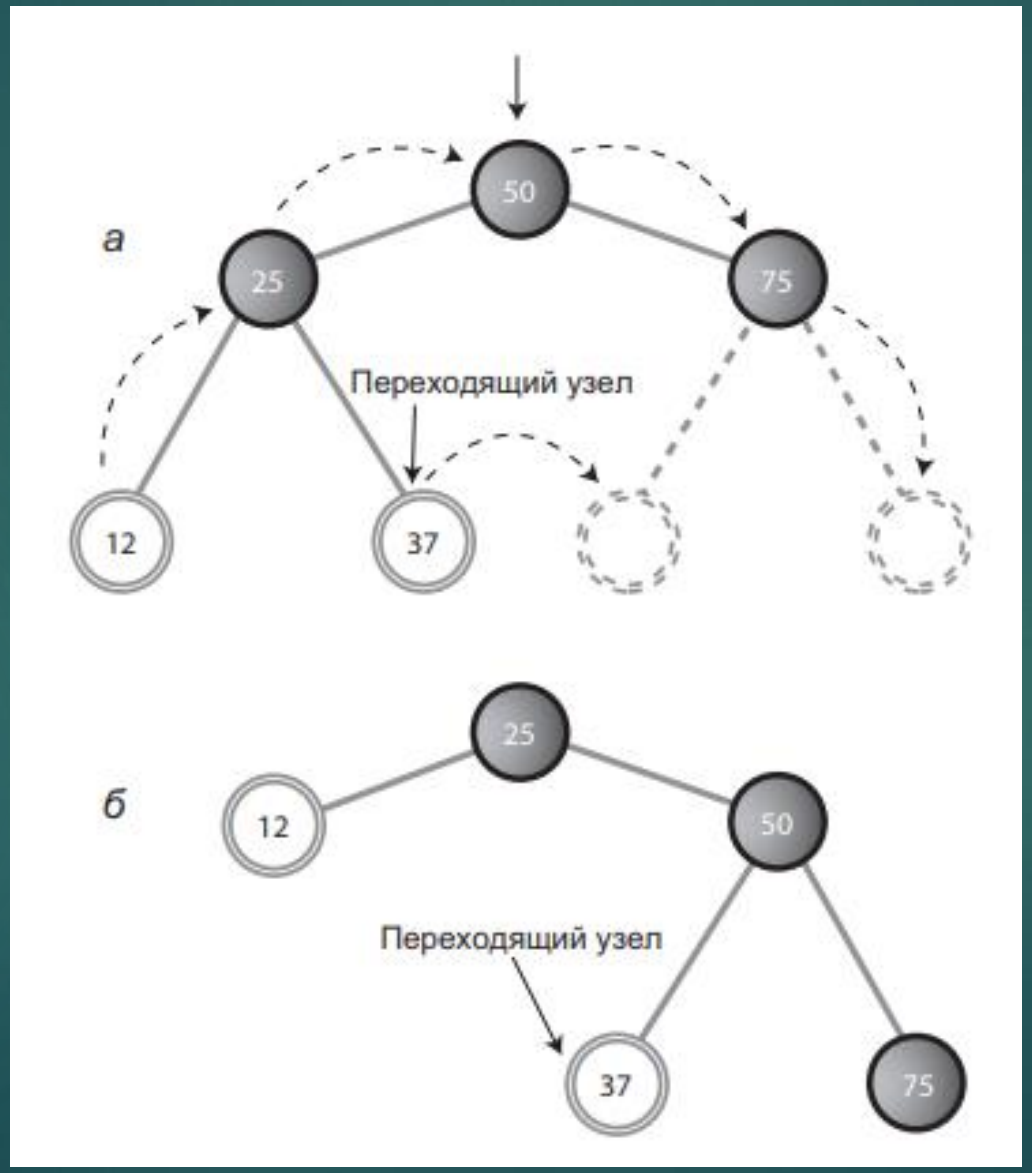
# Повороты

Чтобы сбалансировать дерево, необходимо выполнить физическую перекомпоновку его узлов. Например, если все узлы находятся в левой части дерева, некоторые из них необходимо переместить на правую сторону. Задача решается при помощи поворотов. В этом разделе вы узнаете, что такое повороты и как они выполняются. Поворот должен: Ê поднять одни узлы и опустить другие, чтобы обеспечить сбалансированность дерева; Ê обеспечить соблюдение характеристик дерева двоичного поиска. Вспомните, что в дереве двоичного поиска ключи левых потомков любого узла меньше ключа самого узла, а ключи правых потомков больше либо равны ключу узла.

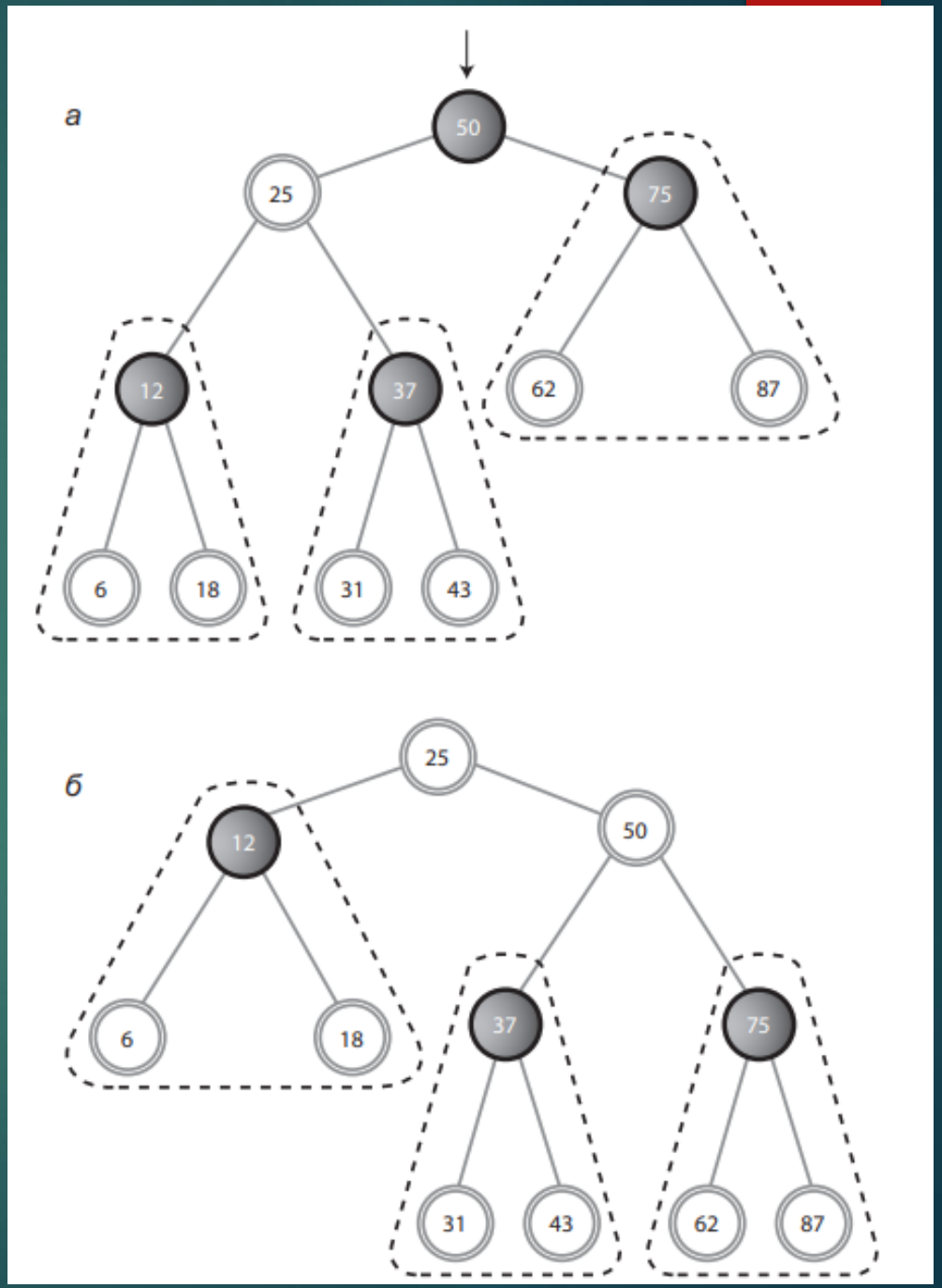
# Повороты

Если бы поворот не поддерживал действительность дерева двоичного поиска, то пользы от него было бы немного; ведь работа алгоритма поиска (см. предыдущую главу) зависит от конфигурации дерева поиска. Обратите внимание: цветовые правила и изменения в цветах узлов только помогают решить, когда должен выполняться поворот. Возня со сменой цветов мало что дает сама по себе; основная работа выполняется поворотами. Цветовые правила можно сравнить с общими правилами строительства зданий (например, «внешние двери должны открываться вовнутрь»), а повороты — с самими строительными операциями.

# Реализация поворота



# Реализация поворота



# Процесс вставки

Алгоритм вставки в красно-черное дерево начинает свою работу практически с того же, что и алгоритм вставки в обычное дерево двоичного поиска: он проходит от корня к месту вставки узла, поворачивая налево или направо в зависимости от результата сравнения ключа текущего узла с искомым ключом. Однако в красно-черном дереве переход к точке вставки усложняется переключениями цветов и поворотами. Переключения были представлены в эксперименте 3; пора рассмотреть их более подробно. Представьте, что алгоритм вставки перемещается вниз по дереву, поворачивая налево или направо на каждом узле, подыскивая место для вставки нового узла. Чтобы цветовые правила не нарушались, алгоритму приходится выполнять переключение цветов при необходимости. Правило выглядит так: каждый раз, когда алгоритм вставки встречает черный узел с двумя красными потомками, он должен перекрасить потомков в черный цвет, а родителя в красный (если только родитель не является корневым узлом, который всегда остается черным).

# Эффективность

В красно-черных деревьях, как и в обычных деревьях двоичного поиска, операции поиска, вставки и удаления выполняются за время O(log2N). Время поиска в красно-черных деревьях практически неотличимо от времени поиска в обычном дереве, потому что при поиске красно-черные характеристики не используются. Единственное отличие — небольшое увеличение затрат памяти на хранение признака красно-черного цвета (логическая переменная).

# Организация вставки

В класс Node необходимо включить поле для хранения цвета (которое может относиться к типу boolean).
При перемещении вниз к точке вставки проверьте, является ли текущий узел черным узлом с двумя красными потомками. Если это условие выполнено, измените цвета всех трех узлов (если только родитель не является корневым узлом — в этом случае он остается черным). После переключения цветов проверьте, что правило 3 не нарушено. Если оно нарушается, выполните соответствующие повороты: один для внешнего внука, два для внутреннего. Достигнув листового узла, вставьте новый узел, как в программе tree.java, убедившись в том, что этот узел красный. Снова проверьте конфликты типа «красныйкрасный» и выполните все необходимые повороты. Как ни странно, программной реализации не нужно вычислять черные высоты разных частей дерева (хотя, возможно, эта информация пригодится в ходе отладки). Проверять нужно только нарушения правила 3, красного родителя с красным потомком, которые могут осуществляться локально (в отличие от черных высот из правила 4, требующих более сложных подсчетов). Если выполнить переключения, изменения и повороты цветов, о которых говорилось ранее, черные высоты узлов выравниваются автоматически, а дерево остается сбалансированным.

# Итоги

- Дерево двоичного поиска должно быть сбалансированным, поскольку это гарантирует минимальное время поиска заданного узла.
- При вставке заранее отсортированных данных создается дерево с максимальной разбалансированностью, время поиска в котором составляет O(N).
- В красно-черной схеме балансировки каждому узлу присваивается новая характеристика: цвет (красный или черный). Ê Допустимое взаиморасположение узлов разных цветов определяется набором правил, называемых красно-черными правилами.
- Эти правила применяются при вставке (и удалении) узлов.
- Переключение цветов превращает черный узел с двумя красными потомками в красный узел с двумя черными потомками.

# Итоги

- При выполнении поворота один узел считается верхним.
- При правом повороте верхний узел перемещается в позицию правого потомка, а левый потомок занимает его прежнее место.
- При левом повороте верхний узел перемещается в позицию левого потомка, а правый потомок занимает его прежнее место.
- В ходе определения позиции для вставки нового узла выполняются переключения цветов, а в некоторых случаях и повороты. Переключение просто возвращает дерево к состоянию красно-черной правильности после выполнения вставки.
- После вставки нового узла снова производится проверка возможных конфликтов типа «красный-красный». Если будет обнаружено нарушение, выполняются повторы, возвращающие дерево к состоянию красно-черной правильности.

# Итоги

- В результате этих исправлений дерево становится сбалансированным (или по крайней мере ограничивается минимальной несбалансированностью).
- Добавление красно-черной балансировки в двоичное дерево лишь незначительно ухудшает среднее быстродействие и избегает худшего сценария со вставкой уже изначально отсортированных данных.