Алгоритмы и структуры данных

Лекция 13. Красно-черные деревья

Антон Штанюк (к.т.н, доцент)

17 мая 2021 г.

Нижегородский государственный технический университет им. Р.Е. Алексеева
Институт радиоэлектроники информационных технологий
Кафедра "Компьютерные технологии в проектировании и производстве"

Содержание

Красно-черные деревья

Понятие красно-черного дерева

Красно-черные деревья - один из способов балансировки деревьев.

Название происходит от стандартной раскраски узлов таких деревьев в красный и **черный** цвета. Цвета узлов используются при балансировке дерева. Во время операций вставки и удаления поддеревья может понадобиться повернуть, чтобы достигнуть сбалансированности дерева.

Оценкой как среднего время, так и наихудшего является $O(logn)$.

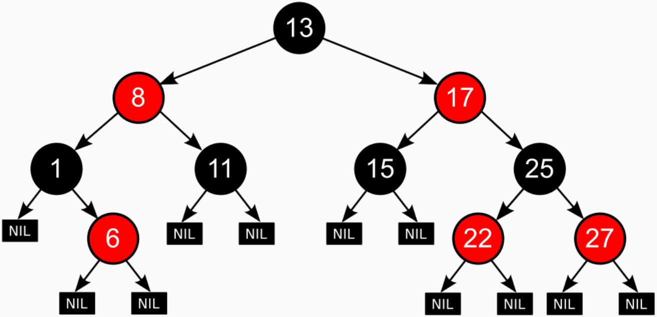
Понятие красно-черного дерева

Изобретателем красно-чёрного дерева считают Рудольфа Байера. Название «красно-чёрное дерево» структура данных получила в статье Л. Гимбаса и Р. Седжвика (1978).

Красно-чёрные деревья являются одними из наиболее активно используемых на практике самобалансирующихся деревьев поиска. В частности, контейнеры set и map в большинстве реализаций библиотеки STL языка C++, класс TreeMap языка Java, так же, как и многие другие реализации ассоциативного массива в различных библиотеках, основаны на красно-чёрных деревьях.

Понятие красно-черного дерева

Красно-чёрное дерево — двоичное дерево поиска, в котором каждый узел имеет атрибут цвета.



Понятие красно-черного дерева

Свойства красно-черного дерева:

1. Узел может быть либо красным, либо **чёрным** и имеет двух потомков;
2. Корень — как правило **чёрный**.
3. Все листья, не содержащие данных — **чёрные**.
4. Оба потомка каждого красного узла — **чёрные**.
5. Любой простой путь от узла-предка до листового узла-потомка содержит одинаковое число **чёрных** узлов.

Понятие красно-черного дерева

Листовые узлы красно-чёрных деревьев не содержат данных, благодаря чему не требуют выделения памяти — достаточно записать в узле-предке в качестве указателя на потомка нулевой указатель (NIL).

Понятие красно-черного дерева

Количество черных узлов на ветви от корня до листа называется **черной высотой дерева**. Перечисленные свойства гарантируют, что самая длинная ветвь от корня к листу не более чем вдвое длиннее любой другой ветви от корня к листу.

Пусть у нас есть красно-черное дерево. Черная высота равна bh (black height).

Если путь от корневого узла до листового содержит минимальное количество красных узлов (т.е. ноль), значит этот путь равен bh.

Если же путь содержит максимальное количество красных узлов (bh в соответствии со свойством 4), то этот путь будет равен $2bh$.

То есть, пути из корня к листьям могут различаться не более, чем вдвое ($h <= 2log(n+1)$), где h — высота поддерева), этого достаточно, чтобы время выполнения операций в таком дереве было $O(log(n))$

Вставка узла в дерево

Вставка узла в дерево

Чтобы вставить узел, мы сначала ищем в дереве место, куда его следует добавить. Новый узел всегда добавляется как лист, поэтому оба его потомка являются NIL-узлами и предполагаются черными. После вставки красим узел в красный цвет. После этого смотрим на предка и проверяем, не нарушается ли красно-черное свойство. Если необходимо, мы перекрашиваем узел и производим поворот, чтобы сбалансировать дерево.

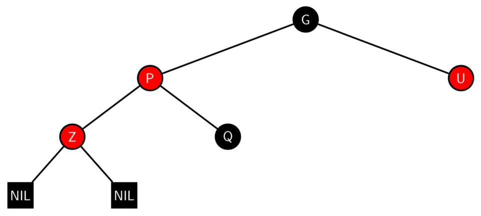
Вставка узла в дерево

1. Добавляем узел красного (R) цвета.
2. Если это корень, то он перекрашивается в черный цвет (B) и свойства не нарушаются.
3. Если родитель для вставляемого узла имеет цвет B, то свойства не нарушаются.
4. Если родитель для вставляемого узла имеет цвет R, то свойства нарушены.

Вставка. Случай 1. ”Дядя” красный

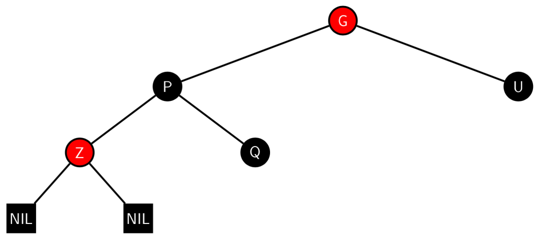
Мы добавляем в дерево узел Z, который автоматически делается красным.

В данном случае нужно выполнить перекрашивание ”родителей” и ”деда”



Вставка. Случай 1. ”Дядя” красный

Результат перекрашивания:

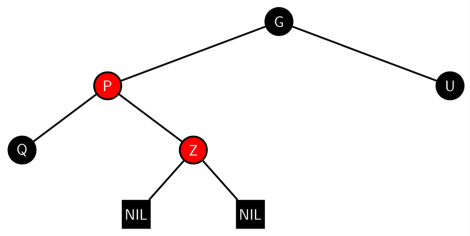


Дедушка G теперь может нарушить свойства 2 (Корень — чёрный) или 4 (Оба потомка каждого красного узла — чёрные) (свойство 4 может быть нарушено, так как родитель G может быть красным). Чтобы это исправить, вся процедура рекурсивно выполняется на G.

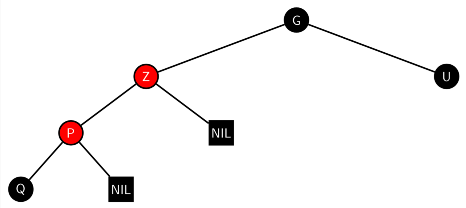
Вставка. Случай 2. ”Дядя” черный

Дядя черный, добавляемый узел Z - справа от родителя P.

В этом случае может быть произведен поворот дерева, который меняет роли текущего узла Z и его предка P.



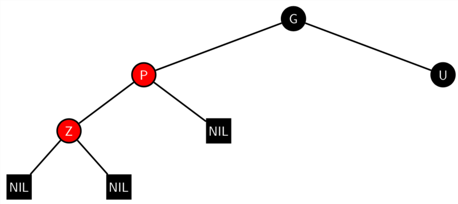
Результат перекрашивания:



Вращение приводит к тому, что некоторые пути (в поддереве Q на схеме) проходят через узел Z, чего не было до этого. Свойство 4 всё ещё нарушается, но теперь задача сводится к Случаю 3.

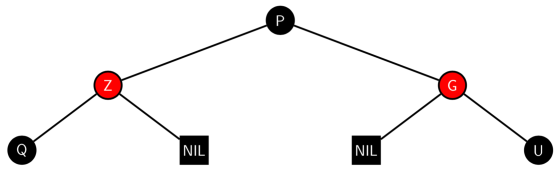
Дядя черный, добавляемый узел Z - слева от родителя P.

В этом случае выполняется поворот дерева на G. В результате получается дерево, в котором бывший родитель P теперь является родителем и текущего узла Z и бывшего дедушки G.



Вставка. Случай 3. ”Дядя” черный

Результат перекрашивания:

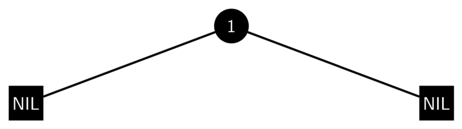


Формирование дерева

Вставка в дерево числовой последовательности

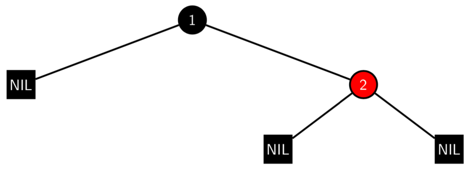
Создадим дерево и будем добавлять в него числа по возрастанию от 1 до 10.

Состояние дерева, после добавления 1:



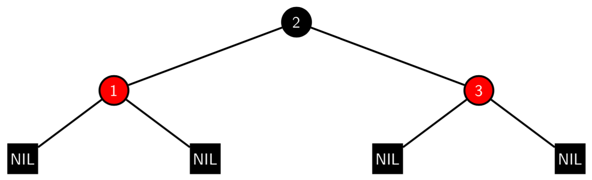
Вставка в дерево числовой последовательности

Состояние дерева, после добавления 2:



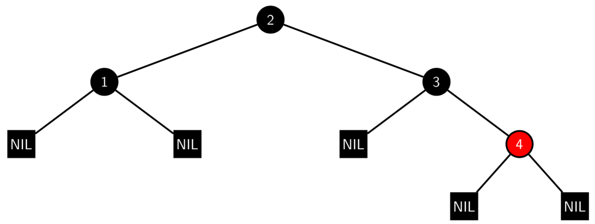
Вставка в дерево числовой последовательности

Состояние дерева, после добавления 3:



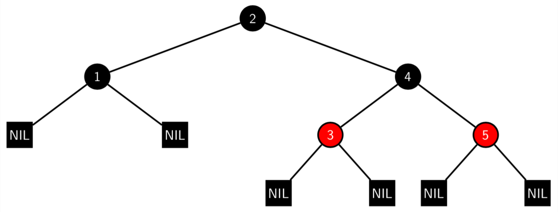
Вставка в дерево числовой последовательности

Состояние дерева, после добавления 4:



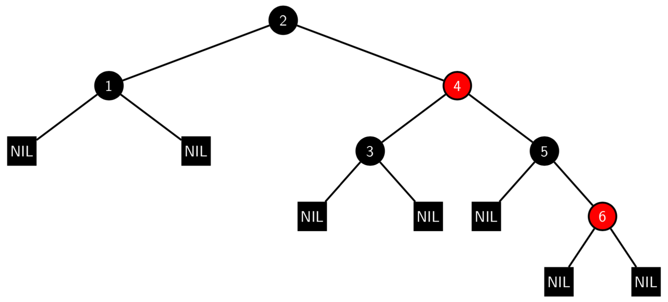
Вставка в дерево числовой последовательности

Состояние дерева, после добавления 5:



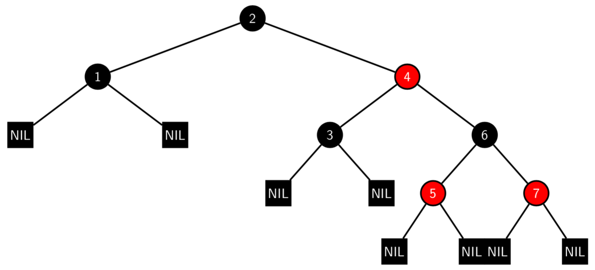
Вставка в дерево числовой последовательности

Состояние дерева, после добавления 6:



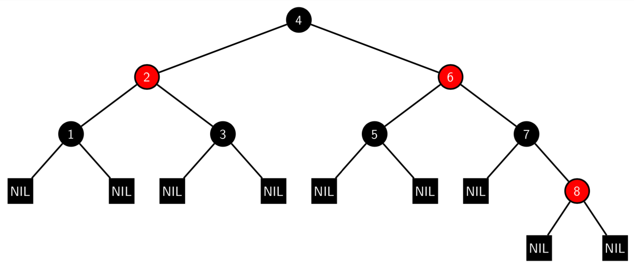
Вставка в дерево числовой последовательности

Состояние дерева, после добавления 7:



Вставка в дерево числовой последовательности

Состояние дерева, после добавления 8:

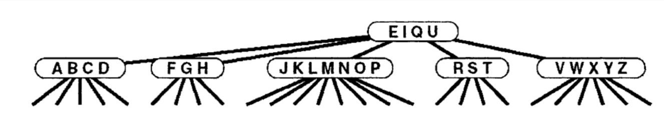


B - деревья

В - деревья

Особой разновидностью сбалансированных деревьев являются B-деревья.

B-дерево порядка M - это дерево, которое либо пусто, либо состоит их K-узлов с K−1 ключами и K связями с деревьями, представляющими каждый из K ограниченных ключами интервалов



B - деревья

B-деревья представляют собой сбалансированные деревья поиска, созданные специально для эффективной работы с дисковой памятью. Они отличаются от красно-чёрных деревьев тем, что узлы могут иметь до тысячи потомков, то есть высокую степень ветвления.

AVL деревья

Список литературы

Список литературы i

- Кормен Т.,Лейзерсон Ч., Ривест Р.
  Алгоритмы: построение и анализ
  МЦНМО, Москва, 2000
- **Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ.**
  2-е изд. — М.: «Вильямс», 2006
- **Википедия**
  Алгоритм
  http://ru.wikipedia.org/wiki/Алгоритм
- **Википедия**
  Список алгоритмов
  http://ru.wikipedia.org/wiki/Список_алгоритмов
- **Традиция**
  Задача коммивояжёра
  http://traditio.ru/wiki/Задача

Список литературы ii

- **Википедия**
  NP-полная задача
  http://ru.wikipedia.org/wiki/NP-полная
- **Серджвик Р.**
  Фундаментальные алгоритмы на C++. Части 1-4
  Diasoft,2001
- **Седжвик Р.**
  Фундаментальные алгоритмы на C. Анализ/Структуры данных/Сортировка/Поиск
  СПб.: ДиаСофтЮП, 2003
- **Седжвик Р.**
  Фундаментальные алгоритмы на C. Алгоритмы на графах
  СПб.: ДиаСофтЮП, 2003
- **Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы.**
  Издательский дом «Вильямс», 2000

Список литературы iii

- **Кнут Д.**
  Искусство программирования, том 1. Основные алгоритмы
  3-е изд. — М.: «Вильямс», 2006
- **Кнут Д.**
  Искусство программирования, том 2. Получисленные методы
  3-е изд. — М.: «Вильямс», 2007
- **Кнут Д.**
  Искусство программирования, том 3. Сортировка и поиск
  2-е изд. — М.: «Вильямс», 2007
- **Кнут Д.**
  Искусство программирования, том 4, выпуск 3. Генерация всех сочетаний и разбиений
  М.: «Вильямс», 2007

Список литературы iv

Кнут Д.
Искусство программирования, том 4, выпуск 4. Генерация всех деревьев. История комбинаторной генерации
М.: «Вильямс», 2007