Красно-чёрное дерево

Изобретателем красно-чёрного дерева считают немца Рудольфа Байера. Название «красно-чёрное дерево» структура данных получила в статье Л. Гимпаса и Р. Седжвика (1978). По словам Гимпаса, они использовали фломастеры двух цветов.. По словам Седжвика, красный цвет лучше всех смотрелся на лазерном принтере.

## Основной принцип устройства, особенности:

Красно-чёрное дерево — двоичное дерево поиска, в котором каждый узел имеет атрибут *цвет*, принимающий значения *красный* или *чёрный*. В дополнение к обычным требованиям, налагаемым на двоичные деревья поиска, к красно-чёрным деревьям применяются следующие требования:

1. Узел либо красный, либо черный.
2. Корень — чёрный.
3. Все листья (NIL) — чёрные.
4. Оба потомка каждого красного узла — чёрные. Нужно отметить, что у черного узла могут быть черные дочерние узлы. Красные узлы в качестве дочерних могут иметь только черные.
5. Всякий простой путь от данного узла до любого листового узла, являющегося его потомком, содержит одинаковое число чёрных узлов.

### Вставка:

Вставка начинается с добавления узла, точно так же, как и в обычном бинарном дереве поиска, и окрашивания его в красный цвет. Но если в бинарном дереве поиска мы всегда добавляем лист, в красно-чёрном дереве листья не содержат данных, поэтому мы добавляем красный внутренний узел с двумя чёрными потомками на место чёрного листа.

Что происходит дальше зависит от цвета близлежащих узлов. Термин *дядя* будем использовать для обозначения брата родительского узла, как и в фамильном дереве. Заметим, что:

* Свойство 3 (Все листья чёрные) выполняется всегда.
* Свойство 4 (Оба потомка любого красного узла — чёрные) может нарушиться только при добавлении красного узла, при перекрашивании чёрного узла в красный или при повороте.
* Свойство 5 (Все пути от любого узла до листовых узлов содержат одинаковое число чёрных узлов) может нарушиться только при добавлении чёрного узла, перекрашивании красного узла в чёрный (или наоборот), или при повороте.

Удаление:

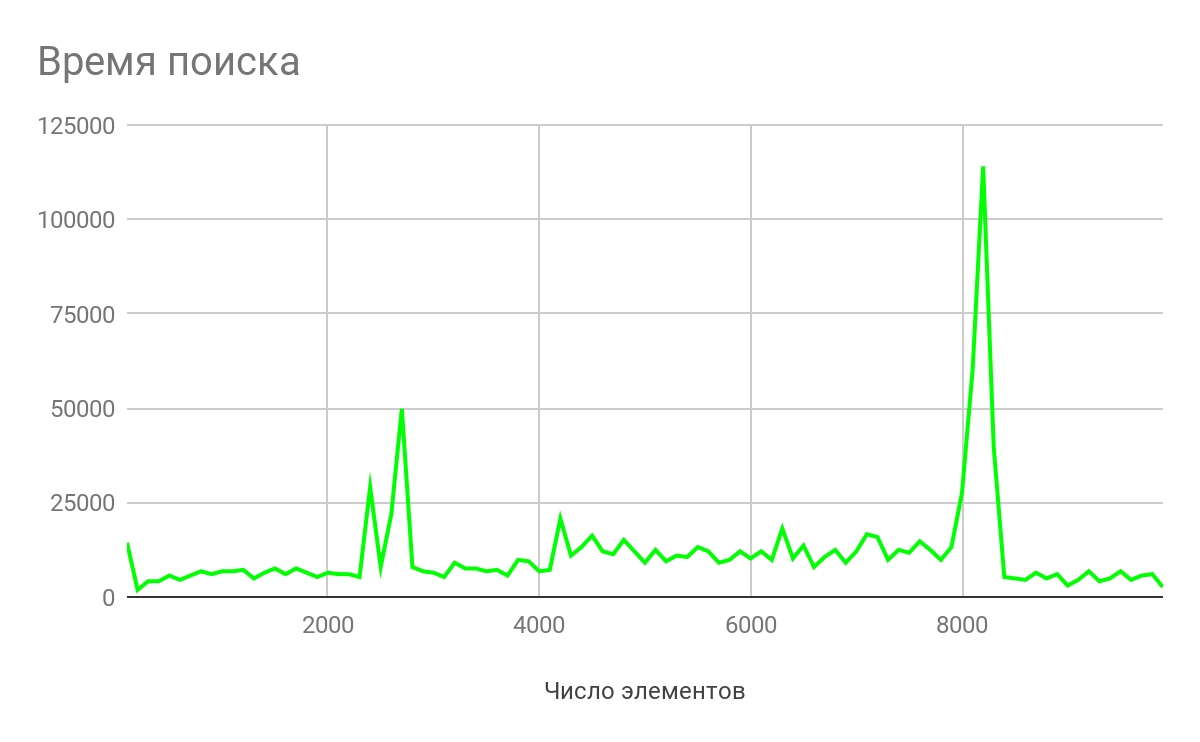
При удалении узла с двумя нелистовыми потомками в обычном двоичном дереве поиска мы ищем либо наибольший элемент в его левом поддереве, либо наименьший элемент в его правом поддереве и перемещаем его значение в удаляемый узел. Затем мы удаляем узел, из которого копировали значение. Копирование значения из одного узла в другой не нарушает свойств красно-чёрного дерева, так как структура дерева и цвета узлов не изменяются. Стоит заметить, что новый удаляемый узел не может иметь сразу два дочерних нелистовых узла, так как в противном случае он не будет являться наибольшим/наименьшим элементом. Таким образом, получается, что случай удаления узла, имеющего два нелистовых потомка, сводится к случаю удаления узла, содержащего как максимум один дочерний листовой узел. Поэтому дальнейшее описание будет исходить из предположения существования у удаляемого узла не более одного нелистового потомка.

## Оценка сложности:

Операции чтения для красно-чёрного дерева ничем не отличаются от оных для бинарного дерева поиска, потому что любое красно-чёрное дерево является особым случаем обычного бинарного дерева поиска. Однако непосредственный результат вставки или удаления может привести к нарушению свойств красно-чёрных деревьев. Восстановление свойств требует небольшого (O(log *n*) или O(1)) числа операций смены цветов (которая на практике очень быстрая) и не более чем трех поворотов дерева (для вставки — не более двух). Хотя вставка и удаление сложны, их трудоемкость остается O(log *n*).

## Графики







## 

## Выводы, применимость:

1. Самое главное преимущество красно-черных деревьев в том, что при вставке выполняется не более O(1) вращений. Это важно, например, в алгоритме построения динамической выпуклой оболочки. Ещё важно, что примерно половина вставок и удалений произойдут задаром.
2. Процедуру балансировки практически всегда можно выполнять параллельно с процедурами поиска, так как алгоритм поиска не зависит от атрибута цвета узлов.
3. Сбалансированность этих деревьев хуже, чем у АВЛ, но работа по поддержанию сбалансированности в красно-чёрных деревьях обычно эффективнее. Для балансировки красно-чёрного дерева производится минимальная работа по сравнению с АВЛ-деревьями.
4. Использует всего 1 бит дополнительной памяти для хранения цвета вершины. Но на самом деле в современных вычислительных системах память выделяется кратно байтам, поэтому это не является преимуществом относительно, например, АВЛ-дерева, которое хранит 2 бита. Однако есть реализации красно-чёрного дерева, которые хранят значение цвета в бите. Пример — Boost Multiindex. В этой реализации уменьшается потребление памяти красно-чёрным деревом, так как бит цвета хранится не в отдельной переменной, а в одном из указателей узла дерева.

Красно-чёрные деревья являются наиболее активно используемыми на практике самобалансирующимися деревьями поиска. В частности, ассоциативные контейнеры библиотеки STL(map, set, multiset, multimap) основаны на красно-чёрных деревьях. TreeMap в Java тоже реализован на основе красно-чёрных деревьев.

## Код:

https://github.com/TimFich/TAMirgaliev\_11-005\_AiSD/tree/master/Semester%20Work