КРАСНО-ЧЁРНОЕ ДЕРЕВО: БАЛАНСИРОВАНИЕ И СЛОЖНОСТЬ

Бадасян Т.С.[1], Авагян С.К.[2] Email: Badasyan1167@scientifictext.ru

[1]Бадасян Тигран Смбатович – магистрант,
факультет прикладной математики и физики;
[2]Авагян Сурен Константинович – магистрант,
кафедра электронных измерительных приборов и метрологии,
Национальный политехнический университет Армении,
г. Ереван, Республика Армения

Аннотация: *красно-чёрное дерево – вариант самобалансирующегося двоичного дерева поиска, которым гарантируется логарифмическое увеличение высоты и скорость выполнения основных операций, представленных добавлением, удалением и поиском узла. Сбалансированность определяется введением «чёрного цвета» или «красного цвета». Red-black tree применяется с целью организации сравнимых данных в виде текстовых фрагментов или числа. В листовых узлах не содержатся данные, поэтому отсутствует необходимость выделять память, а работа предполагает ведение записи в узле-предке (указатель на потомка). В статье рассмотрено красно-чёрное дерево: его балансирование и сложность.*

Ключевые слова: *красно-черное дерево, балансирование, алгоритм, эффективность, элемент.*

RED-BLACK TREE: BALANCING AND COMPLEXITY

Badasyan T.S.[1], Avagyan S.K.[2]

[1]Badasyan Tigran Smbatovich - Undergraduate,
FACULTY OF APPLIED MATHEMATICS AND PHYSICS;
[2]Avagyan Suren Konstantinovich - Undergraduate,
DEPARTMENT OF ELECTRONIC MEASURING INSTRUMENTS AND METROLOGY,
NATIONAL POLYTECHNIC UNIVERSITY OF ARMENIA,
YEREVAN, REPUBLIC OF ARMENIA

Abstract: *a red-black tree is a variant of a self-balancing binary search tree, which guarantees a logarithmic increase in height and speed of performing basic operations, represented by adding, deleting and searching a node. Balance is determined by the introduction of "black" or "red". Red-black tree is used to organize comparable data in the form of text fragments or numbers. The leaf nodes do not contain data, so there is no need to allocate memory, and the work involves maintaining records in the ancestor node (a pointer to a descendant). The article considers the red-black tree: its balancing and complexity.*

Keywords: *red-black tree, balancing, algorithm, efficiency, element.*

УДК 004.43:517.91

Красно-Чёрное Дерево: Балансирование и Сложность: Некоторые реализации предполагает упрощение алгоритма посредством явных листовых узлов. Следует отметить, что узлы могут быть красными или чёрными, с наличием пары потомков, а корень, чаще всего бывает чёрным, что не оказывает непосредственного воздействия на показатели работоспособности модели. Чёрные листья не содержат данные, а потомки красных узлов бывают чёрными. Важно помнить, что все простые пути от узлов-предков в сторону листовых узлов-потомков отличаются содержанием одинакового числа чёрных узлов. Данные ограничения характеризуют длину пути от корня до наиболее удалённого листа.

Итак, красно-черные деревья представляют собой сбалансированные бинарные деревья поиска, обладающие особыми свойствами. При этом в действительности красно-черными деревьями не гарантирована строгая сбалансированность, свойственная для АВЛ-деревьев, но в условиях соблюдения свойств обеспечивается осуществление операций: вставка, удаление и выборка за определённое время. Безусловно, двоичное дерево поиска представляет собой структуру данных, предназначенных для хранения требуемых элементов с наличием возможности максимально быстрого поиска. Такую идею характеризует простота и гениальность: «меньшее – влево, большее – вправо». Однако, видимая простота двоичного дерева поиска вполне естественно дополняется достаточно сложными вопросами, касающимися такого понятия, как балансировка дерева. Такое мероприятие направлено на предотвращение превращения Red-black tree в крайне неудобную и слишком длинную ветку. С точки зрения отсутствия полноценного сбалансирования, на некоторых участках высота красно-чёрного дерева вполне может иметь различия практически в два раза. Наличие данного допущения не обладает ассимптотическим влиянием на показатели скорости поиска элементов, однако, способно в значительной степени ускорять все процессы размещения новых элементов.

Эта особенность обусловлена отсутствием необходимости постоянно в условиях добавления элементов «существенно трясти» дерево, поэтому допускается простое добавление красного элемента, не затрагивая остальные и не производя замены «чёрной высоты». Вставка нового элемента требует присваивания ему красного цвета, а при необходимости новые образованные вершины просто перекрашиваются в необходимый цвет. Для добавления нового элемента нужно найти подходящий лист для осуществления вставки, а затем выкрасить красным цветом такой вновь создаваемый элемент, после чего выполнить перекрашивание узлов и произвести все повороты. К наиболее распространённым нарушениям свойств любого красно-чёрного дерева относится вставка нового красного узла при наличии чёрного корня, а также при наличии у красных узлов двух чёрных дочерних узлов. К восстановлению свойств необходимо приступать с нового элемента в условиях постепенного продвижения к корню Red-black tree. С целью удаления в соответствии с заданным ключом находится и удаляется элемент с обязательным последующим перекрашиванием узлов и выполнением поворотов, что позволяет восстановить структуру.

Существует несколько вариантов балансирования красно-чёрных деревьев, но с обязательным учётом строго определённых требований. Следует также помнить, что чёрная высота такого дерева представлена общим количеством узлов чёрного цвета на ветках от листа до корня. Для длиннейшего возможного расстояния характерно окрашивание узлов чередованием красного и чёрного. С этой точки зрения длиннейший конструируемый путь представлен чередованием с удвоенной длиной пути через узлы исключительно чёрного цвета. Важно помнить, что операции, осуществляемые над Red-black tree, должны осуществлять работу с самыми разными свойствами. Например, вставка и удаление в обязательном порядке сопровождаются полным сохранением абсолютно всех ключевых свойств. Как показывает практика, любое бинарное дерево работает лучше всего при сбалансированности, характеризуемой общей длиной пути от корня до одного из листьев, находящейся в строго определённых пределах, обусловленных числом вершин [3].

Наиболее распространённые варианты представлены красным предком и красным «дядей», при которых в результате стандартного перекрашивания легко и быстро устраняется красно-красное нарушение. В этом случае после перекрашивания необходимо выполнить проверку «дедушки» нового узла, который вполне может оказаться красного цвета. Также внимания требует понятие, представленное распространением воздействия узла красного цвета на верхние узлы в условиях Red-black tree. В этом случае корень окрашивается в чёрный цвет, благодаря чему при красном корне получается увеличить чёрную высоту дерева. При наличии ситуации с красным предком и чёрном «дяде» нарушение устраняется посредством вращения узлов, что позволяет осуществить простую, но эффективную корректировку в отношении поддеревьев. Как показывает практика, в этом месте может наблюдаться остановка алгоритма, спровоцированная полным отсутствием красно-красных конфликтов. Вершину дерева потребуется окрасить в подобной ситуации в чёрный цвет. Особенно важно уделить внимание ситуации, при которой такой узел изначально представлял собой правого потомка. В этом случае сначала используется правило применения левого вращения, которое позволяет сделать данный узел стандартным левым потомком.

Некоторое время красно-чёрное дерево представлялось своеобразной абстракцией, которая в определённый момент превратилась в абсолютно самостоятельную структуру данных, вполне естественно обладающих достаточно простыми и очень понятными правилами, направленными на сохранение требуемого уровня сбалансированности. В первую очередь нужно учитывать тот факт, что красные узлы не могут дополняться дочерними узлами такого же цвета, что обусловлено отсутствием удовлетворения правил абстракции. С этой точки зрения красные узлы представляют собой часть логического узла, соответственно допустимо окрашивание последующего узла только в чёрный цвет. В этом плане действует правило, согласно которым связи красных узлов – это часть связей узлов B-дерева с последующими узлами B-дерева, а пренебрежение данным требованиям становится причиной образования так называемых красных нарушений. Во-вторых, во всех рассматриваемых B-деревьях путь, занимаемый расстояние от вершины до листа представлен одним и тем же количеством узлов, с учётом расположения всех листьев на одинаковом уровне. При этом количество узлов чёрного цвета в условиях абсолютно любого пути Red-black tree одинаково.

Таким образом, отношение материнского чёрного узла с его красными дочерними узлами должно рассматриваться в качестве части логического узла, поэтому учёт красных узлов в таком пути не осуществляется. Подобная ситуация носит название чёрной высоты, а несоблюдение правила подпадает под понятие чёрного нарушения. Кроме прочего, вершина определяется именно чёрным узлом, что имеет смысловую нагрузку, потому что только для чёрного узла доступно наличие детей красного цвета. Нужно помнить, что особенностью вершины является полное отсутствие такого понятия, как родительский узел. Распространение этого правила на абстракцию самобалансирующегося B-дерева является широко известной практикой. В конечном итоге вершину самобалансирующегося дерева вполне допускается окрашивать в чёрный или красный цвет, потому что в подобной ситуации полностью

отсутствует воздействие на любые значимые показатели эффективности самобалансирующегося Red-black tree [1].

К алгоритмам, обеспечивающим сбалансированность красно-чёрного дерева, предъявляется целый ряд строго определённых требований, включая полное отсутствие нарушений какого-либо правила. В условиях соблюдения таких правил, самобалансирующееся Red-black tree отличается высотой, минимальная длина которого lg(n+1), в условиях максимальной высоты в пределах 2lg (n+1). Рассматриваемые величины согласно правилам являются строго логарифмическими, поэтому гарантируется аппроксимация самого лучшего случая бинарного дерева поиска. Следует отметить, что получение красно-чёрными деревьями широкого распространения и оригинальной абстракции способствовало заменена структурой данных. Таким образом устраняется вероятность нарушений пары ключевых правил, к числу которых относится отсутствие потомков красного цвета у красного узла, а также наличие одинакового количества узлов чёрного цвета в каждом пути. Строгое соблюдение перечисленных правил позволяет выполнять наиболее важные требования, которые предъявляются к самобалансирующимся Red-black tree с точки зрения оптимизации показателей производительности. Нужно отметить тот факт, что вершину не обязательно нужно окрашивать в чёрный цвет, но такой подход чаще всего способствует существенному упрощению всех применяемых алгоритмов.

Балансирование заключается, как правило, в простейших, но достаточно эффективных действиях, согласно которым на начальном этапе следует подготовить основу и так называемый скелет для кода. Ведение цветового окрашивание определяется пользователем, а категория общего решения представлено полем перечисляемого типа, принимающим значение существующей константы (чёрный и красный цвет). При использовании флага предпочтительнее применять узел красного цвета, а при его отсутствии – чёрного окрашивания. Безусловно, вполне допустимо применять инвертирование значения, а балансировка включает в себя замену применения традиционного правого и левого указателя массивом, в котором для элемента (индекс 0) характерно наличие левого указателя, а для элемента (индекс 1) – правого указателя. Таким образом, существенно облегчается решение задачи, направленной на соединение всех случаев симметричного типа в условиях заметного сокращения кода с гарантией необходимого уровня корректности. С целью предупреждения захламления кода в результате осуществления дополнительных проверок, вполне можно применять небольшие вспомогательные функции, определяющие красный цвет узла[2].

Правила размещения элементов в красно-чёрном дереве

Каждый узел либо красный, либо чёрный, NIL-листья всегда чёрные.

Корень дерева всегда чёрный

Оба потомка каждого красного узла — чёрные

Путь вниз от любого узла до любого листа-потомка содержит одинаковое число чёрных узлов.

При вставке нового элемента ему присваивается красный цвет. Для выполнения первых двух правил достаточно просто перекрашивать новые вершины в нужный цвет.

Рассмотрим правила балансировки для выполнения 3 и 4 пункта.

На каждом рисунке предполагается, что мы уже добавили элемент X, который нарушает 3 правило, нужно так изменить структуру дерева, чтобы 3 правило выполнялось, а 4 не нарушилось.

Условные обозначения вершин:

- X — добавленный элемент, который нарушает 3 пункт правил.
- P — папа элемента X
- G — дедушка элемента X, папа элемента P
- U — дядя элемента X, брат элемента P, второй сын элемента G.

Случай первый — красный дядя

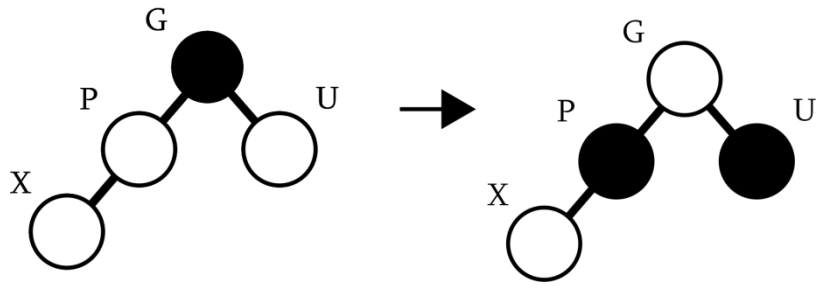

Рис. 1. Случай 1: Балансировка для G

Если и отец, и дядя красного цвета, то мы можем «спустить» чёрный цвет с уровня деда на уровень отца и перекрасить узлы, как показано на рисунке. В этом случае «чёрная высота» останется прежней, однако возможно нарушение 3 правила для элемента G, поэтому необходимо рекурсивно вызвать дальнейшую балансировку для этого узла.

Случай второй — чёрный дядя — папа и дед в разных сторонах.

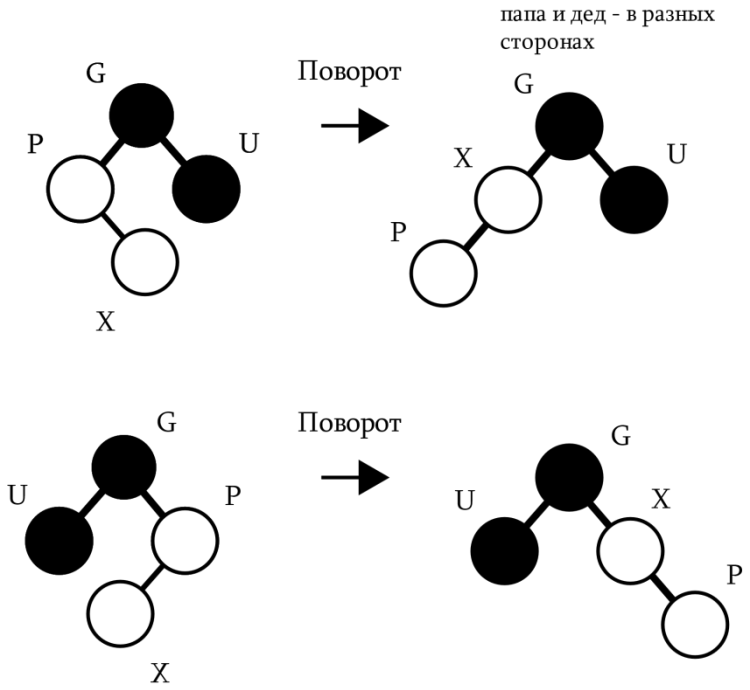

Рис. 2. Случай 2: Дядя черный, папа и дед в разных углах. Переход к Случаю 3 для P

Эту структуру необходимо привести к третьему случаю, когда папа и дед идут в одну сторону. Для этого нужно выполнить малый поворот от сына X к его отцу (правила поворотов в этой статье не рассматриваются) и вызвать 3 случай для элемента P.

Случай третий — чёрный дядя — папа и дед в одной стороне

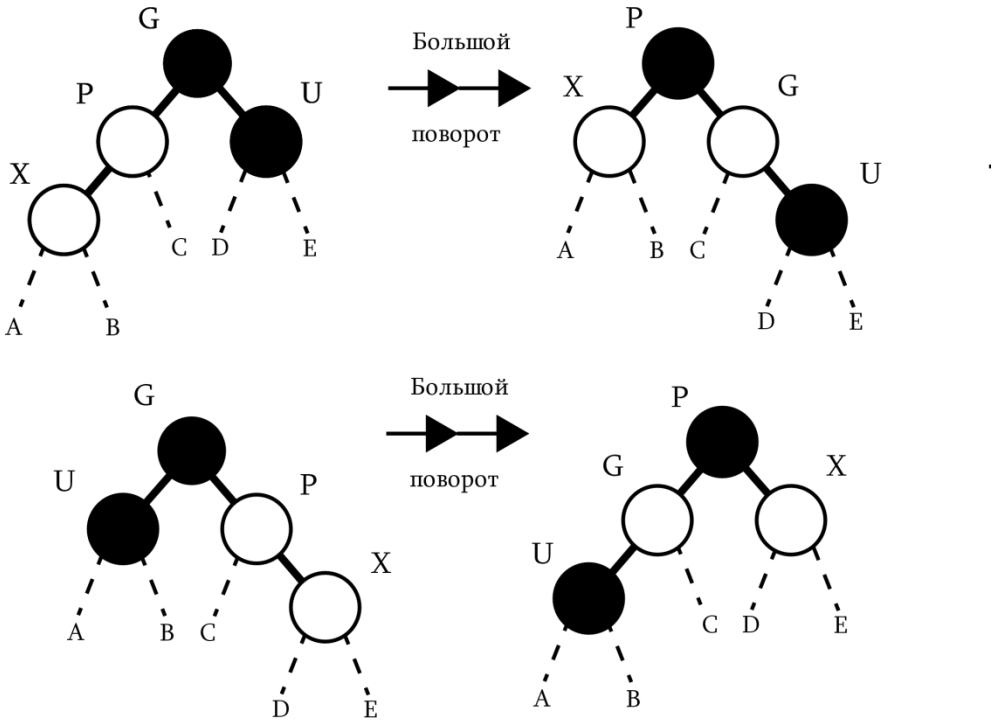

Рис. 3. Случай 3: чёрный дядя, папа и дед в одной стороне. Папа идет в одной стороне

В этом случае мы уже можем совершить большой поворот от отца через деда к чёрному дяде и перекрасить P в чёрный, а G в красный. В результате этого поворота 3-правило будет выполнено.

Убедимся, что 4 правило тоже выполняется. Предположим, что до большого поворота чёрная высота элемента G была N+2. Тогда высота «подвешенных» элементов будет следующей:

A = N+1,

B = N+1,
C = N+1,
D = N,
E = N.

На практике любую схему балансировки деревьев характеризует применение правила, которое заключается во вращении с целью изменения структурных характеристик при сохранении всех действующих правил. Очень удачным вариантом является использование такой технологии, как перекрашивание вращаемых узлов. В условиях одиночного поворота осуществляется стандартный поворот узла с «окрашиванием» старой вершины красным цветом, а новой вершины – чёрным. Выполнение двойного поворота предполагает совершение пары одиночных поворотов. Кроме прочего, перекрашивание является полезным не только в условиях вращения, но также и при применении алгоритма операции по удалению. Исходя из перечисленных выше сведений, вполне можно сделать вывод, что любые сбалансированные деревья представляют собой абсолютно нетривиальную структуру данных, но именно самобалансирующиеся Red-black tree обладают особенностями с точки зрения правильности восприятия. Именно по этой причине абсолютно любые, даже самые незначительные преобразования к какой-либо части дерева способны стать причиной нарушений на следующих участках.

Список литературы / References

1. *Ахо Альфред В., Хопкрофт Джон Э., Ульман Джеффри Д.* Структуры данных и алгоритмы. М.: Вильямс, 2000. С. 384.
2. *Мейн Майкл, Савитч Уолтер.* Структуры данных и другие объекты в C++. 2-е изд. М.: Вильямс, 2002. С. 832.
3. *Седжвик Роберт.* Фундаментальные алгоритмы на С. Анализ / Структуры данных / Сортировка / Поиск. СПб.: ДиаСофтЮП, 2003. С. 672.