5. Красно-черное дерево

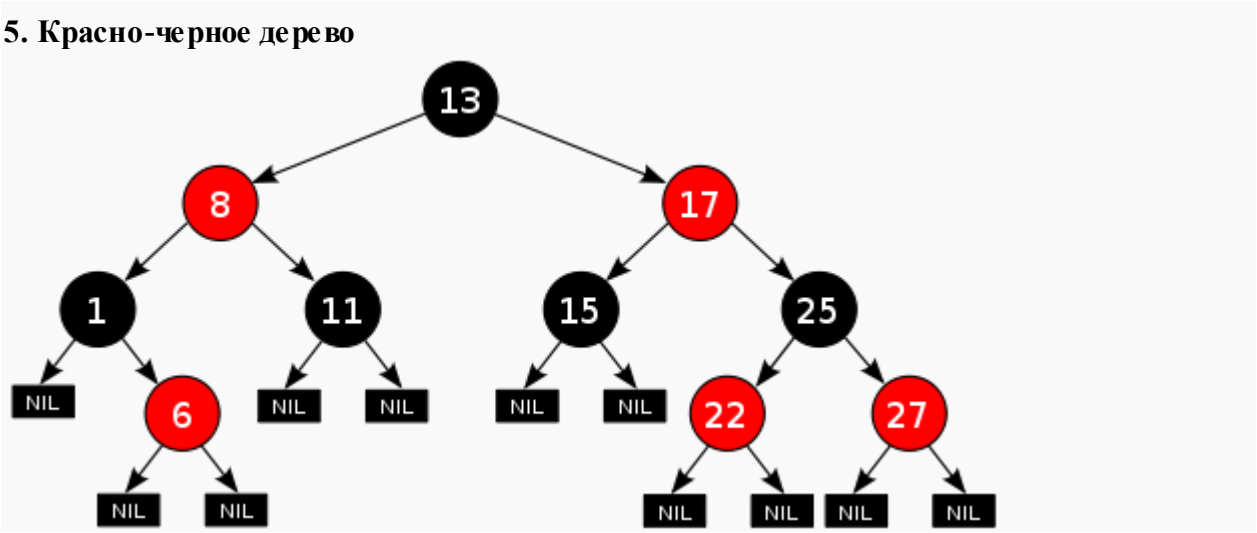

Рисунок 5.1.Пример красно-чёрного дерева

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

1. Узел либо красный, либо чёрный.
2. Корень — чёрный.
3. Все листья — черные.
4. Оба потомка каждого красного узла — черные.
5. Всякий простой путь от данного узла до любого листового узла, являющегося его потомком, содержит одинаковое число черных узлов.

Эти ограничения реализуют критическое свойство красно-черных деревьев: путь от корня до самого дальнего листа не более чем в два раза длиннее пути от корня до ближайшего листа (если дальний лист расположен на 3-м уровне). Результатом является то, что дерево примерно сбалансировано. Так как такие операции как вставка, удаление и поиск значений требуют в худшем случае времени, пропорционального длине дерева, эта теоретическая верхняя граница высоты позволяет красно-чёрным деревьям быть более эффективными в худшем случае, чем обычные двоичные деревья поиска.

Чтобы понять, почему это гарантируется, достаточно рассмотреть эффект свойств 4 и 5 вместе. Пусть для красно-чёрного дерева T число черных узлов в свойстве 5 равно B. Тогда кратчайший возможный путь от корня дерева T до любого листового узла содержит B черных узлов. Более длинный возможный путь может быть построен путем включения красных узлов. Однако, свойство 4 не позволяет вставить несколько красных узлов подряд. Поэтому самый длинный возможный путь состоит из 2B узлов, попеременно красных и черных. Любой максимальный путь имеет одинаковое число черных узлов (по свойству 5), следовательно, не существует пути, более чем вдвое длинного, чем любой другой путь.

Во многих реализациях структуры дерева возможно, чтобы узел имел только одного потомка и листовой узел содержал данные. В этих предположениях реализовать красно-чёрное дерево возможно, но изменятся несколько свойств и алгоритм усложнится. По этой причине используется «фиктивные листовые узлы», которые не содержат данных и просто служат для указания, где дерево заканчивается. Следствием этого является то, что все внутренние (не являющиеся листовыми) узлы имеют два потомка, хотя один из них может быть нулевым листом. Свойство 5 гарантирует, что красный узел обязан иметь в качестве потомков либо два черных нулевых листа, либо два черных внутренних узла. Для чёрного узла с одним потомком нулевым листовым узлом и другим потомком, не являющимся таковым, свойства 3, 4 и 5 гарантируют, что последний должен быть красным узлом с двумя черными нулевыми листьями в качестве потомков.

Операции с красно-черными деревьями

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

Вставка

Вставка начинается с добавления узла, как 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 черными потомками на место чёрного листа.

Дальнейшие действия зависят от цвета близлежащих узлов. Будем использовать термин *дядя* для обозначения брата родительского узла, как и в фамильном дереве. Заметим, что:

- Свойство 3 (Все листья черные) выполняется всегда.
- Свойство 4 (Оба потомка любого красного узла — черные) может нарушиться только при добавлении красного узла, при перекрашивании чёрного узла в красный или при повороте.
- Свойство 5 (Все пути от любого узла до листа содержат одинаковое число черных узлов) может нарушиться только при добавлении чёрного узла, перекрашивании красного узла в чёрный (или наоборот), или при повороте.
- Примечание: Обозначим буквой N текущий узел (окрашенный красным). Сначала это новый узел, который вставляется, но эта процедура может быть рекурсивно применена

к другим узлам (смотрите случай 3). P - обозначение предка N, через G обозначается дедушку N, а U - дядя N.

Случай 1: Текущий узел **N** в корне дерева. В этом случае, он перекрашивается в чёрный цвет, чтобы оставить верным Свойство 2 (Корень — чёрный). Так как это действие добавляет один чёрный узел в каждый путь, Свойство 5 (Все пути от любого данного узла до листовых узлов содержат одинаковое число черных узлов) не нарушается.

Случай 2: Предок **P** текущего узла чёрный, то есть Свойство 4 (Оба потомка каждого красного узла — черные) не нарушается. В этом случае дерево правильное. Свойство 5 (Все пути от любого данного узла до листовых узлов содержат одинаковое число черных узлов) не нарушается, потому что текущий узел **N** имеет двух черных листовых потомков, но так как **N** является красным, пути до каждого из этих потомков содержит такое же число черных узлов, что и путь до чёрного листа, который был заменен текущим узлом, который был чёрный, так что свойство остается верным.

Примечание: В следующих случаях предполагается, что у **N** есть дедушка **G**, так как его родитель **P** является красным, а если бы он был корнем, то был бы окрашен в черный цвет. Таким образом, **N** также имеет дядю **U**, хотя он может быть листовым узлом в случаях 4 и 5.

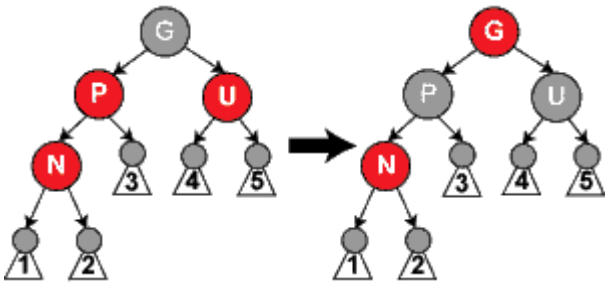

Рис. 5.2. Схема случая 3

Случай 3: Если и родитель P и дядя U — красные, то они оба могут быть перекрашены в чёрный и дедушка G станет красным (для сохранения свойства 5 (Все пути от любого данного узла до листовых узлов содержат одинаковое число черных узлов)). Теперь у текущего красного узла N чёрный родитель. Так как любой путь через родителя или дядю должен проходить через дедушку, число черных узлов в этих путях не изменится. Однако, дедушка G теперь может нарушить свойства 2 (Корень — чёрный) или 4 (Оба потомка каждого красного узла — черные) (свойство 4 может быть нарушено, так как родитель G может быть красным). Чтобы это исправить, вся процедура рекурсивно выполняется на G из случая 1.

Примечание: В оставшихся случаях предполагается, что родитель P является левым потомком своего предка. Если это не так, необходимо поменять лево и право.

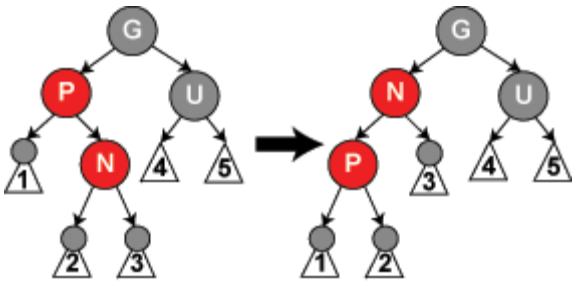

Рис.5.3. Схема случая 4.

Случай 4: Родитель P является красным, но дядя U — чёрный. Текущий узел N — правый потомок P, а P — левый потомок своего предка G. В этом случае может быть произведен поворот, который меняет роли текущего узла N и его предка P. Тогда, бывший родительский узел P рассматривается, используя случай 5 (перенумеровывающий N и P), потому что Свойство 4 (Оба потомка любого красного узла — черные) все ещё нарушено. Вращение приводит к тому, что некоторые пути (в поддереве 1 на схеме) проходят через узел N, чего до поворота не было. Это приводит к тому, что некоторые пути не проходят через узел P. Однако, оба из этих узлов являются красными, так что Свойство 5 (Все пути от любого данного узла до листовых узлов содержат одинаковое число черных узлов) не нарушается при вращении.

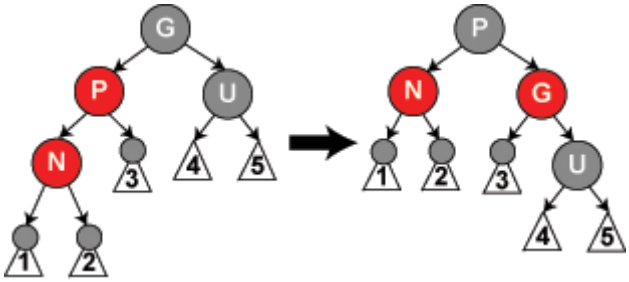

Рис. 5.4. Схема случая 5

Случай 5: Родитель **P** красный, но дядя **U** — чёрный, текущий узел **N** — левый потомок **P** и **P** — левый потомок **G**. В этом случае выполняется поворот дерева на **G**. В результате получается дерево, в котором бывший родитель **P** теперь является родителем и текущего узла **N** и бывшего дедушки **G**. Известно, что **G** — чёрный, так как его бывший потомок **P** не мог бы в противном случае быть красным (без нарушения Свойства 4). Тогда цвета **P** и **G** меняются и в результате дерево удовлетворяет Свойству 4 (Оба потомка любого красного узла — черные). Свойство 5 (Все пути от любого данного узла до листовых узлов содержат одинаковое число черных узлов) также остается верным, так как все пути, которые проходят через любой из этих трех узлов, ранее проходили через G, поэтому теперь они все проходят через P. В каждом случае, из этих трех узлов только один окрашен в чёрный.

Удаление

При удалении узла с двумя не листовыми потомками в обычном двоичном дереве поиска ищется либо наибольший элемент в его левом поддереве, либо наименьший

элемент в его правом поддереве и перемещается его значение в удаляемый узел. Затем, удаляется узел, из которого копировали значение.

Обозначим через M удаляемый узел; через C обозначим потомка M, который будем называть просто «его потомок». Если M имеет не листового потомка, возьмем его за C. В противном случае за C возьмем любой из листовых потомков.

Если M является красным узлом, заменим его своим потомком C, который по определению должен быть чёрным. (Это может произойти только тогда, когда M имеет двух листовых потомков, потому что если красный узел M имеет чёрного не листового потомка с одной стороны, а с другой стороны — листового, то число черных узлов на обеих сторонах будет различным, таким образом, дерево перестанет быть красно-чёрным деревом из-за нарушения Свойства 5.) Все пути через удаляемый узел просто будут содержать на один красный узел меньше, предок и потомок удаляемого узла должны быть черными, так что Свойство 3 («Все листья — черные») и Свойство 4 («Оба потомка красного узла — черные») все ещё сохраняется.

Другой простой случай: M — чёрный и C — красный. Простое удаление чёрного узла нарушит Свойство 4 («Оба потомка красного узла — черные») и Свойство 5 («Всякий простой путь от данного узла до любого листового узла, содержит одинаковое число черных узлов»), но если мы перекрасим C в чёрный, оба эти свойства сохранятся.

Сложный случай: M и C — черные. (Это возможно только тогда, когда удаляется чёрный узел, который имеет два листовых потомка, потому что если чёрный узел M имеет чёрного не листового потомка с одной стороны, а с другой — листового, то число черных узлов на обеих сторонах будет различным и дерево станет неправильным красно-чёрным деревом из-за нарушения Свойства 5.) Начнем с замены узла M своим потомком C. Назовем этот потомок (в его новом положении) N, а его «брата» (другого потомка его нового предка) — S. (До этого S был «братом» M.) На рисунках, расположенных далее, будем использовать обозначение P для нового предка N (старого предка M), SL для левого потомка S и SR для правого потомка S(S не может быть листовым узлом, так как если N по нашему предположению является чёрным, то поддерево P, которое содержит N, чёрной высоты два и поэтому другое поддерево P, которое содержит S должно быть также чёрной высоты два, что не может быть в случае, когда S — лист).

Примечание: В некоторых случаях мы меняем роли и обозначения узлов, но в каждом случае любое обозначение продолжает означать тот же узел, что и в начале случая. Любые цвета, изображенные на рисунке либо предполагаются случаем, либо получается из других предположений. Белый означает неизвестный цвет (либо красный, либо черный).

Если оба N и его текущий отец черные, тогда удаление отца приведет к тому, что пути, которые проходят через N, будут иметь на один чёрный узел меньше, чем

пути, которые не проходят через него. Так как это нарушает свойство 5 (все пути из любого узла к его листовым узлам содержат одинаковое количество черных узлов), дерево должно быть сбалансировано. Рассмотрим несколько случаев:

Случай 1: N — новый корень. В этом случае, все сделано. Удален один чёрный узел из каждого пути и новый корень является чёрным узлом, так что свойства сохранены.

Примечание: В случаях 2, 5 и 6 предполагаем, что **N** является левым потомком своего предка **P**. Если он — правый потомок, *left* и *right* нужно поменять местами во всех трех случаях. При написании кода необходимо принять это во внимание.

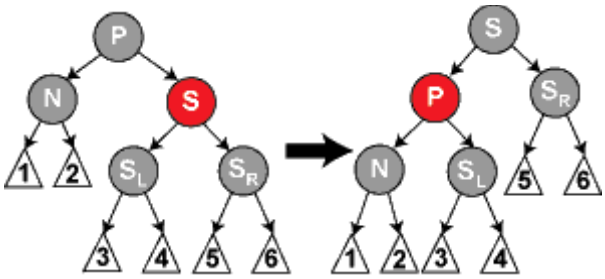

Рис. 5.5. Схема случая 2

Случай 2: S — красный. В этом случае мы меняем цвета **P** и **S**, и затем выполняется вращение влево вокруг **P**, при этом **S** становится дедушкой **N**. Заметим, что **P** должен быть чёрным, если он имеет красного потомка. Хотя все пути по прежнему содержат одинаковое количество черных узлов, **N** имеет чёрного брата и красного отца. Таким образом, можно перейти к шагу 4, 5 или 6. (Его новый брат является чёрным потому, что он был потомком красного **S**.) Далее через **S** будет обозначен новый брат **N**.

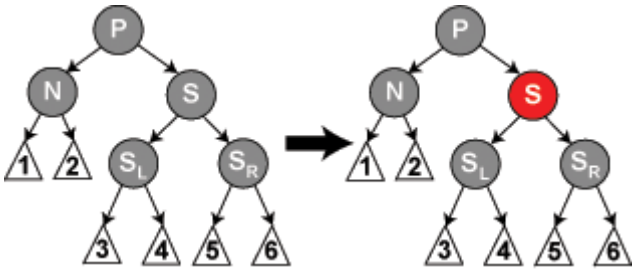

Рис. 5.6. Схема случая 3

Случай 3: P, **S**, и дети **S'** - черные. В этом случае мы просто перекрашиваем **S** в красный. В результате все пути, проходящие через **S**, но не проходящие через **N**, имеют на один чёрный узел меньше. Так как удаления отца **N** приводит к тому, что все пути, проходящие через **N**, содержат на один чёрный узел меньше, то такие действия выравнивают баланс. Тем не менее, все проходящие через **P** пути теперь содержат на один чёрный узел меньше, чем пути, которые через **P** не проходят, поэтому свойство 5 (все пути из любой вершины к её листовым узлам содержат одинаковое количество черных узлов) все ещё нарушено. Чтобы это исправить, мы применяем процедуру балансировки к **P**, начиная со случая 1.

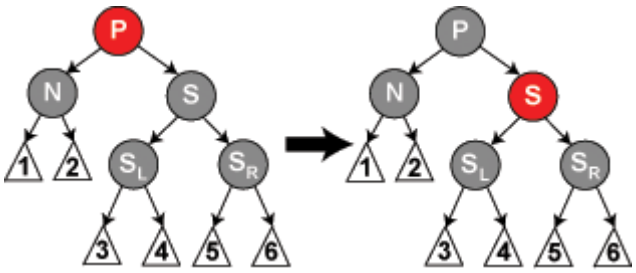

Рис. 5.7. Схема случая 4

Случай 4: S и его дети - черные, но **P** - красный. В этом случае просто **S** и **P меняются цветами**. Это не влияет на количество черных узлов на путях, проходящих через **S**, но добавит один к числу черных узлов на путях, проходящих через **N**, восстанавливая тем самым влияние удаленного чёрного узла.

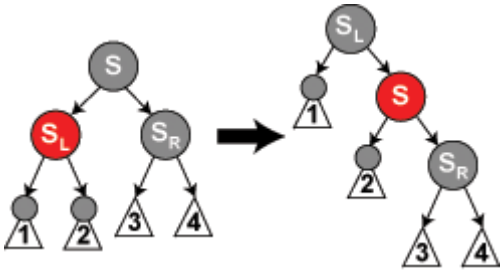

Рис. 5.8. Схема случая 5

Случай 5: S - чёрный, левый потомок **S** - красный, правый потомок **S** - чёрный, и **N** – левый потомк своего отца. В этом случае дерево вращается вправо вокруг **S**. Таким образом левый потомок **S** становится его отцом и новым братом **N**. После этого цвета у **S** и его нового отца меняются. Все пути по прежнему содержат одинаковое количество черных узлов, но у **N** есть чёрный брат с красным правым потомком, поэтому переходим к случаю 6. Ни **N**, ни его отец не влияют на эту трансформацию. (Для случая 6 обозначим через **S** нового брата **N**.)

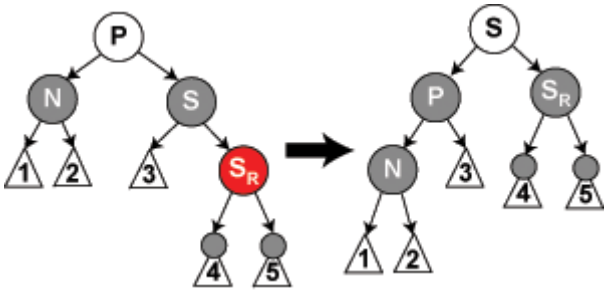

Рис. 5.9. Схема случая 6

Случай 6: S - чёрный, правый потомок **S** - красный, и **N** является левым потомком своего отца **P**. В этом случае дерево вращается влево вокруг **P**, после чего **S** становится отцом **P** и своего правого потомка. Далее изменяем цвета у **P** и **S**, и делаем правого потомка **S** чёрным. Поддерево по прежнему имеет тот же цвет корня, поэтому свойства 4 (Оба потомка каждого красного узла - черные) и 5 (все пути из любой вершины к её листовым узлам содержат одинаковое количество черных узлов) не нарушаются. Тем не менее, у **N** теперь появился дополнительный чёрный предок: либо **P** стал чёрным, или он был чёрным

и **S** был добавлен в качестве чёрного дедушки. Таким образом, проходящие через **N** пути проходят через один дополнительный чёрный узел.

Между тем, если путь не проходит через **N**, то есть 2 возможных варианта:

- Он проходит через нового брата **N**. Тогда, он должен проходить через **S** и **P**, которые просто поменяли цвета и места. Поэтому путь содержит то же количество черных узлов.
- Он проходит через нового дядю **N**, правого потомка **S**. Когда-то он проходил через **S**, отца **S** и правого потомка **S**(который был красным), но теперь он проходит только через **S**, который принял на себя цвет своего прежнего родителя, и правого потомка **S**, который был перекрашен из красного в чёрный (Предполагаем, что цвет **S**: чёрный). Эффект заключается в том, что этот путь проходит через такое же количество черных узлов.

  В любом случае, число черных узлов на этих путях не изменится. Поэтому, восстанавливается свойство 4 (Оба потомка каждого красного узла - черные) и 5 (все пути из любой вершины к её листовым узлам содержат одинаковое количество черных узлов). Белый узел на диаграмме может быть как красным так и чёрным, но должен указывать на тот же цвет как в начале, так и в конце трансформации

Дополнительные замечания

- Вставка требует до 2 поворотов в дереве.
- Удаление из красно-чёрного дерева требует до 3 поворотов.
- Красно-чёрное дерево в каждом узле хранит цвет (1 бит). Таким образом, красно-чёрное дерево экономично.
- Красно-чёрное дерево, которое содержит *n* внутренних узлов, имеет высоту $O(log(n))$.