Тема 4. Методы поиска

4.2. Логарифмический поиск в динамических таблицах

4.2.1. Красно-черные-деревья

Как и АВЛ-деревья, *красно-черные деревья* (*RB-деревья*) являются одним из типов сбалансированных деревьев бинарного поиска, используемых для организации динамических таблиц. В таких деревьях специальные операции балансировки гарантируют, что высота дерева не превзойдет $O(\log n)$, а следовательно, и табличные операции (поиск, включение и исключение) будут выполняться за время, не превышающее $O(\log n)$.

RB-дерево – это дерево бинарного поиска, вершины которого разделены на красные и черные. При этом должны выполняться определенные требования, которые гарантируют, что уровни двух любых листьев отличаются не более чем в два раза, поэтому дерево называется сбалансированным.

Каждая вершина RB-дерева может быть представлена узлом, состоящим из полей *color* (цвет), *info* (имя), *left* (указатель на левого сына) и *right* (указатель на правого сына). Для удобства считается, что RB-дерево является расширенным бинарным деревом, т. е. значения Λ, хранящиеся в полях *left* и *right*, являются ссылками на фиктивные внешние узлы (листья) дерева, которые будем называть Λ-сыновьями. В таком случае каждый узел, содержащий имя, имеет двух сыновей и является внутренним узлом. При разработке алгоритмов, чтобы упростить обработку граничных условий, можно использовать единственный фиктивный узел, соответствующий всем внешним узлам бинарного дерева (Λ-сыновьям). Этот фиктивный узел имеет ту же структуру, что и обычный узел дерева. Тогда достаточно все указатели Λ заменить указателями на фиктивный узел.

Дерево бинарного поиска называется красно-черным деревом (RB-деревом), если оно обладает следующими свойствами, называемыми *RB-свойствами*:

1) каждая вершина – либо красная, либо черная;

2) каждый лист (внешняя вершина) – черный;

3) если вершина красная, оба ее сына черные;

4) все пути, идущие от корня к листьям, содержат одинаковое количество черных вершин.

Будем предполагать, что в RB-деревьях корень черный, и поддерживать это свойство при преобразовании структуры дерева. Пример RB-дерева показан на рис. 4.11. Черные вершины изображены как темные, красные – как светлые.

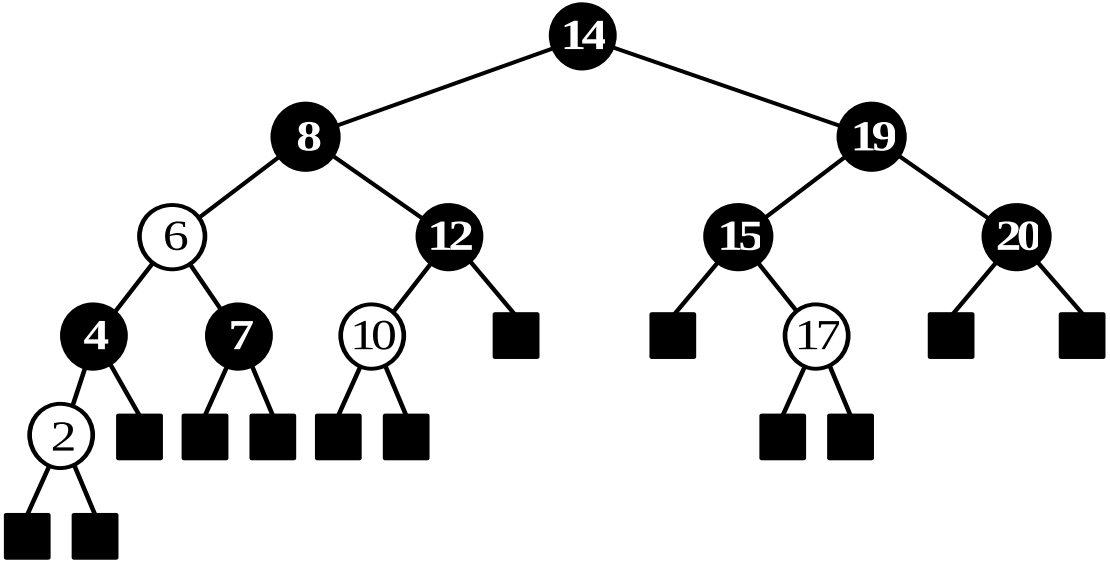

Рис. 4.11. Красно-черное дерево

Все пути, ведущие от произвольной вершины x к листьям, содержат одно и то же число черных вершин. Число черных вершин в любом из них (сама вершина x в это число не входит) называется *черной высотой* вершины x и обозначается $bh(x)$. Например, $bh(8) = 2$, $bh(6) = 2$, а $bh(14) = 3$. Черная высота дерева определяется черной высотой его корня. RB-дерево с n внутренними вершинами имеет высоту не более $2\log(n + 1)$. Поэтому поиск выполняется за время $O(\log n)$, так как время его выполнения есть $O(h)$ для дерева высоты h.

Операции включения и исключения изменяют дерево, поэтому результат может не обладать RB-свойствами. Для восстановления этих свойств требуется перекрасить некоторые вершины и изменить структуру дерева. Структура дерева меняется с помощью локальных операций левого и правого вращений, аналогичных вращениям для АВЛ-деревьев.

Включение новой вершины. Для включения новой вершины применяется обычный алгоритм включения имени *z* в дерево бинарного поиска. Новая вершина окрашивается красным цветом. После этого необходимо восстановить RB-свойства, которые могли быть нарушены включением. Для этого путь из корня в лист проходится в обратном направлении.

После включения новой красной вершины выполняются все RB-свойства, кроме одного – новая красная вершина может иметь красного отца. При этом следует помнить, что новая красная вершина имеет двух черных Λ-сыновей. Следует отметить, что черная высота дерева при включении не меняется, так как новая красная вершина с черными Λ-сыновьями замещает внешнюю черную вершину (лист). Если новая вершина включается в пустое дерево, то она перекрашивается в черный цвет, и процесс включения завершается. Таким образом, если новая красная вершина имеет черного отца, никакие RB-свойства не нарушаются и никакие действия по балансировке не производятся.

Если же отцом новой красной вершины является красная вершина, то действия, предпринимаемые при прохождении вершин по восходящему пути, зависят от направления движения (от левого или правого сына) и цвета брата отца (дяди) новой вершины. Пусть p указывает на исследуемую на текущем этапе вершину (на начальном этапе – это добавленная красная вершина), f – на вершину-отца, u – на вершину-дядю и g – на вершину-деда вершины p. Тогда если вершина p – корень дерева или имеет черного отца, то движение вверх прекращается и процесс завершается. Если же вершина p имеет красного отца f и не является корнем дерева, то возможны шесть случаев. Три из них симметричны трем другим, их различие в том, является ли отец f вершины p левым или правым сыном своего отца (вершины g). Рассмотрим один из симметричных вариантов, а именно: отец f вершины p является левым сыном своего отца g. Тогда возможны следующие случаи, проиллюстрированные на рис. 4.12, где все поддеревья α, β, γ, δ и ε имеют черный корень и одинаковую черную высоту.

Случай 1. Если вершина u красная (вершина f тоже красная), то вершины f и u перекрашиваются в черный цвет, а вершина g – в красный. При этом число черных вершин на любом пути от корня к листьям останется прежним. Действия не зависят от того, является ли вершина p правым (рис. 4.12, *а*) или левым (рис. 4.12, *б*) сыном вершины f. Нарушение RB-свойств возможно в единственном месте нового дерева: у красной вершины g может быть красный отец. Поэтому процесс движения вверх продолжается, присвоив указателю p значение указателя g.

Случай 2. Если вершина u черная и красная вершина p является правым сыном красной вершины f (рис. 4.12, *в*), то выполняется левое вращение вокруг вершины f и указателю p присваивается значение указателя f. В результате этих действий случай 2 сводится к случаю 3. Следует отметить, что после вращения количество черных вершин на путях от корня к листьям остается прежним.

Случай 3. Если вершина u черная и красная вершина p является левым сыном красной вершины f (рис. 4.12, *г*), которая, в свою очередь, является левым сыном черной вершины g (вершина g черная, поскольку ее сыновья f и u разных цветов, а это возможно только для черных вершин), то выполняется правое вращение вокруг вершины g. Затем для устранения нарушений RB-свойств перекрашиваются: вершина f в черный, а вершина g – в красный цвет. Поскольку теперь отец (вершина f) вершины p черный, движение вверх прекращается и процесс завершается.

Анализ рассмотренных случаев показывает, для чего корень дерева необходимо поддерживать черным: если отец новой вершины красный, он не может быть корнем, и вершина-дед всегда существует (а значит, всегда есть вершина-дядя). Для поддержания этого свойства на случай, если цвет корня поменялся в результате преобразований, процесс включения необходимо завершать операцией окраски корня дерева в черный цвет.

Поскольку высота RB-дерева есть $O(\log n)$, то операция включения требует времени $O(\log n)$. Причем движение вверх по дереву продолжается только для случая 1 (при этом перекрашиваются только некоторые вершины). В случаях 2 и 3 после выполнения вращений процесс движения прекращается, при этом выполняется не более двух вращений.

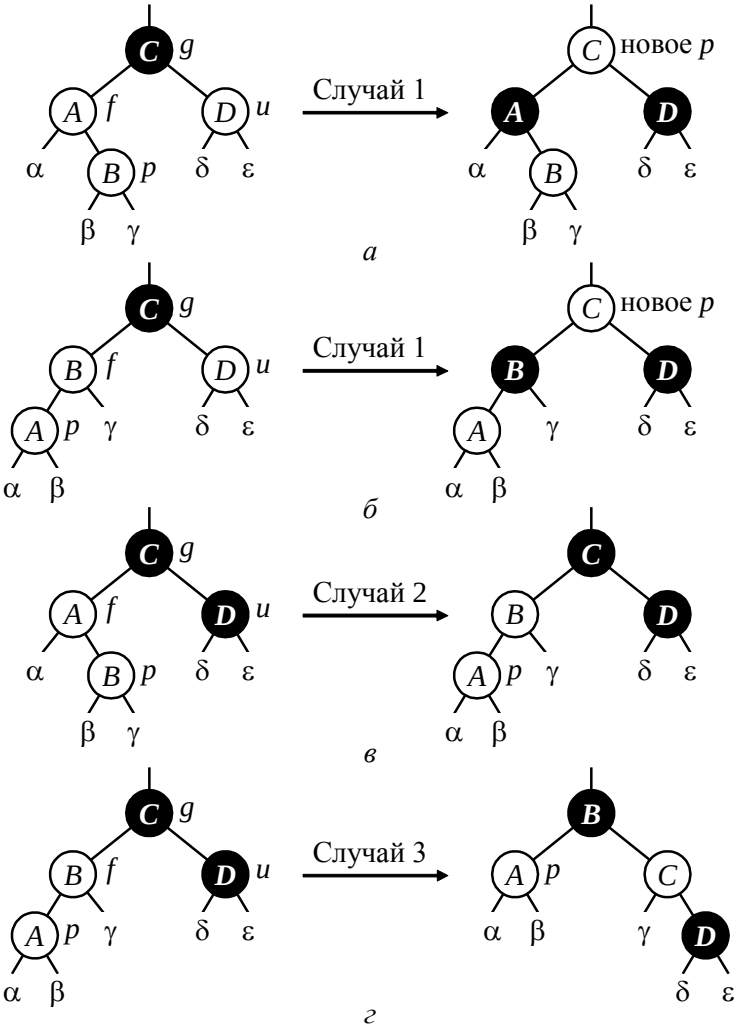

Рис. 4.12. Восстановление сбалансированности RB-дерева после включения:
а – вершина *и* красная, вершина *p* – правый сын вершины *f*;
б – вершина *и* красная, вершина *p* – левый сын вершины *f*;
в – вершина *и* черная и вершина *p* – правый сын вершины *f*;
г – вершина *и* черная и вершина *p* – левый сын вершины *f*

Исключение вершины. Как известно, из дерева всегда исключается вершина, имеющая не более одного непустого поддерева, т. е. вершина, имеющая одного или двух Λ-сыновей. Следует отметить, что при исключении красной вершины черные высоты не меняются, и красные вершины не могут стать смежными, т. е. RB-свойства не нарушаются. Поэтому нарушение RB-свойств может вызвать только исключение черной вершины. В этом случае любой проходивший через нее путь теперь содержит на одну черную вершину меньше. Поэтому для восстановления RB-свойств необходимо пройти путь от исключенной вершины вверх до корня дерева с информацией об уменьшении черной высоты поддерева, предпринимая необходимые действия.

Пусть p указывает на исследуемую на текущем этапе вершину (на начальном этапе – это вершина, занявшая место исключенной вершины, т. е. либо единственный сын исключенной вершины, либо фиктивная вершина, если у исключенной вершины были только Λ-сыновья), f – на вершину-отца и w – на вершину-брата вершины p. Вершина p является корнем поддерева, черная высота которого уменьшилась на единицу в результате исключения вершины. Тогда если вершина p – корень дерева или окрашена в красный цвет, то для восстановления RB-свойств достаточно перекрасить ее в черный цвет. В результате движение вверх прекращается и процесс завершается. Если же вершина p – черная и не является корнем дерева, то возможны восемь случаев. Четыре из них симметричны четырем другим, их разница в том, является ли черная вершина p левым или правым сыном вершины f. Рассмотрим один из симметричных вариантов, а именно – вершина p является левым сыном вершины f. Тогда возможны следующие случаи, представленные на рис. 4.13, где α, β, γ, δ, ε и ζ обозначают произвольные поддеревья (возможно, и пустые), серым цветом обозначены вершины, которые могут быть и красными, и черными.

Случай 1. Если вершина *w* красная (в этом случае их отец *f* окрашен в черный цвет и оба сына вершины *w* – черные), то вершина *w* окрашивается в черный цвет, а вершина *f* – в красный, и производится левое вращение вокруг вершины *f* (рис. 4.13, *а*). В результате этих действий случай 1 сводится к одному из случаев 2, 3 или 4.

Случай 2. Если вершина *w* черная и оба ее сына черные, то вершина *w* перекрашивается в красный цвет (рис. 4.13, *б*). В результате черная высота поддерева с корнем *f* уменьшается на единицу, так как поддерево с корнем *p* уже имело уменьшенную черную высоту, а перекрашивание вершины *w* уменьшило черную высоту поддерева с корнем *w*. Поэтому продолжается движение вверх с информацией об уменьшении черной высоты поддерева присваиванием указателю *p* значения указателя *f*. Следует отметить, что если процесс попал в случай 2 из случая 1, то вершина *p* (новое значение *p*) имеет красный цвет. Поэтому перекрашивание вершины *p* в черный цвет восстанавливает RB-свойства (ранее уменьшенная черная высота увеличивается на единицу), движение вверх прекращается и процесс завершается.

Случай 3. Если вершина w черная, ее левый сын красный, а правый – черный, то вершина w перекрашивается в красный цвет, левый сын – в черный, а затем применяется правое вращение вокруг вершины w (рис. 4.13, *в*). В результате новым братом w вершины p будет черная вершина с красным правым сыном, что сводит случай 3 к случаю 4.

Случай 4. Если вершина w черная, а ее правый сын красный, то вершина w окрашивается в цвет вершины f, затем вершина f и правый сын вершины w окрашиваются в черный цвет и производится левое вращение вокруг вершины f (рис. 4.13, *г*). В результате восстанавливаются RB-свойства, движение вверх прекращается и процесс завершается.

Поскольку высота RB-дерева есть $O(\log n)$, то операция исключения требует времени $O(\log n)$. Причем как только обнаруживается случай 1, 3 или 4, движение вверх прекращается и процесс завершается. При этом выполняется $O(1)$ операций и самое большее три вращения. До этого возможно несколько повторений случая 2, но при каждом повторении указатель p перемещается вверх по дереву, и никакие вращения не производятся. Таким образом, при восстановлении RB-свойств после исключения вершины производится не более трех вращений. Для сравнения – в АВЛ-деревьях может потребоваться $\lfloor h/2 \rfloor$ вращений для дерева высоты h.

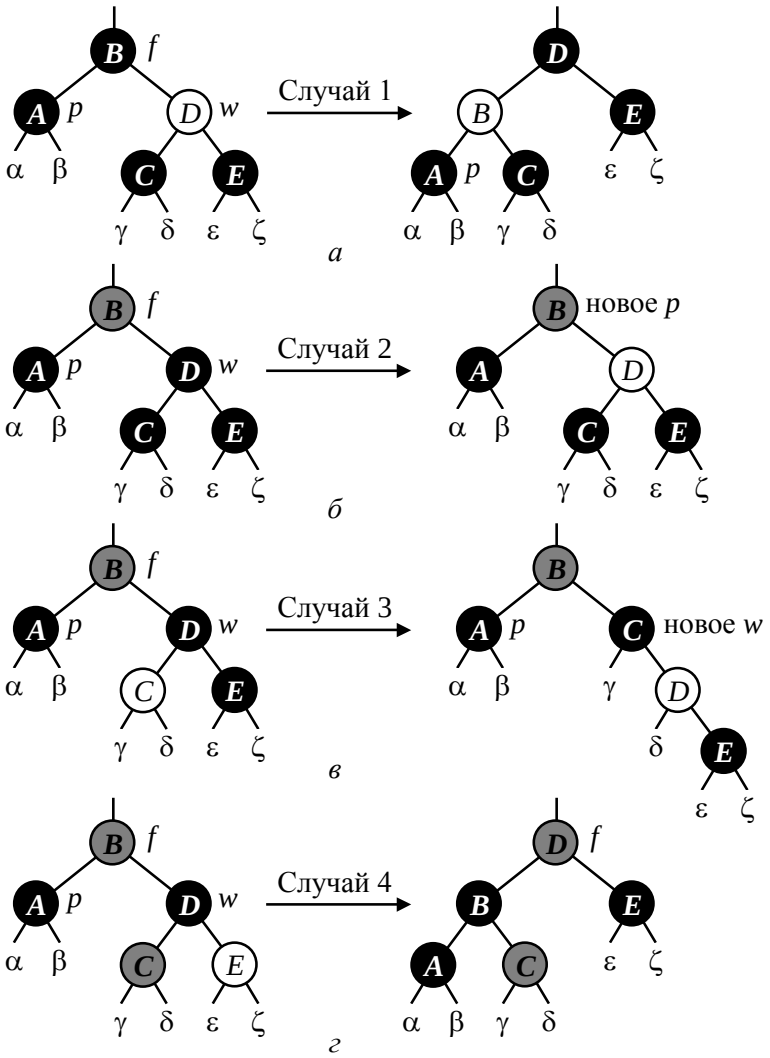

Рис. 4.13. Восстановление сбалансированности RB-дерева после исключения:
а – вершина *w* красная; *б* – вершина *w* черная и оба ее сына черные;
в – вершина *w* черная, ее левый сын красный, правый сын черный;
г – вершина *w* черная и ее правый сын красный

Как при включении, так и при исключении требуется пройти путь из корня в лист в обратном направлении. Для обеспечения возвращения необходимо модифицировать этап поиска так, чтобы сохранить спускающийся путь в стеке. Другой подход предполагает добавление в структуру узла специального поля *father*, указывающего на отца данной вершины. В этом случае нет необходимости в сохранении спускающегося пути в стеке.