Задание 7

Структуры данных

Литература:

1. **[Кормен 1]** *Кормен Т., Лейзерсон Ч., Ривест Р.*
   Алгоритмы: Построение и Анализ.
   М.: МЦНМО, 2002.

2. **[Кормен 2]** *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.*
   Алгоритмы: Построение и Анализ. (2-е изд.)
   М.: Вильямс, 2005.

Домашнее задание

Задачи 32-34 из КДЗ-2017.

Задача 1. Изучите алгоритм удаления вершины из двоичного дерева поиска, описанный в **[Кормен 2]**. Докажите корректность данной процедуры.

Задача 2. Известно, что в структуре данных потребуется хранить k-элементное подмножество A n-элементного множества. После того как в структуру данных будет загружено множество A, с помощью неё будет нужно проверить принадлежит ли A элемент x. Для этого можно совершить не более t запросов к структуре: каждый запрос q представляет собой конечную строку битов, ответ на каждый запрос – 1 бит.

Структура данных представляет собой таблицу с двумя столбцами: первый столбец состоит из всевозможных запросов q (известных заранее), а правый из битов-ответов на запрос – правый столбец формируется после загрузки в структуру множества A. Пусть $s(n, k, t)$ – минимальное количество строк в такой таблице которое достаточно отвести под такую структуру данных.

1. Чему равно $s(n, k, 1)$?
2. Найдите $\min\limits_{t}(s(n, k, t))$, при каком t он достигается?

Красно-чёрные деревья

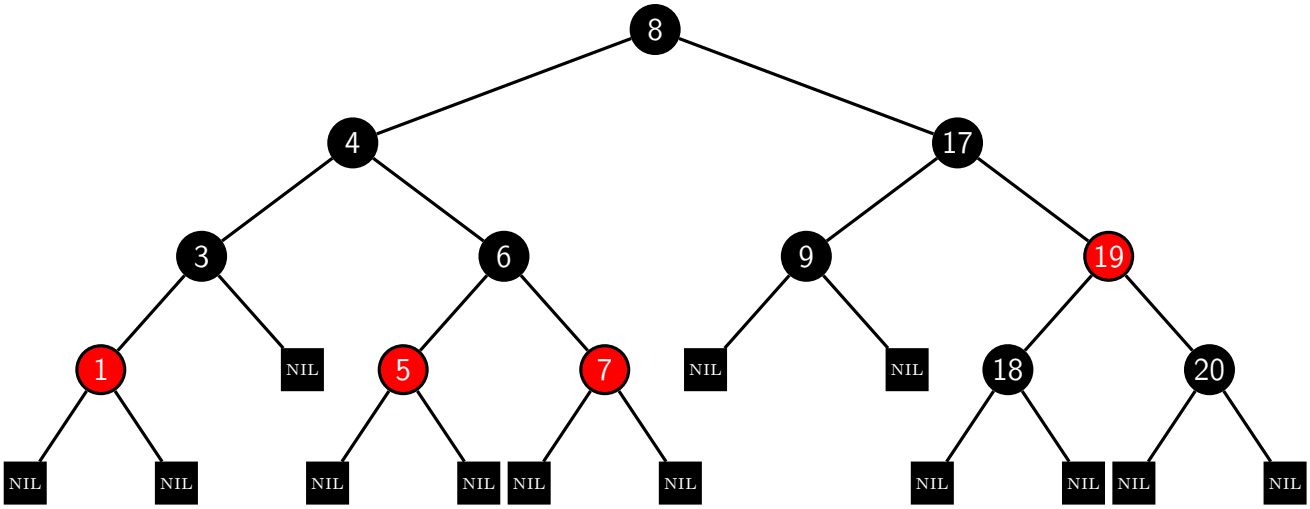

Рис. 1: Красно-чёрное дерево.

В данном разделе мы приведём определение красно-чёрного дерева и опишем процедуру добавления в него новой вершины. В конце раздела есть задачи на дом. Изложение ведётся на основе **[Кормен 2]**. Здесь я приведу более подробные картинки, опишу алгоритм добавления вершины в красно-чёрное дерево и докажу его корректность. За более формальным изложением алгоритма (псевдокоду) стоит обращаться к **[Кормен 1]**(**[Кормен 2]**).

Красно-чёрное дерево – это двоичное дерево поиска специального вида. Во-первых, на него налагается следующее техническое условие: если у вершины двоичного дерева нет дочерней вершины, то тогда мы добавляем к ней соответсвующее ребро в терминальный лист NIL. Мы будем называть листьями красно-чёрного дерева – листья двоичного дерева поиска, а узлы NIL – терминальными листьями. Во-вторых, для красно-чёрного дерева выполнены следующие условия:

1. Каждая вершина дерева либо красная, либо чёрная.
2. Каждый терминальный лист – чёрный.
3. Корень – чёрный.

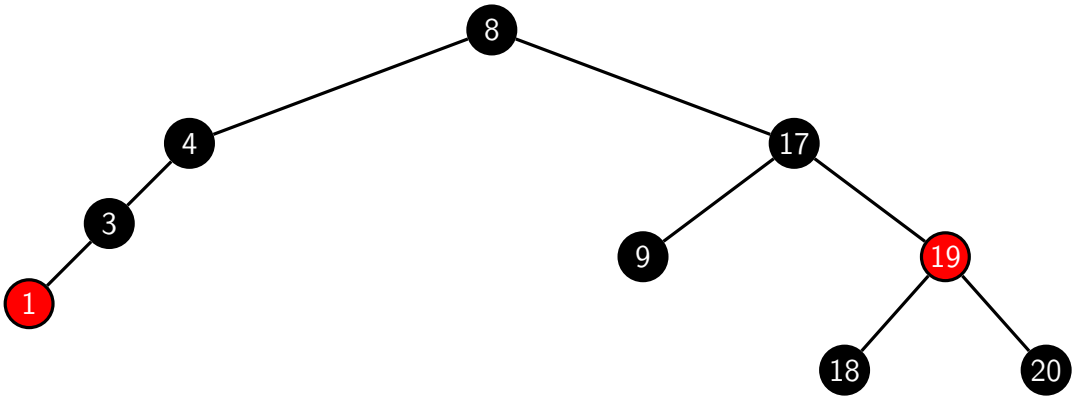

Рис. 2: Не красно-чёрное дерево.

4. На пути из корня в каждый терминальный лист находится одинаковое количество чёрных вершин.

5. Обе дочерних вершины красной вершины – чёрные.

На рис. 1 приведён пример красно-чёрного дерева[1]. Терминальные узлы важны для условия баланса. В частности, на рис. 2, приведено не красно-чёрное дерево, хотя оно будет удовлетворять определению, если не добавлять терминальные листы и изменить соответственно условие 2. И это отличие существенно: при добавлении вершины со значением 6 в дерево на рис. 2 будет нарушено условие баланса 4.

Первый шаг процедуры добавления вершины в красно-чёрное дерево совпадает с алгоритмом добавления вершины в обычное двоичное дерево поиска: мы находим соответсвующий значению новой вершины терминальный лист и заменяем его на добавляемую вершину. Мы красим новую вершину в красной и добавляем к ней терминальные листья.

Однако, данное действие может нарушить условие 5, что случится если мы добавим в дерево на рис. 1 вершину со значением 2. Это нарушение можно исправить путём перестроения дерева и перекраски его вершин, сохраняющей при этом свойства 1-5. В качестве процедуры перестроения дерева мы используем операцию поворота, приведённую на рис. 3.

[1]Картинки в задании построены на основе стиля https://github.com/MartinThoma/LaTeX-examples/blob/master/tikz/red-black-tree/red-black-tree.tex

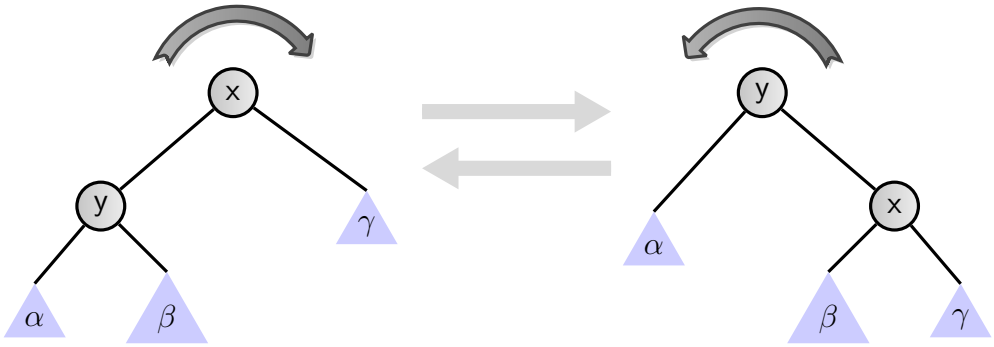

Рис. 3: Повороты

Напомним, что если спроецировать значения всех вершин двоичного дерева поиска на ось Ox, то мы получим неубывающую числовую последовательность. Более того, это является необходимым и достаточным условием для того, чтобы двоичное дерево было двоичным деревом поиска. Исходя из этого мы и будем обосновывать корректность операций поворотов.

Обратимся к рис. 3. Греческие буквы в треугольниках обозначают на нём поддеревья. Обозначим через $[\alpha]$ отрезок, полученной проекцией значений всех вершин поддерева α на ось Ox. Тогда проекции на ось Ox значений вершин левого и правого деревьев будут совпадать: они равны $[\alpha]y[\beta]x[\gamma]$. Значит, эти деревья эквивалентны: одно можно заменить на другое не нарушив при этом порядка на вершинах. Также их можно заменить как поддеревья в некотором двоичном дереве поиска и не нарушить при этом его корректность. Замена левого дерева на правое называется *правым поворотом*, а обратная замена – *левым поворотом*.

Мы будем использовать повороты вместе с допустимой перекраской вершин, чтобы решать коллизии, возникающие в ходе добавления новой вершины в красно-чёрное дерево. Общий план алгоритма решения коллизий состоит в следующем: мы будем преобразовывать красно-чёрное дерево таким образом, чтобы в результате преобразования конфликтующие вершины поднимались вверх; при этом не будет нарушаться условие 4 о сохранении чёрной высоты. В результате этого процесса коллизия либо разрешится на этапе подъёма, либо достигнет корня. Однако, корень всегда можно перекрасить в чёрный цвет не нарушая ни одного из свойств 1-5! Ведь тогда количество чёрных вершин на каждом пути из

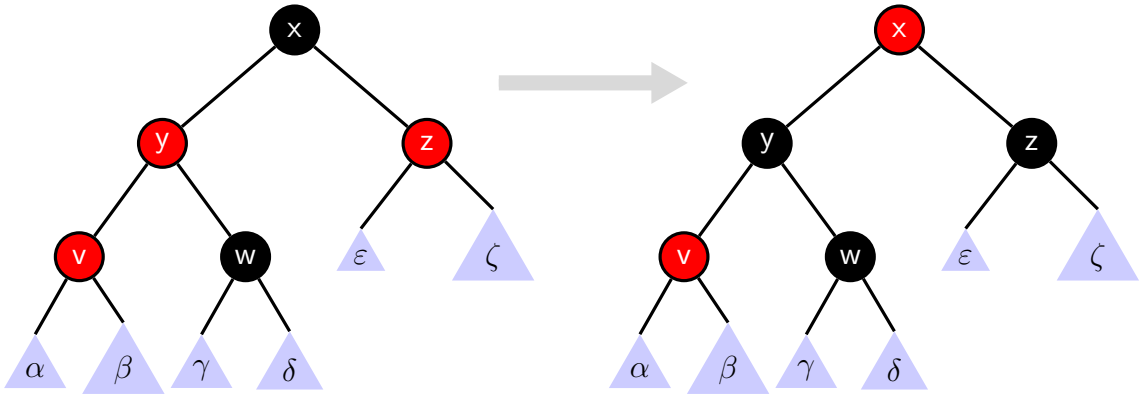

Рис. 4: Первый случай

корень в терминальный лист лишь возрастёт на 1.

Для контроля за выполнением условия 4, мы вводим функцию *чёрной высоты* $\mathbf{bh}(u)$ вершины u. Она возвращает количество чёрных вершин на пути из u в каждый терминальный лист, считая u, если она чёрная. Функция **bh** не определена, если из u есть два пути в терминальные листы с разным количеством чёрных вершин.

Итак, начнём рассматривать всевозможные случаи, возникающие в результате коллизий. Заметим, что поскольку корень всегда чёрный, то коллизии всегда возникают на глубине не меньшей, чем 3. При этом, в результате добавления вершины возникает ровно одна коллизия, и мы будем поддерживать этот инвариант. Так, на рис. 4 коллизия возникла из-за добавления или подъёма вершины v. При этом, чёрная высота вершин v и w одинаковая, также как у y и z, а w – чёрная вершина, иначе в дереве было бы две коллизии. Заметим, что вершина x всегда покрашена в чёрный: иначе была бы коллизия между x и y.

Первый случай самый простой: для его решения достаточно перекрасить вершину y и z в чёрный цвет, а x – в красный. Данная перекраска не меняет чёрную высоту указанных вершин. В результате перекраски x, коллизия либо вовсе исчезнет, либо поднимется выше. Если вершина x была корнем дерева, то её можно просто перекрасить в чёрный.

Также под первый случай попадает симметричный случай, когда вершина v – чёрная, а w – красная. Эта симметрия, в отличие от второго и третьего случаев, не играет роли. Также мы будем далее опускать симметричные случаи конфликтов в правых поддеревьях x.

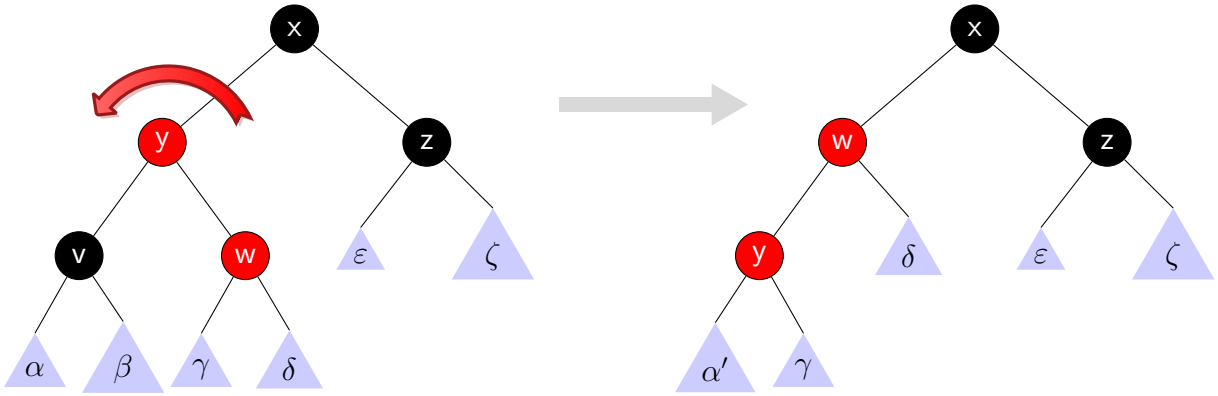

Рис. 5: Второй случай

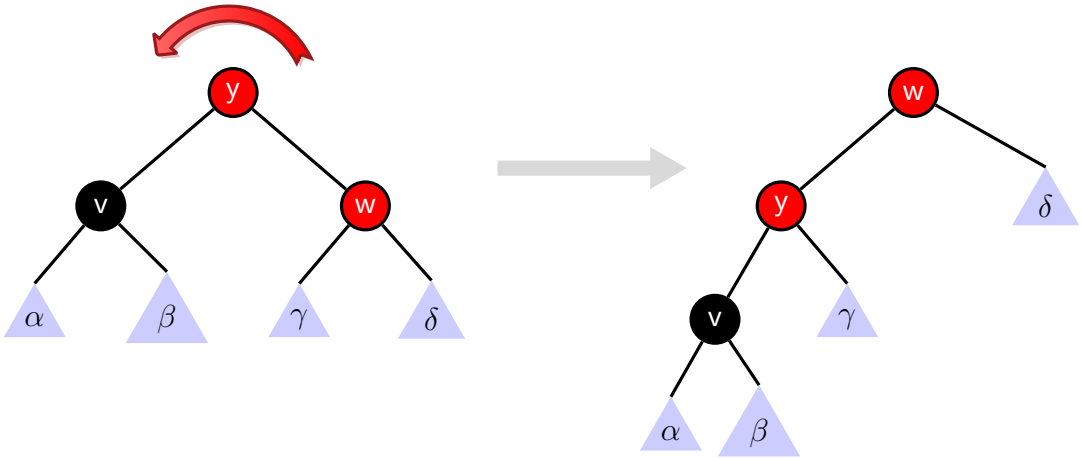

Рис. 6: Сведение поворотом к третьему случаю

Перейдём теперь ко второму случаю, изображённому на рис. 5. Его анализ довольно короткий: мы преобразуем его к третьему случаю левым поворотом. Поскольку поворот, как мы показали выше, преобразует поддерево двоичного дерева поиска в эквивалентное, то новых коллизий кроме коллизии между y и w не возникнет.

Третий случай, пожалуй, самый содержательный. Заметим, что чёрная высота вершин v, w, y, z совпадает, в то время как $\mathbf{bh}(x) = \mathbf{bh}(y) + 1$. Применив правый поворот относительно вершины x мы получаем, что $\mathbf{bh}(v) = \mathbf{bh}(x) - 1$, а значит из y есть два пути в листья с разным количе-

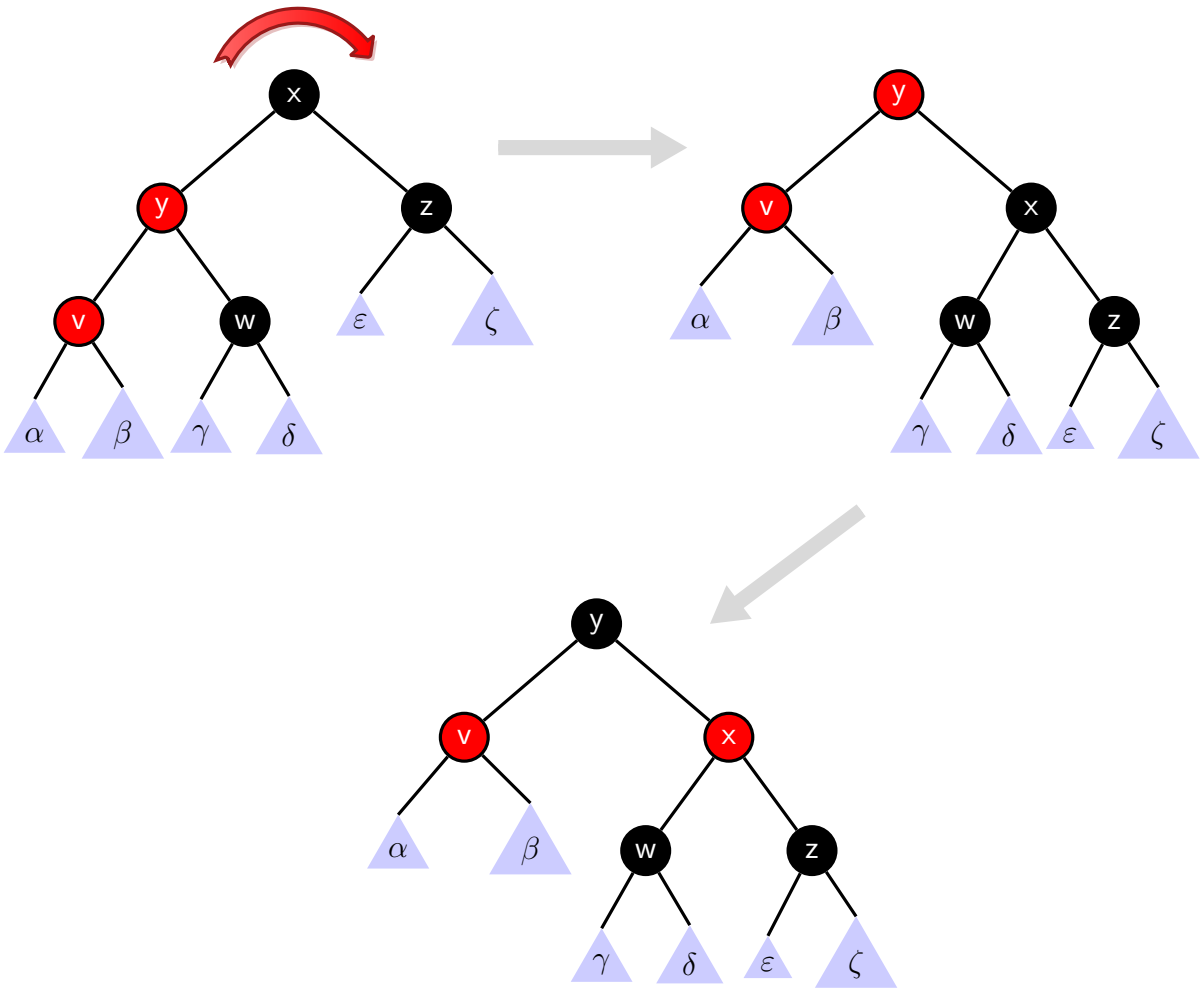

Рис. 7: Третий случай

ством чёрных вершин. Исправим это, обменяв цвета вершин x и y: теперь чёрная высота вершин x и v выровнялась, а значит свойство 4 осталось выполненным. Заметим, что при повороте не были нарушены свойство 4 для путей, проходящих через x, поскольку $\mathbf{bh}(w) = \mathbf{bh}(z) = \mathbf{bh}(x) - 1$.

Итак, алгоритм решения коллизий состоит в применении указанных преобразований, в зависимости от одного из трёх случаев коллизии. Мы показали как в каждом случае либо решить коллизию, либо переместить коллизию выше, не нарушив оставив при этом свойства 1-4 выполненными. Таким образом, либо коллизия исчезнет не достигнув корня, либо

корень будет окрашен в красный цвет – тогда перекрасим корень в чёрный.

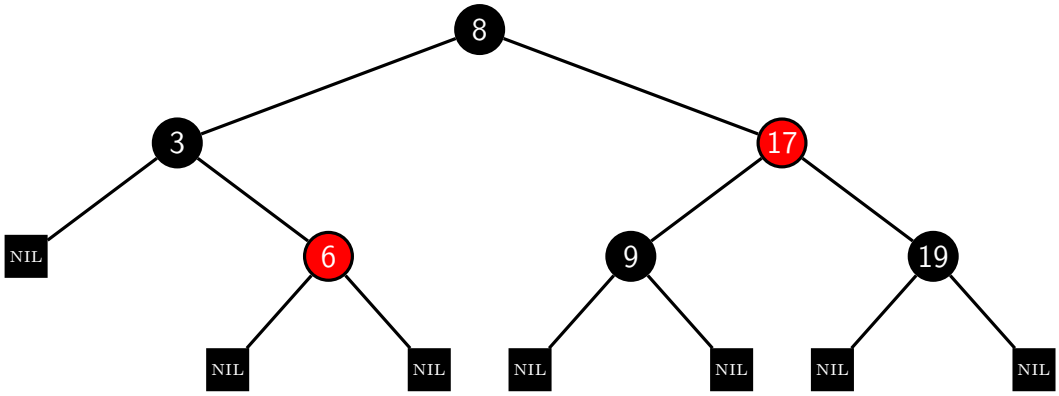

Рис. 8: К задаче 3.

Задача 3. Вставьте в красно-чёрное дерево с рис. 8 вершину со значением 5.