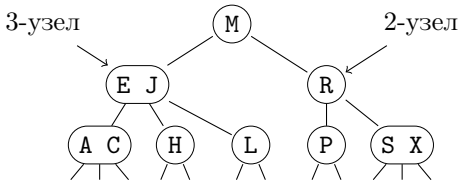
Т.В. Абрамовская

Материалы для курса «Дискретная математика»,
мат-мех, 171 группа, 2016-17

16/12/2016 **Двоичные деревья поиска. 2-3 и красно-чёрные деревья.**

Напомню, **2-3 дерево**, если оно не пусто, — это

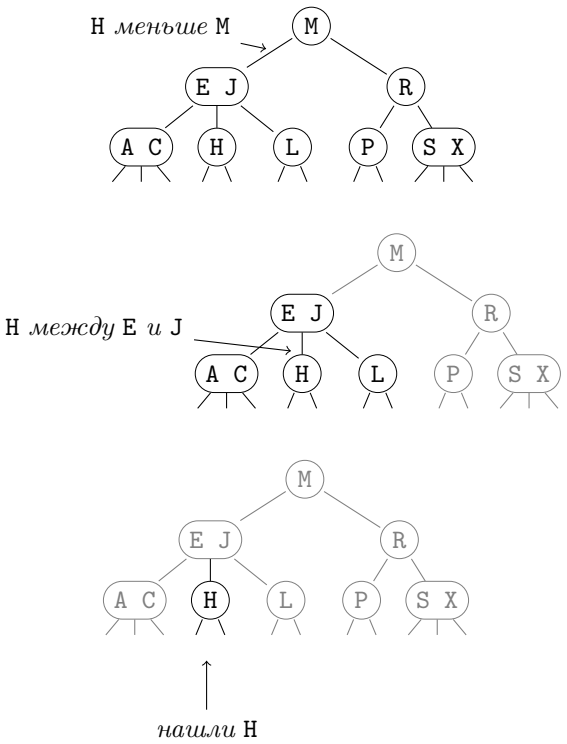
- либо 2-узел, которому соответсвует единственное значение ключа[1] и два поддерева: левое (если не пусто) содержит узлы с меньшими значениями ключей, правое — с большими;
- либо 3-узел, с двумя ключами и тремя поддеревьями: левое с меньшими значениями ключей, среднее — с ключами между значениями ключей корня, правое — с большими значениями ключей.



Деревья, которые мы будем строить, будут *совершенными*, т. е. такими, что все расстояния от корня до каждого листа будут равны.

Поиск в 2-3 дереве интуитивно совершенно ясен, он мало отличается от поиска в двоичном дереве поиска: в том случае, когда рассматривается 3-узел, если искомый ключ лежит между ключами 3-узла, то дальнейший поиск инициируется в его среднем поддереве.

Например, проследим за поиском элемента с ключом H в дереве, нарисованном выше.



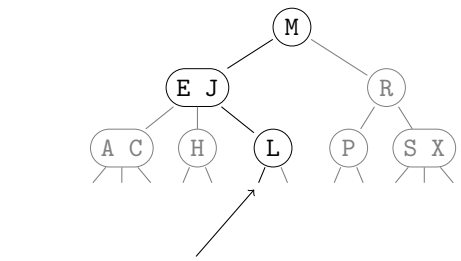
[1]Здесь я напомню, мы занимаемся поиском некоторого значения в множестве значений, где каждому элементу приписан уникальный ключ. По этим ключам организовывается поиск.

Вставка нового элемента с новым ключом всегда сопряжена с предварительным поиском по ключу, который должен закончится ссылкой на пустое дерево, и затем непосредственное добавление нового элемента. Здесь нас ждут несколько вариантов.

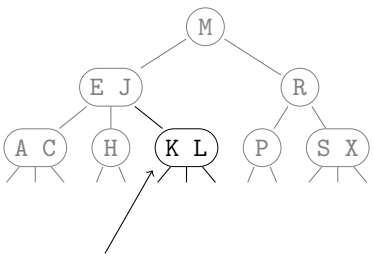
Будем считать, что элемент добавляется в совершенное дерево, и вставка будет организована так, чтобы это свойство сохранилось.

Вставка в 2-узел — это простая трансформация 2-узла в 3-узел, очевидно, сохраняющая совершенность.

Для примера, добавим элемент с ключом K.



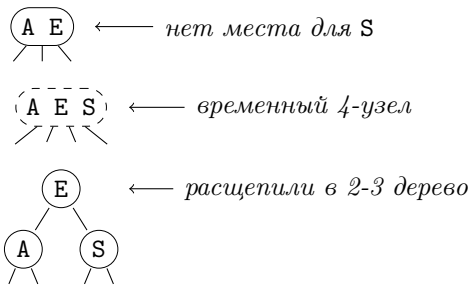
поиск K *заканчивается здесь*



заменяем 2-узел новым 3-узлом, содержащим K

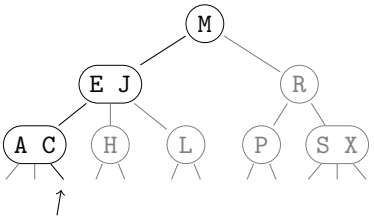
Вставку в 3-узел рассмотрим для начала на примере дерева, состоящего из единственного 3-узла. Представим себе, что образуется 4-узел с тремя ключами и, соответственно, четырьмя поддеревьями, а затем расщепим его в три новых 2-узла: средний из ключей отправляется в корень, слева от которого повесим узел с меньшим ключом, справа — с большим.

Вставим S в узел с ключами A и E.

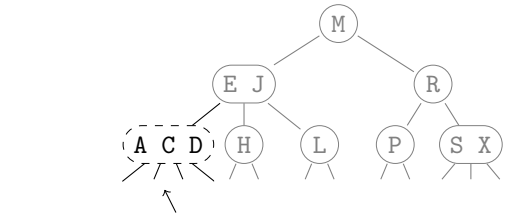


В общем случае, 2-узел-родитель будет образовываться только тогда, когда расщепляется корень дерева. Если же расщеплялся не корень, мы попробуем добавить ключ к предку.

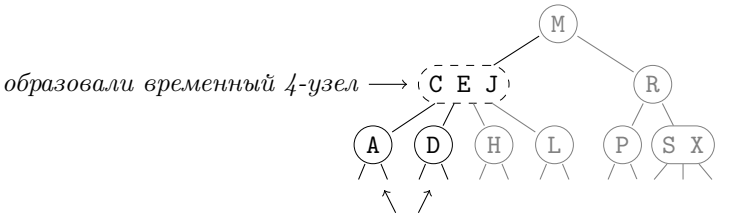
Для примера, вставим элемент с ключом D.



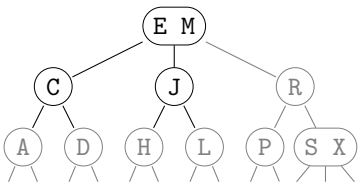
поиск D *заканчивается здесь*



образовали временный 4-узел



разбили 4-узел, отдав среднее значение ключа предку

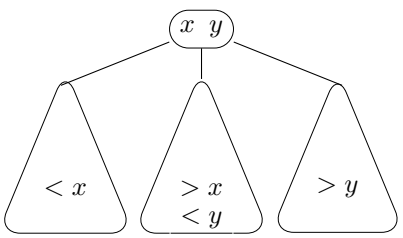


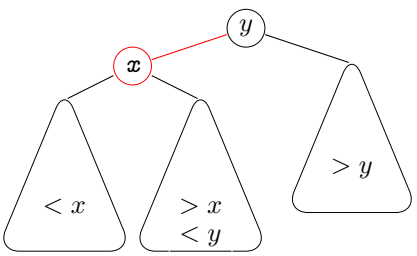
Добавили к 2-узлу в корне — образовали 3-узел, который не требует расщепления.

Замечательный факт состоит в том, что для вставки элемента в 2-3 дерево всякое локальное перестроение не требует дополнительного просмотра или модификации других частей дерева (в АВЛ деревьях, например, нам необходимо знать, левое или правое поддерево имеет бóльшую высоту). И, несомненно, дерево остаётся совершенным.

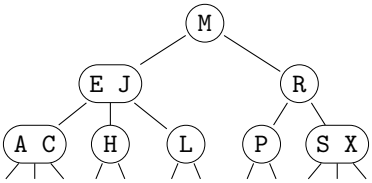
Далее нас интересует адаптация двоичного дерева поиска к 2-3 деревьям.

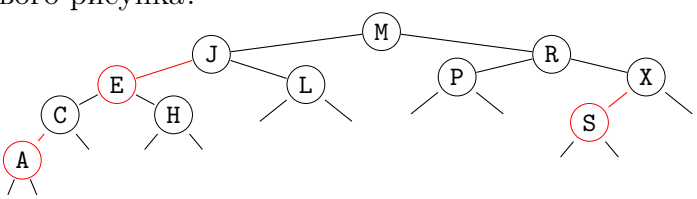
Мы воспользуемся тем, что ключи и поддеревья, входящие в 3-узел, можно представить как пару 2-узлов с соответствующими поддеревьями. Носителем информации о том, что это пара узлов особого рода, будет дочёрний левый узел: он помечается красным.





Как тогда будет выглядеть дерево с первого рисунка?





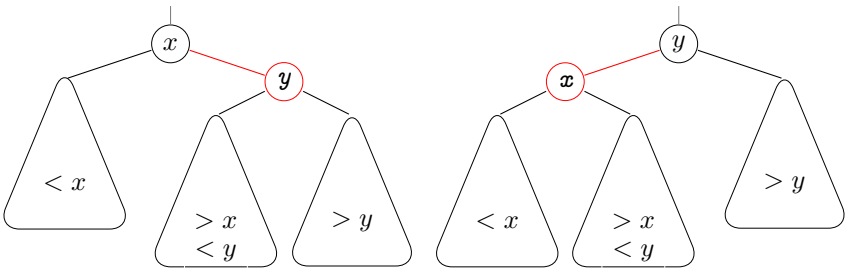
Красно-чёрные деревья — это двоичные деревья поиска, где каждый узел снабжён маркером цвета и может быть либо красным, либо чёрным: корень дерева — всегда чёрный, как и оба потомка любого красного узла; «пустые» узлы считаются чёрными; все пути от корня до каждого листа содержат одинаковое число чёрных узлов.

Красно-чёрные деревья, как нетрудно заметить, — более общее понятие, чем та структура, которая получится, если в 2-3 дереве все 3-узлы представить парами, как показано выше. Нас же интересует их частный случай — добавим требование иметь не более одного красного потомка, причём если только левого (именно так будут представлены 3-узлы). Далее будут определены операции преобразования, и в процессе иногда могут появляться правый красный потомок или даже пара красных потомков, но всякий раз дерево будет исправляться так, что итоговое дерево будет иметь только левых красных потомков.

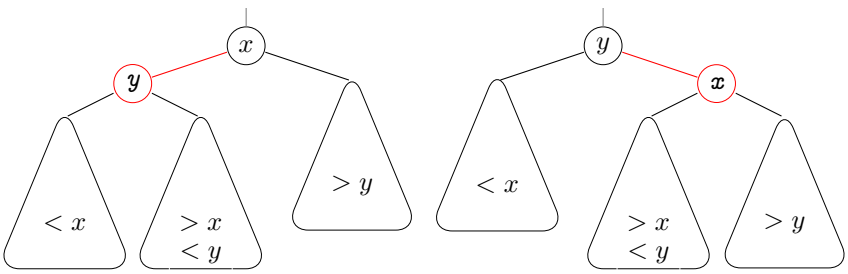
Поиск и вставка происходит тут так же, как в обычном двоичном дереве поиска. Но раз нас интересует реализация 2-3 дерева при помощи красно-чёрных, то нужно ещё определить то, что мы раньше называли «трансформацией 2-узла в 3-узел» и «составление и последующее расщепление 4-узла». О сбалансированности, конечно, речи не идёт, но используются идеи вращений из АВЛ деревьев.

Покажем схематично основные преобразования. Чуть позже мы рассмотрим пример, и станет понятно, когда и для каких целей эти операции будут использоваться.

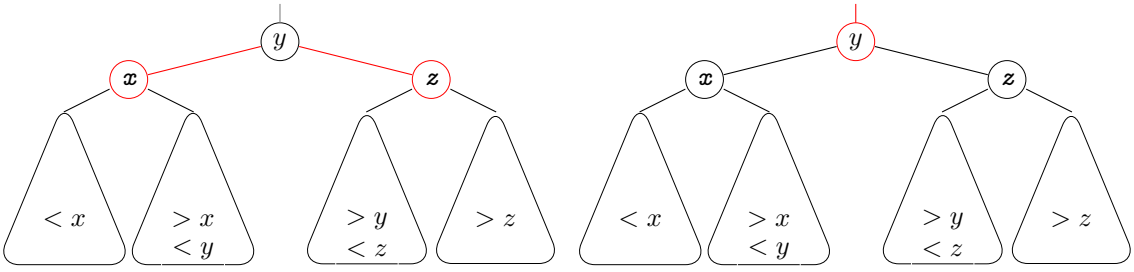
Поворот влево вокруг x (при этом сам узел x может быть как красным, так и чёрным! И его цвет перейдёт узлу y после вращения):



Поворот вправо вокруг x (аналогично, узел x может быть и красным, и чёрным):



Смена цветов (узел y может быть левым или правым потомком):



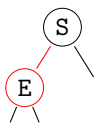
Теперь рассмотрим, что происходит после вставки элемента в двоичное дерево поиска. Будем помнить, что всякий раз преобразования делаются до тех пор, пока все красные потомки не станут левыми. Далее выписаны правила, а после рассмотрен пример.

- Новый узел добавляется как в обычном двоичном дереве поиска. Если это не корень дерева, то его цвет меняется на красный.
- Перестроение дерева инициируется сменой цвета узла.
- Появление красного левого потомка у чёрного узла не требует дальнейших изменений. (Например, в примере ниже — это случай добавления `E`.)
- Появление красного правого потомка у чёрного узла, у которого левый потомок — красный, требует смены цветов (как при добавлении `P`).
- Появление красного правого потомка у чёрного узла, у которого левый потомок — чёрный, требует поворота влево (как при добавлении `C`).
- Появление красного левого потомка у красного узла требует поворота вправо вокруг его предка, а затем смены цветов (как при добавлении `A`).
- Появление красного правого потомка у красного узла требует поворота влево (что приведёт к красному левому потомку, см. предыдущий пункт). (Этот случай возник, например, при добавлении `P`, после смены цветов `M` и его потомков.)
- Корень всегда помечается чёрным.

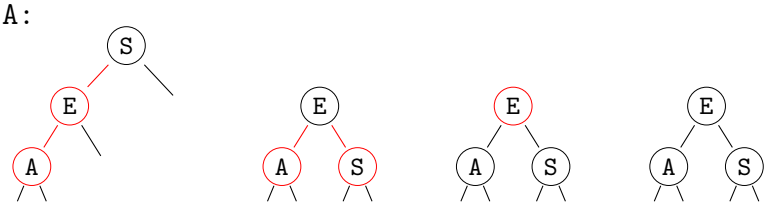
Будем последовательно добавлять ключи `S E A R C H E X A M P L E`. Некотые символы повторяются, будем пропускать уже существующие ключи.

`S:`

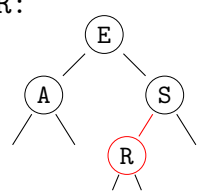
`E:`



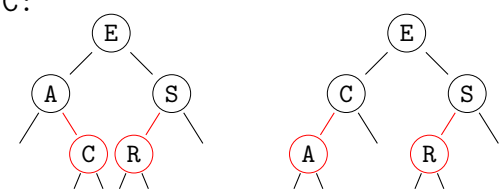
`A:`



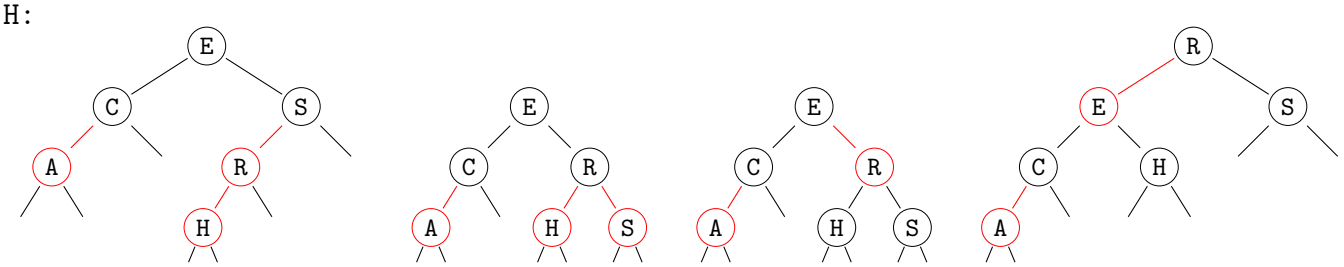
`R:`



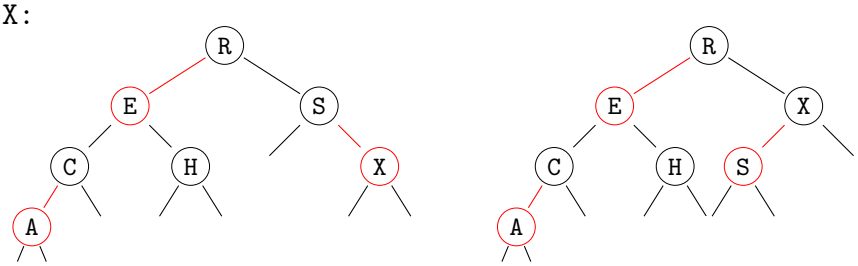
C:


E
A
S
C
R
E
C
S
A
R

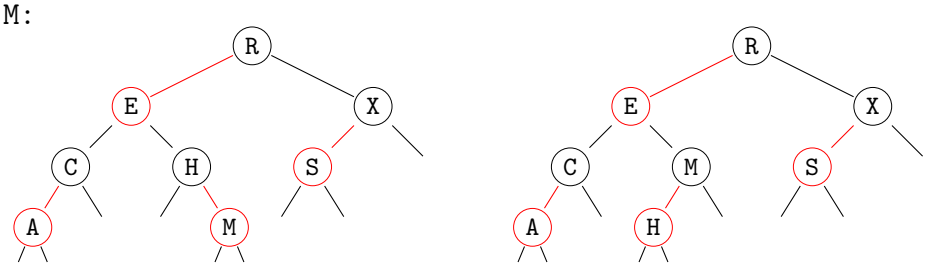
H:


E
C
S
A
R
H
E
C
R
A
H
S
E
C
R
A
H
S
R
E
S
C
H
A

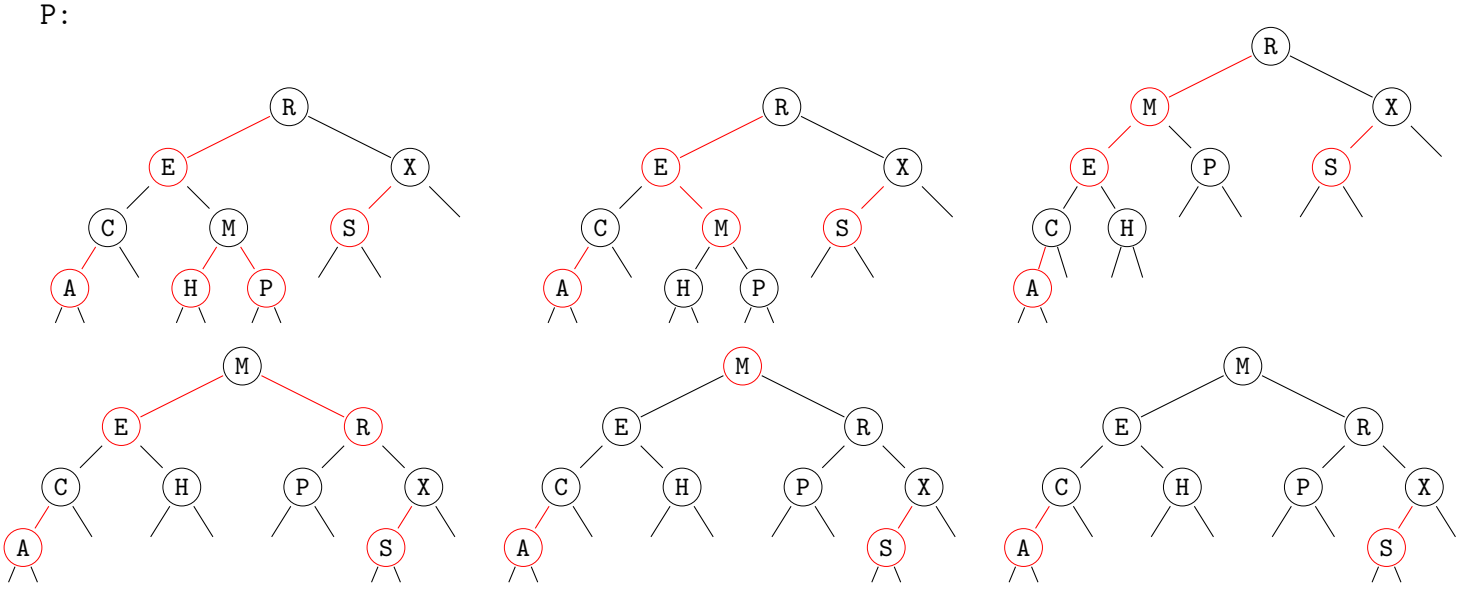
X:


R
E
S
C
H
X
A
R
E
X
C
H
S
A

M:


R
E
X
C
H
S
A
M
R
E
X
C
M
S
A
H

P:


R
E
X
C
M
S
A
H
P
R
E
X
C
M
S
A
H
P
R
M
X
E
P
S
C
H
A
M
E
R
C
H
P
X
A
S
M
E
R
C
H
P
X
A
S
M
E
R
C
H
P
X
A
S

L: