Удаление вершин из АВЛ-дерева

Процесс немного более сложный, чем добавление.

Хотя алгоритм операции балансировки *остается тем же*, что и при включении вершин.

Балансировка по-прежнему выполняется с помощью одного из четырех поворотов вершин:

LL, LR, RL, RR.

Если при добавлении вершины название поворота определяется ***путем от вершины в которой нарушен баланс, к добавленной***, то при удалении картина симметрична.

Если удалена вершина из левого поддерева, потребуется **один из правых поворотов**

(RR или RL),

а при удалении из правого поддерева потребуется **один из левых поворотов**

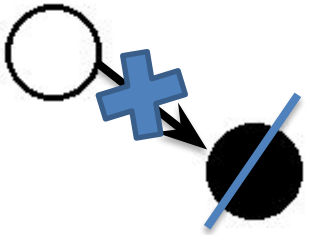
(LL или LR)

Идея алгоритма:

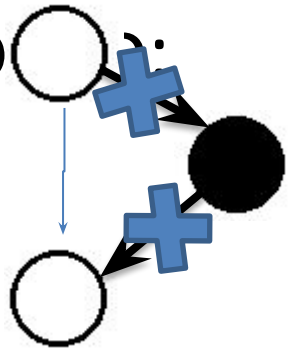
1. Удаляем вершину так же, как это делалось для случайного дерева поиска (СДП);
2. Двигаясь назад по пути поиска от удаленной вершины к корню дерева, будем пересчитывать баланс каждой вершины и при необходимости восстанавливать с помощью поворотов.

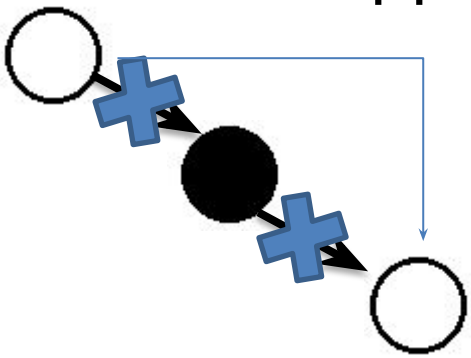
Нарушение баланса возможно **в нескольких вершинах**, в худшем случае - во всех вершинах по пути поиска.

1. Если удаляемая вершина не имеет поддеревьев:

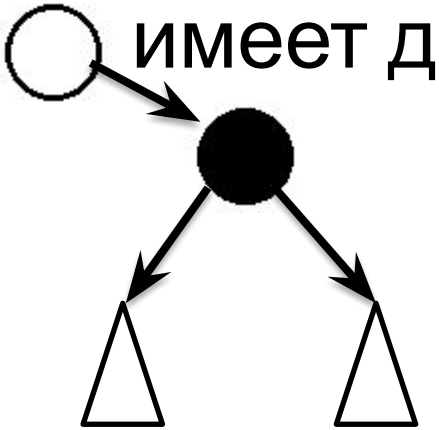


2. Если на удаляемой вершине одно поддер[illegible]:





3. Если верши[illegible] имеет два поддерева:



Правила удаления:

а) На место удаляемой вершины ставится *наибольшая вершина из её левого поддерева,* т.е. самая правая вершина из левого поддерева, не имеющая правого поддерева.

б) На место удаляемой вершины ставится *наименьшая вершина из её правого поддерева,* т.е. самая левая вершина из её правого поддерева, не имеющая левого поддерева.

Будем строить алгоритмы на правиле «а»

Как и в случае добавления вершин, введем логическую переменную *Умен*, показывающую, уменьшилась ли высота поддерева.

Балансировка производится только, когда ***Умен=истина***. *Это значение присваивается, если:*

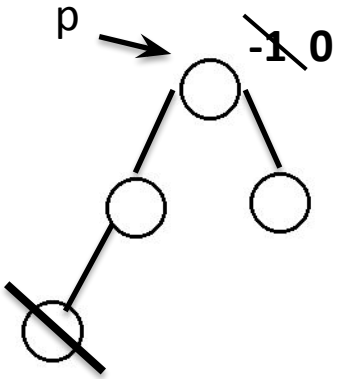
1. Обнаружена и удалена вершина;

2. Уменьшилась высота в ходе балансировки.

Введем две симметричные процедуры балансировки:

BL – при удалении из левого поддерева,

BL (Vertex *&p)



IF(p-->Bal = -1) p-->Bal = 0

ELSE IF(p-->Bal = 0) p->Bal = 1, *умен*=нет

ELSE IF (p-->Bal = 1)

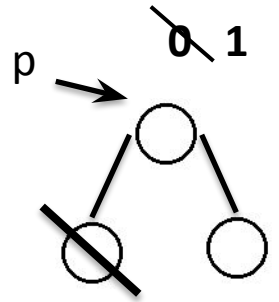
IF (p-->Right-->Bal >= 0) <RR1-поворот>

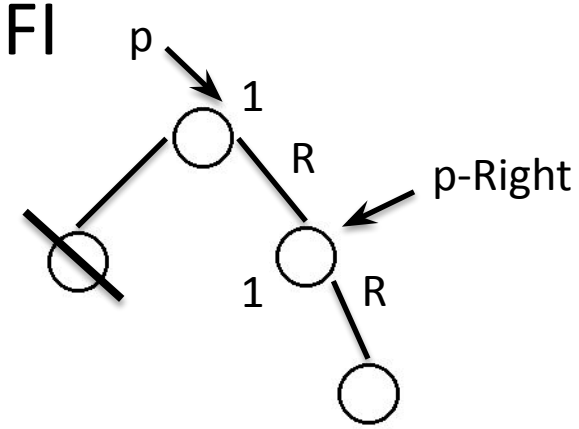
ELSE <RL-поворот>

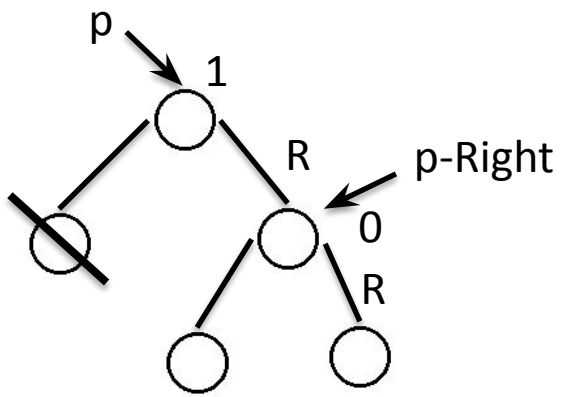
FI

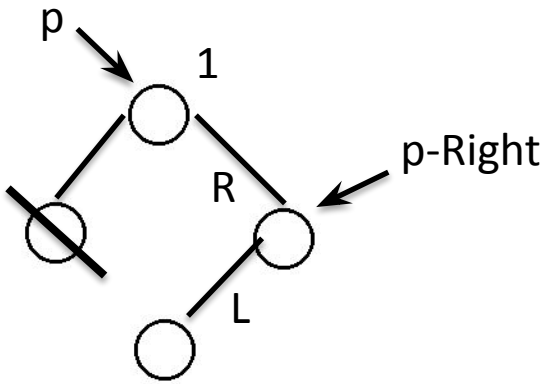
FI

FI

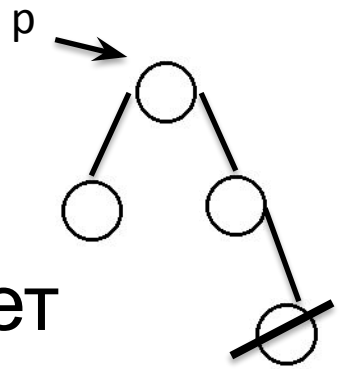








BR (Vertex *&p)



IF (p-->Bal = 1) p-->Bal = 0

ELSE IF (p-->Bal = 0) p-->Bal = -1, *умен*=нет

ELSE IF (p-->Bal = -1)

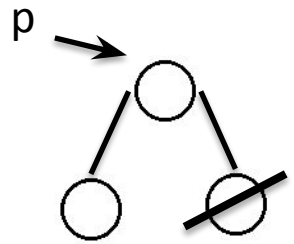
IF (p-->Left-->Bal <= 0) <LL1-поворот>

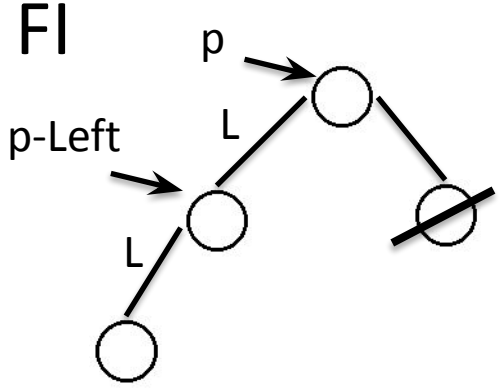
ELSE <LR-поворот>

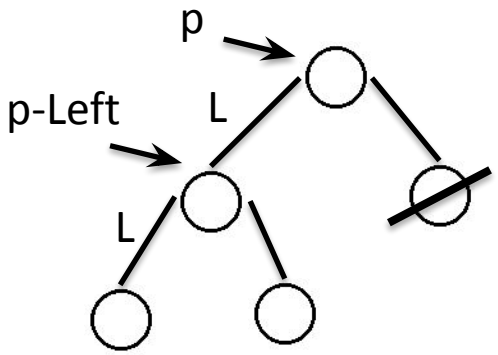
FI

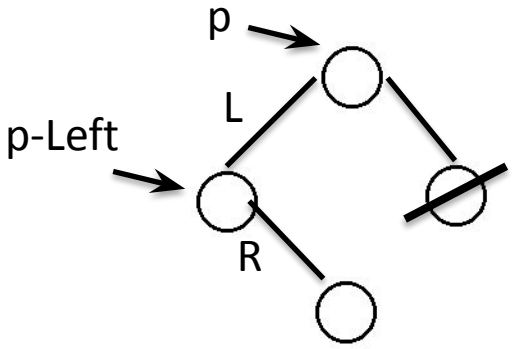
FI

FI









При добавлении вершины не может быть случая, когда **p-->Left-->Bal = 0** или **p-->Right-->Bal = 0**.
Чтобы учесть эти случаи, LL-поворот необходимо изменить на ***LL1-поворот***, а RR-поворот – на ***RR1-поворот*** .

LL1-поворот

q = p-->Left
IF (q-->Bal = 0) q-->Bal = 1, p-->Bal = -1, Умен = нет
ELSE q-->Bal = 0, p-->Bal = 0
FI
p-->Left = q-->Right
q-->Right = p
p = q

Аналогично изменяется RR-поворот на RR1-поворот, повороты LR и RL – не изменяются.

*DELETE (x, vertex *&p)*

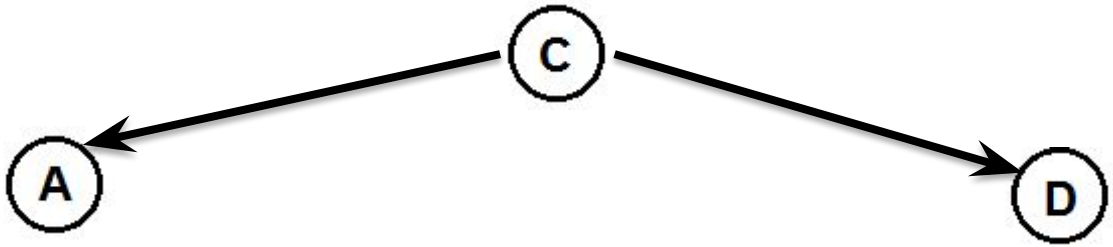
```
IF (p = NULL)
ELSE IF (x < p-->Data)   DELETE (x, p-->Left)
                              IF  Умен=да  BL(p)  FI
      ELSE IF (x > p-->Data)   DELETE (x, p-->Right)
                                    IF  Умен=да   BR(p)  FI
ELSE  q = p
       IF (q-->Left = NULL) p = q-->Right,  Умен=да
       ELSE IF (q-->Right = NULL) p = q-->Left,  Умен=да
       ELSE  del (q-->Left)
                IF  Умен=да  BL(p)  FI
       FI
free(q)
FI
```

Функция **del** удаляет вершину, имеющую два поддерева, т.е. заменяет ее на самую большую (правую) вершину из левого поддерева.

del (vertex *&r)

```
IF (r-->Right != NULL)  del (r->Right)
                                IF Умен=да BR(r) FI
ELSE q-->Data = r-->Data
        q = r
        r = r-->Left
        Умен = да
FI
```

B 9 2 4 1 7 E F A D C



Замечание: «По экспериментальным оценкам на каждые два включения встречается ***один поворот***, а при исключении поворот происходит ***в одном случае из пяти***». Н. Вирт

Трудоёмкость работы с АВЛ-деревьями

Поиск элемента с заданным ключом, **включение элемента**, **удаление элемента** – все операции производятся в худшем случае за $\mathbf{O(\log_2 n)}$ операций сравнения.

Отличия между процедурой включения и удаления: включение может привести самое большое к одному повороту, а удаление может потребовать повороты во всех вершинах по пути поиска.

Наихудший случай с точки зрения количества поворотов – удаление самой правой вершины у плохого АВЛ-дерева

T_5 : H=5