Дерево поиска	Высота		
	Минимальная	Средняя	Максимальная
ИСДП	$\log_2(n+1)$	$\log_2(n+1)$	$\log_2(n+1)$
СДП	$\log_2(n+1)$	**1,39** $\log_2(n+1)$	n

Рекурсивная процедура добавления в случайное дерево поиска

Добавить рекурсивно (D, Vertex*&p)
IF (p=NULL)
    память (p), p->Data=D,
    p->Left=NULL, p->Right=NULL
ELSE IF (D< p->Data)
        Добавить рекурсивно(D, p->Left)
    ELSE IF (D> p->Data)
            Добавить рекурсивно(D, p->Right)
        ELSE <Вершина есть в дереве>
FI

Вызов процедуры:

Добавить рекурсивно (D, root)

Удаление вершин из СДП

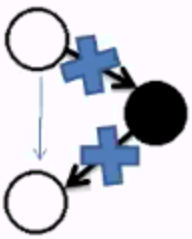
Идея удаления:
Сначала нужно найти вершину с ключом X, двигаясь влево или вправо по пути поиска, пока не остановимся на вершине с ключом X или пока не достигнем листовой вершины с нулевыми указателями.

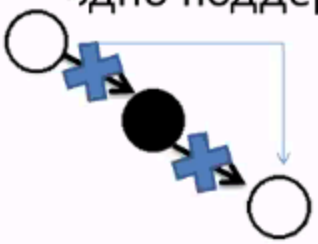
Кстати:
Поиск в дереве НИКОГДА НЕ осуществляется перебором (обходом).

Обходом осуществляется лишь распечатка вершин.

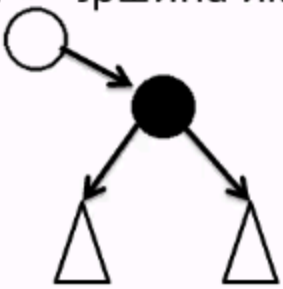
1. Если удаляемая вершина не имеет поддеревьев:

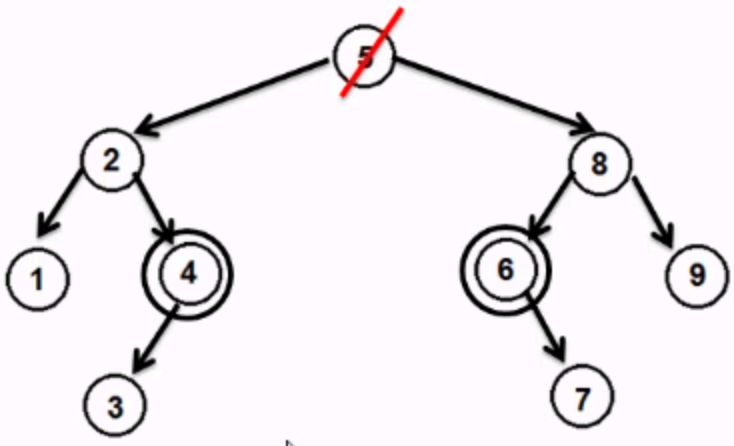
2. Если на удаляемой вершине одно поддерево:





3. Если удаляемая вершина имеет два поддерева:





При удалении вершины 5, чтобы не нарушить поиск, на её место можно поставить либо 4, либо 6.

Правила удаления:

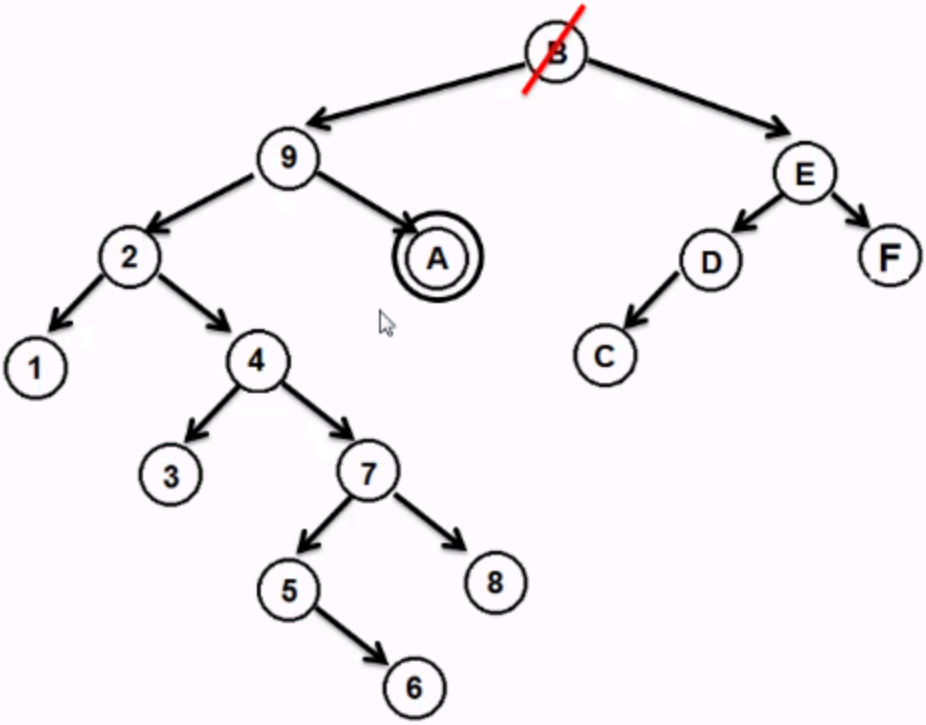
а) На место удаляемой вершины ставится наибольшая вершина из её левого поддерева, т.е. самая правая вершина из левого поддерева, не имеющая правого поддерева.

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.

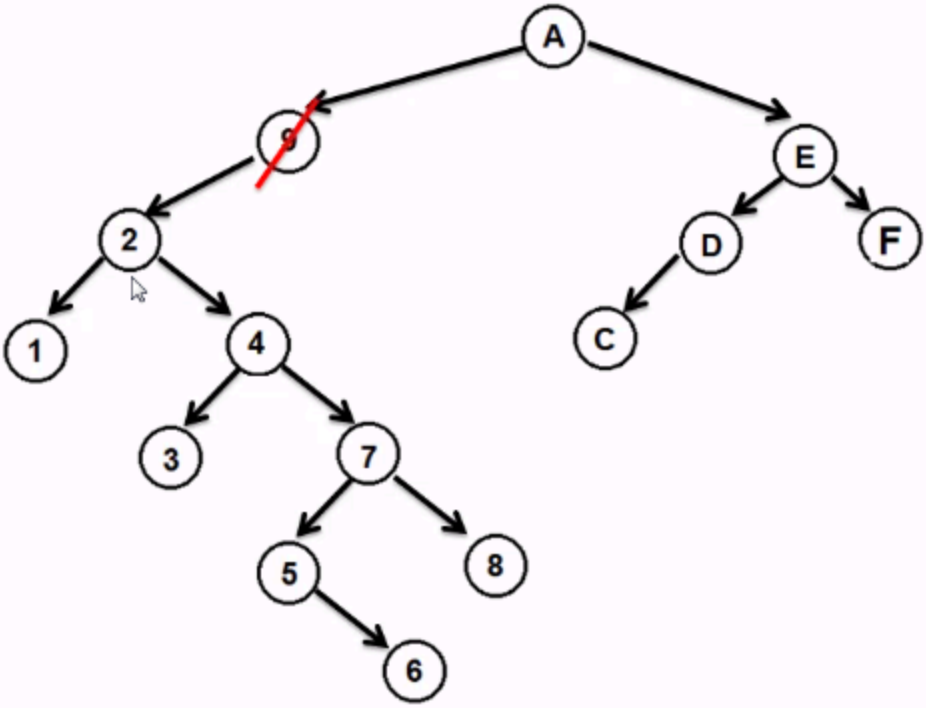
Будем строить алгоритмы на правиле «а»

Пример для домашки:

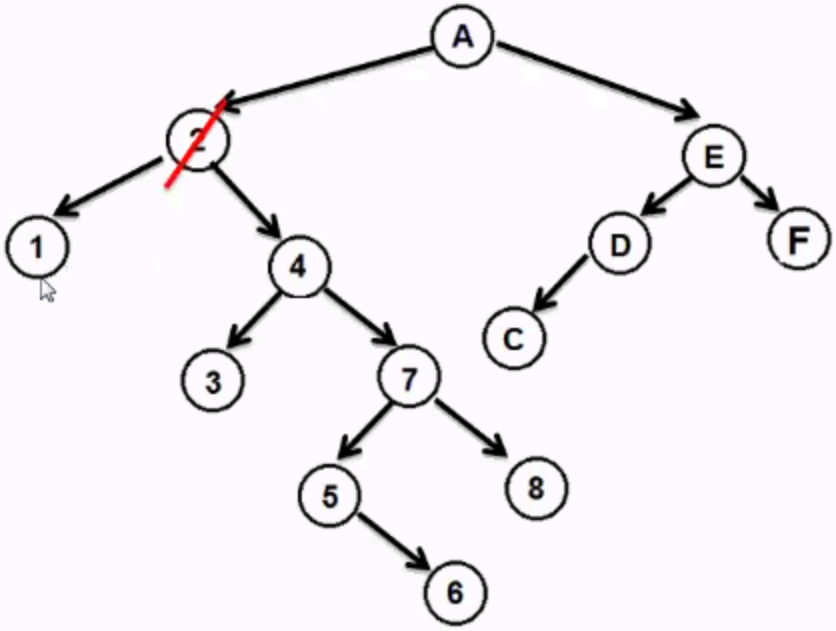
B 9 2 4 1 7 E F A D C 3 5 8 6



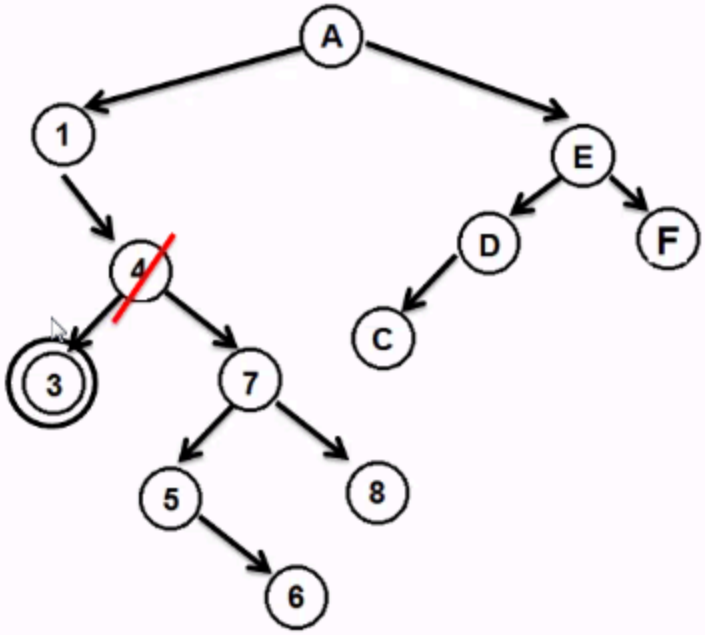
B 9 2 4 1 7 E F A D C 3 5 8 6



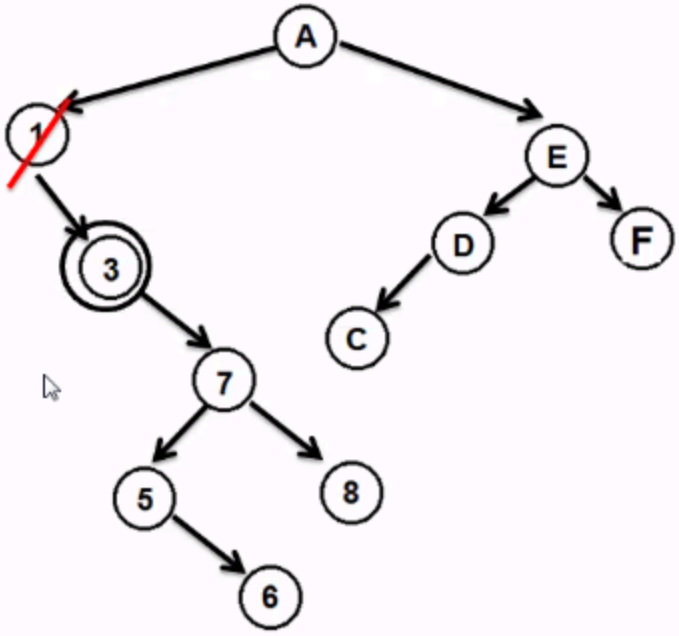
B 9 2 4 1 7 E F A D C 3 5 8 6



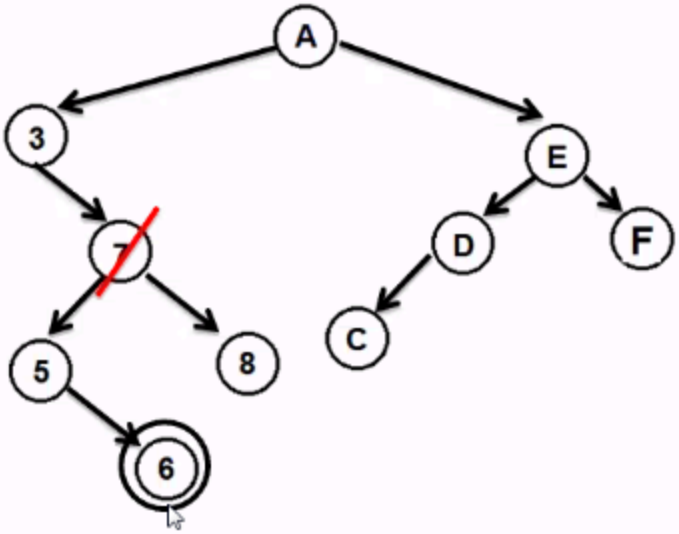
B 9 2 4 1 7 E F A D C 3 5 8 6



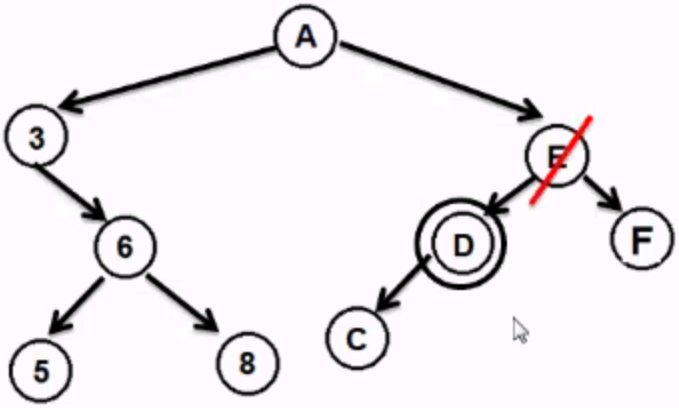
B 9 2 4 1 7 E F A D C 3 5 8 6



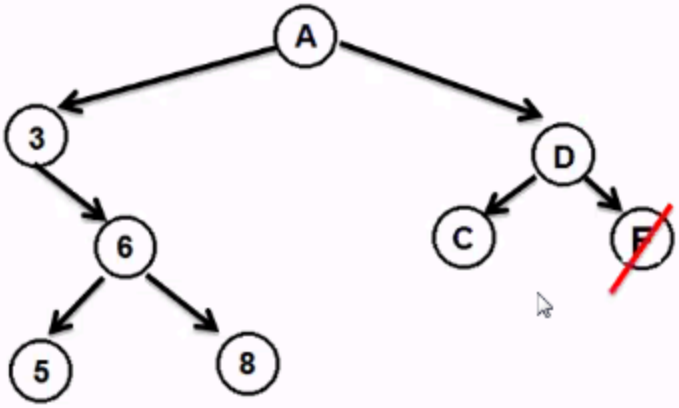
B 9 2 4 1 7 E F A D C 3 5 8 6



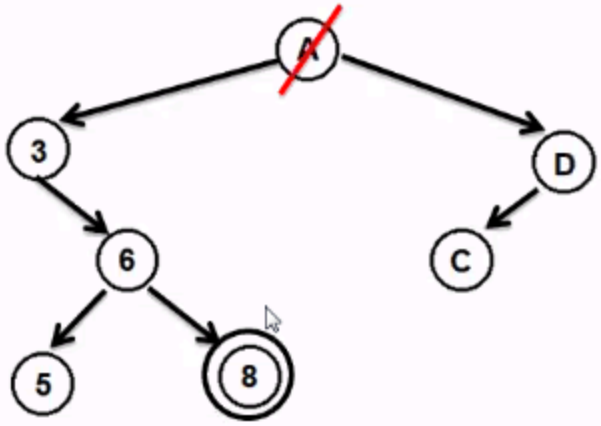
B 9 2 4 1 7 E F A D C 3 5 8 6



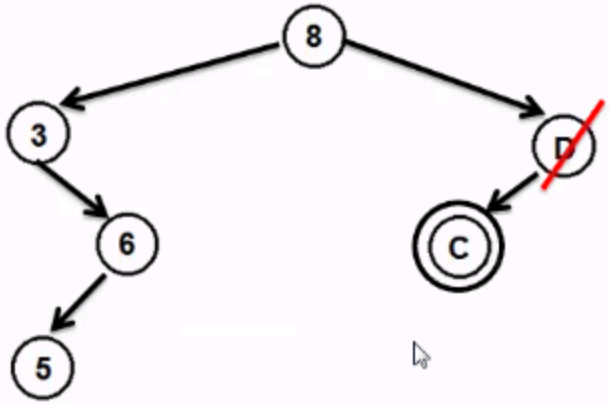
B 9 2 4 1 7 E F A D C 3 5 8 6



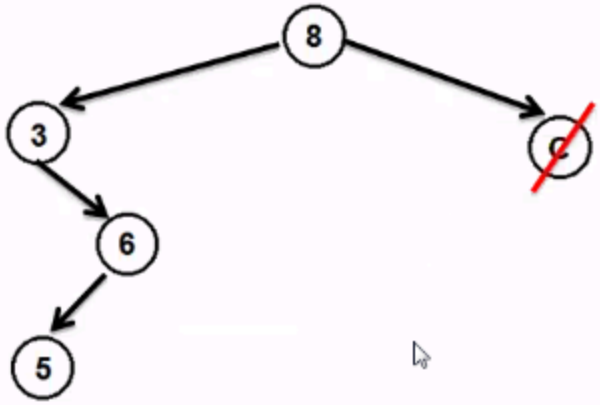
B 9 2 4 1 7 E F A D C 3 5 8 6



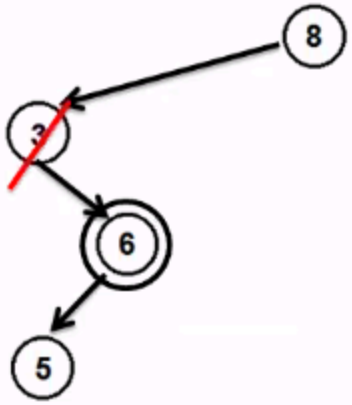
B 9 2 4 1 7 E F A D C 3 5 8 6



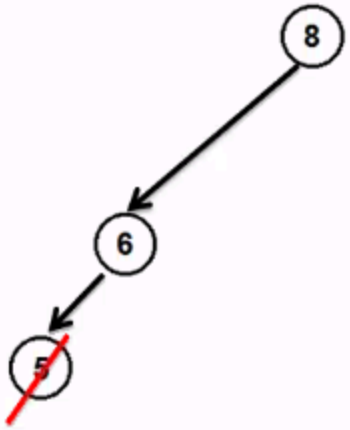
B 9 2 4 1 7 E F A D C 3 5 8 6



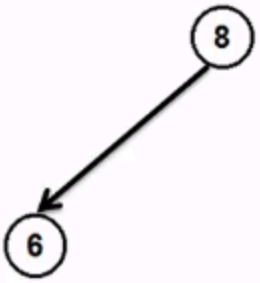
B 9 2 4 1 7 E F A D C 3 5 8 6



B 9 2 4 1 7 E F A D C 3 5 8 6

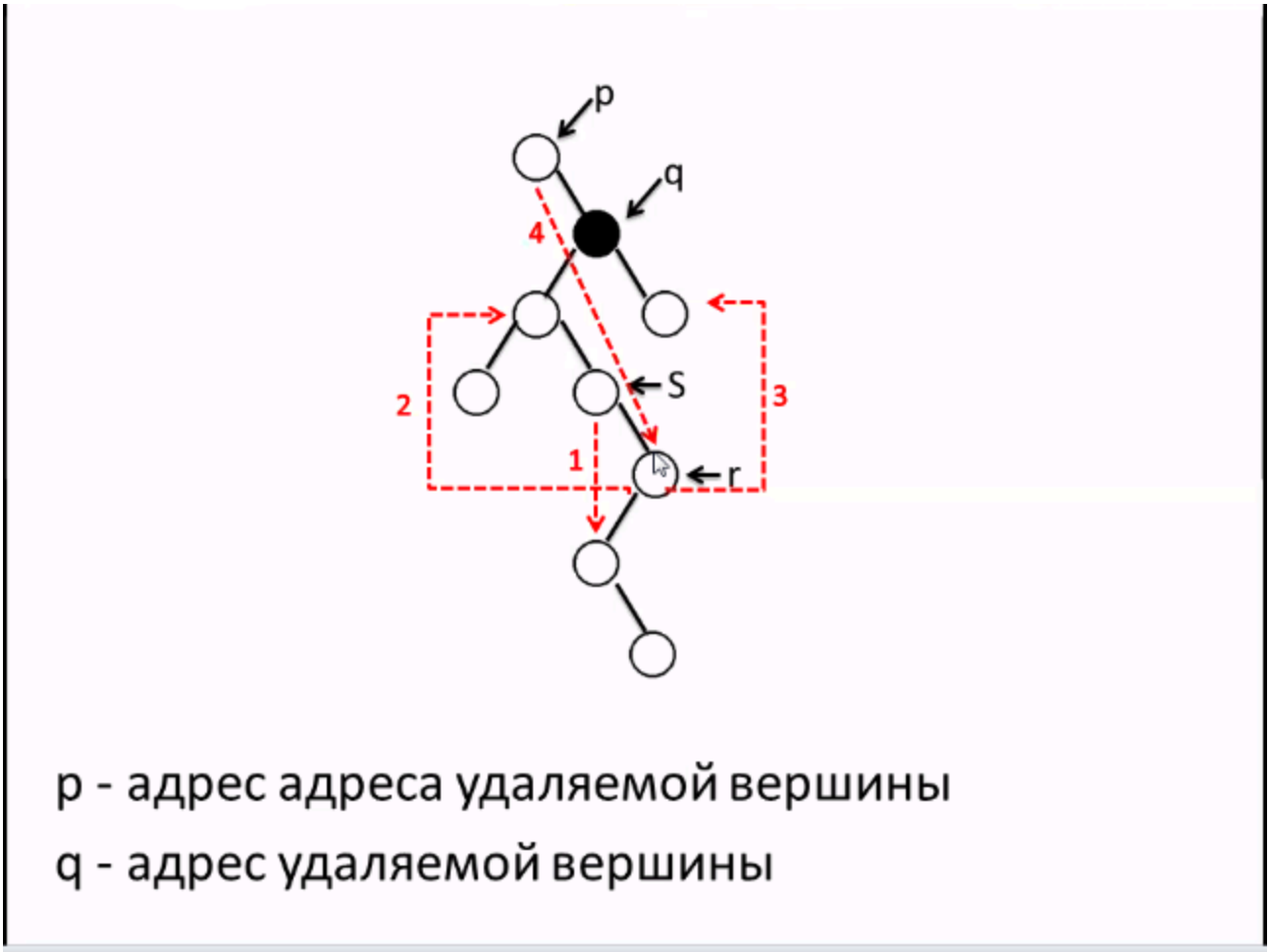


B 9 2 4 1 7 E F A D C 3 5 8 6



B 9 2 4 1 7 E F A D C 3 5 8 6

Продолжение лекции:



r - адрес удаляемой вершины

S - предыдущий родитель r

Удаление (D, *Root)

```
p=&Root
 DO (*p ≠ NULL)     // поиск элемента
     IF (D<(*p)-->Data)  p:=&((*p)-->Left)
     ELSE  IF (D>((*p)-->Data)  p:=&((*p)-->Right)
             ELSE  OD  {данные есть в дереве}
     FI
 OD
IF (*p ≠ NULL)
    q:=*p
    IF (q-->Left=NULL ) *p:=q-->Right;
    ELSE  IF (q-->Right=NULL ) *p:=q-->Left;
           ELSE   /*2 поддерева*/
                  r:=q-->Left ;  S:=q;
```

```
IF (r-->Right = NULL)
        r-->Right := q-->Right ;    (3)
        *p := r;                    (4)
 ELSE
          DO (r-->Right ≠ NULL)
               S := r;  r := r-->Right ;
          OD
        s-->Right := r--> Left ;    (1)
        r-->Left := q-->Left;       (2)
        r-->Right := q-->Right;     (3)
        *p := r;                    (4)
  FI
 FI
free(q)
FI
```