Обход всего дерева

```
void wrtree(ttree *p)
{
 if (p==NULL) return;
  wrtree(p->left);
 cout << p->inf  << "  ";
  wrtree(p->rigth);
}
```

Поиск элемента с заданным ключем

```
void poisktree(ttree *p,int key,
                                    bool &b, int &inf)
 {
if ((p != NULL) && (b != true))
{
    if (p->inf !=key)
    {
   poisktree(p->left,   key, b, inf);
   poisktree(p->rigth, key, b, inf);
    }
```

```
else
    {
        b=true;
        inf=p->inf;
    }
}
return;
}
```

Поиск элемента с максимальным ключем

```
int poiskmaxtree(ttree *p)
{
while (p->rigth != NULL) p = p->rigth;
return p->inf;
}
```

Поиск элемента с минимальным ключем

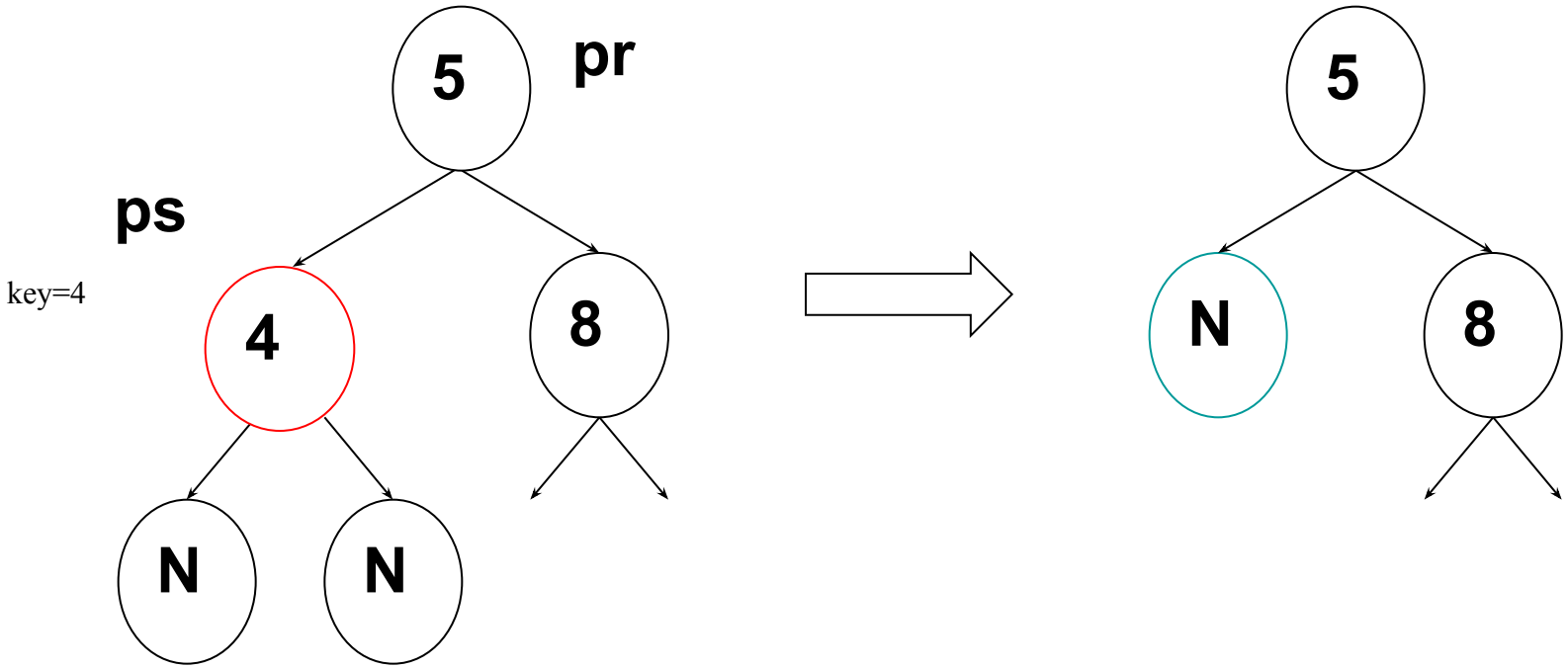
```
int poiskmintree(ttree *p)
{
while (p->left != NULL) p = p-> left;
return p->inf;
}
```

Удаление всего дерева

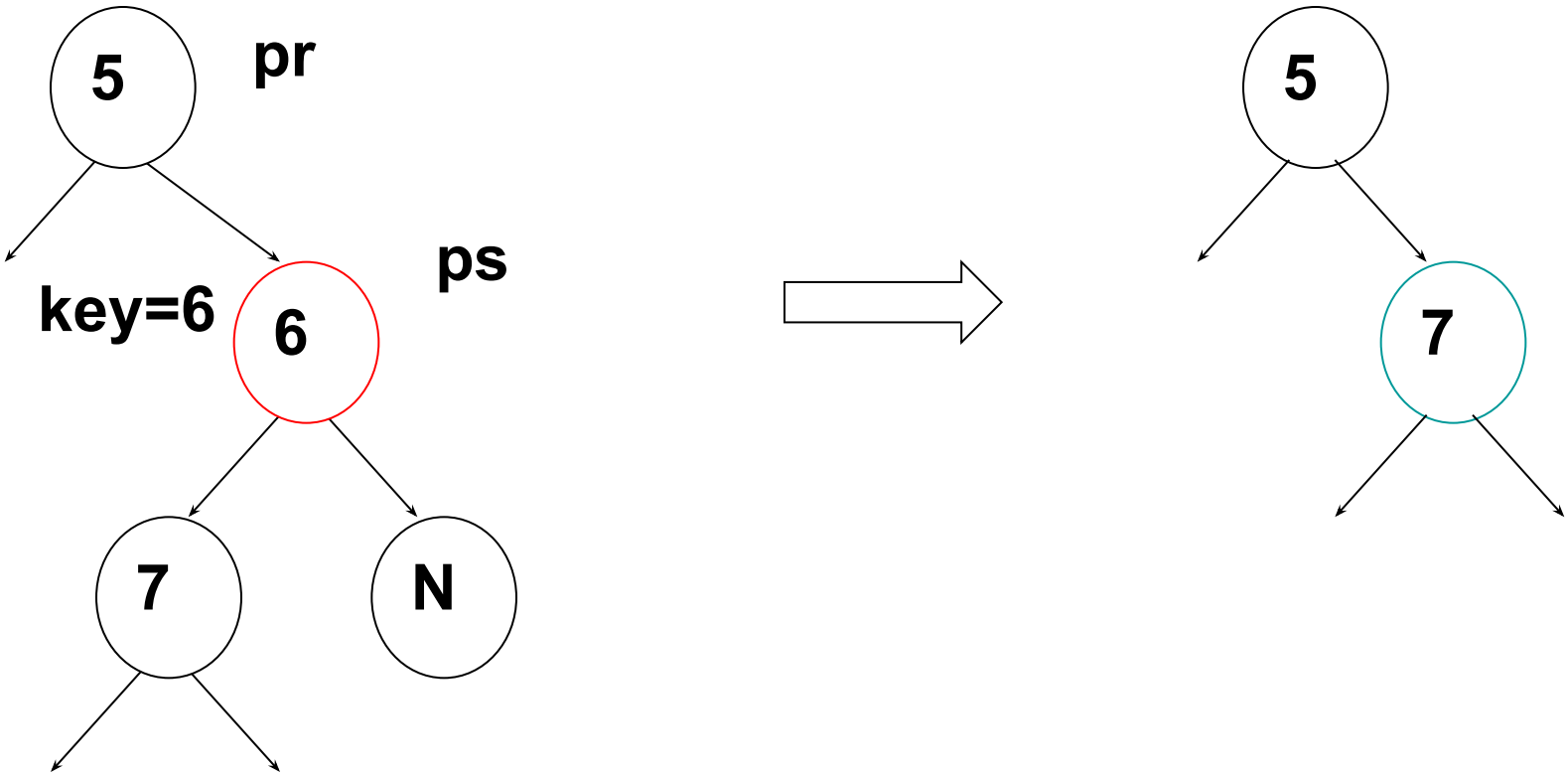
```
ttree *deltree(ttree *p)
{
 if (p==NULL) return NULL;
  deltree(p->left);
  deltree(p->rigth);
  delete(p);
return NULL;
}
```

Удаление элемента с заданным ключем

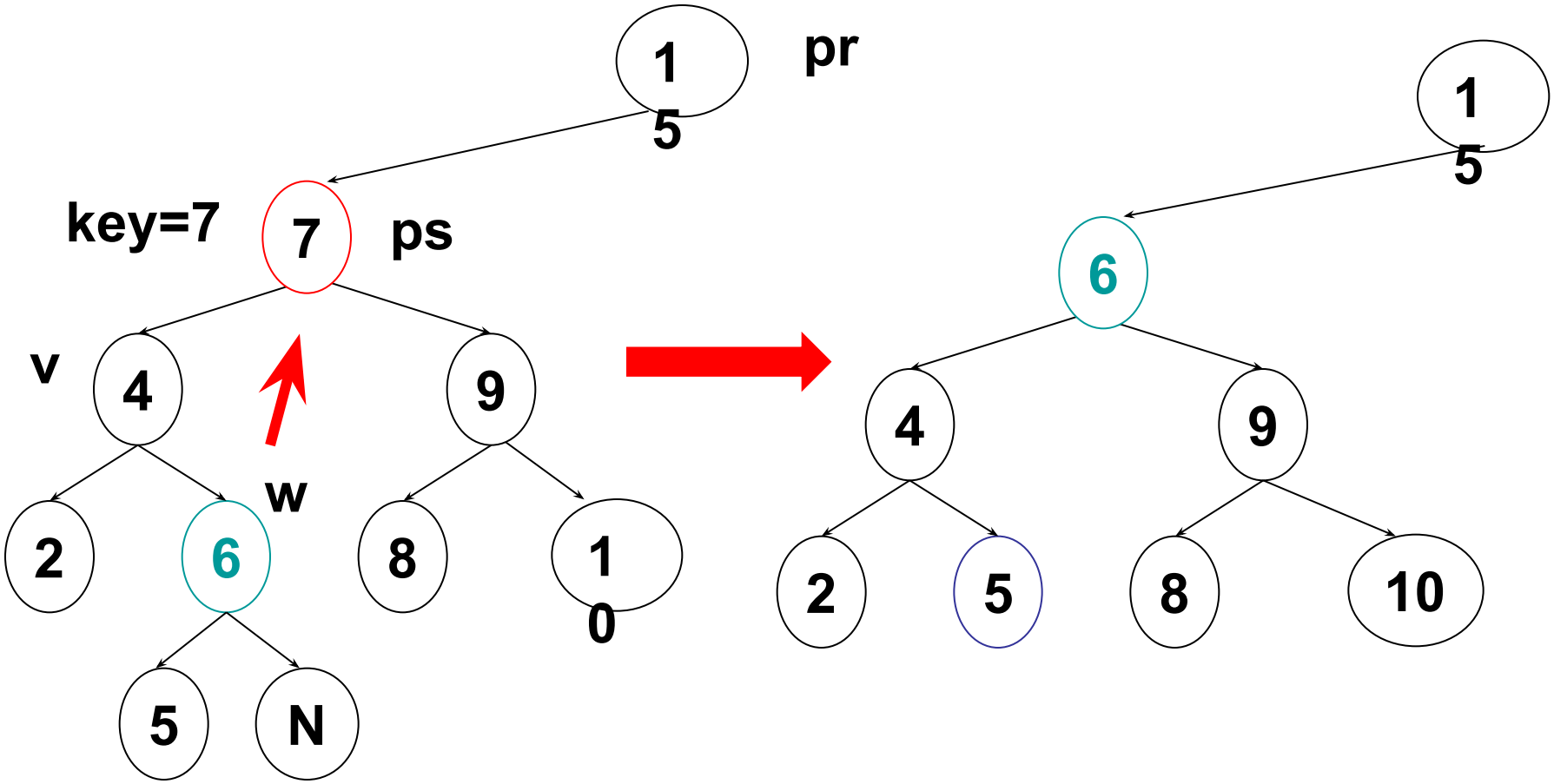
Удаление листа с ключом key:



Удаление узла имеющего одну дочь:



Удаление узла, имеющего двух дочерей



```
ttree *dellist(ttree *proot, int inf)
{
ttree *ps = proot, *pr = proot, *w, *v;
// Поиск удалемого узла
 while ((ps != NULL) && (ps->inf != inf))
{
pr = ps;
 if (inf < ps->inf) ps = ps->left;
                else ps = ps->rigth;
}
 if (ps == NULL) return proot; // Если узел не найден
```

```
// Если узел не имеет дочерей
if ((ps->left == NULL) &&
                              (ps->rigth == NULL))
{

      if (ps == pr) // Если это был последний элемент
     {
     delete(ps);
     return NULL;
     }
```

```
  if (pr->left == ps) // Если удаляемый узел слева
        pr->left = NULL;
  else                // Если удаляемый узел спава
        pr->rigth = NULL;
  delete(ps);
  return proot;
}
```

```
// Если узел имеет дочь только справа
 if (ps->left == NULL)
 {
    if (ps == pr) // Если удаляется корень
          {
        ps = ps->rigth;
        delete(pr);
        return ps;
          }
```

```
    if (pr->left == ps) // Если удаляемый узел слева
        pr->left = ps->rigth;
    else                // Если удаляемый узел спава
        pr->rigth = ps->rigth;
    delete(ps);
    return proot;
}
```

```
// Если узел имеет дочь только слева
 if (ps->rigth == NULL)
 {
    if (ps == pr) // Если удаляется корень
          {
        ps = ps->left;
        delete(pr);
        return ps;
          }
```

```
   if (pr->left == ps) // Если удаляемый узел слева
         pr->left = ps->left;
   else                // Если удаляемый узел спава
         pr->rigth = ps->left;
   delete(ps);
    return proot;
}
```

// Если узел имеет двух дочерей

```
w = ps->left;
if (w->rigth == NULL) // Если максимасальный
                      // следует за ps
    w->rigth = ps->rigth;
```

```
 else    // Если максимасальный не следует за ps
 {
   while (w->rigth != NULL)
   {
    v = w;
    w = w->rigth;
   }
v->rigth = w->left;
w->left = ps->left;
w->rigth = ps->rigth;
 }
```

```
if (ps == pr) // Если удаляется корень
        {
    delete(ps);
    return w;
        }
```

```
    if (pr->left == ps) // Если удаляемый узел слева
        pr->left = w;
    else                // Если удаляемый узел справа
        pr->rigth = w;
    delete(ps);
    return proot;
}
```