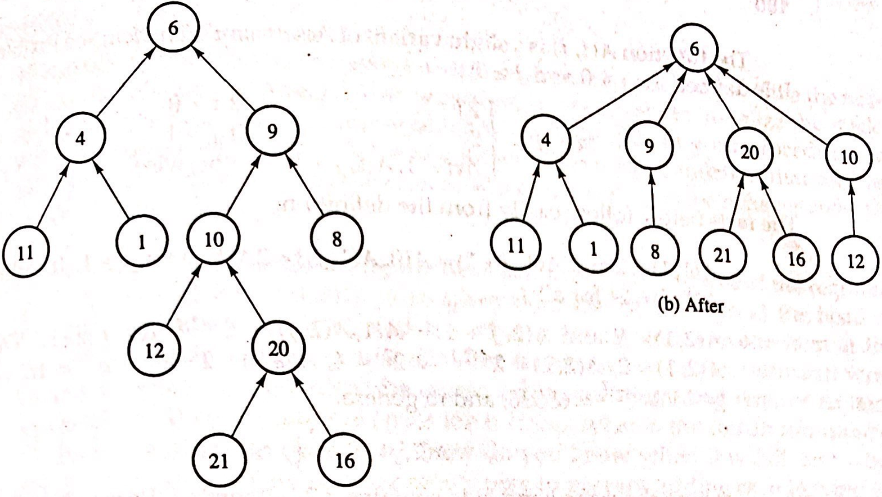
The height of the trees can ... node i in its current tree. W...
that *height*[i] gives the height of node i in its current tree. W...
of a set, *height*[a] therefore gives the height of the correspon...
these are the only heights that concern us. Initially, *height*[i] is...
The procedure *find2* is unchanged, but we must modify *merg*...

procedure *merge3*(a, b)
  {Merges the sets labelled a and b; we assume $a \neq b$}
  if *height*[a]= *height*[b]
  then
    height[a]$\leftarrow$ *height*[a]+1
    set[b]$\leftarrow a$
  else
    if *height*[a]> *height*[b]
    then *set*[b]$\leftarrow a$
    else *set*[a]$\leftarrow b$

If each consultation or modification of an array element coun... operation, the time needed to execute an arbitrary sequence o... *merge3* operations, starting from the initial situation, is in $\Theta($... worst case; assuming n and N are comparable, this is in the exa...

By modifying *find2*, we can make our operations faster s... trying to determine the set that contains a certain object x, w... edges of the tree leading up from x to the root. Once we kno... traverse the same edges again, this time modifying each node e... way so its pointer now indicates the root directly. This techn... *compression*. For example, when we execute the operation *find*... Figure 5.22a, the result is the tree of Figure 5.22b: nodes 20, 10 a... the path from node 20 to the root, now point directly to the roo... the remaining nodes have not changed.

This techni...



(a) Before (b) After

Figure 5.22. Path compression

contents of the root node. However path compression can only reduce the height of a tree, never increase it, so if a is the root, it remains true that *height*$[a]$ is an upper bound on the height of the tree; see Problem 5.31. To avoid confusion we call this value the *rank* of the tree; the name of the array used in *merge*3 should be changed accordingly. The *find* function is now as follows.

function *find*3(x)
  {Finds the label of the set containing object x}
  $r \leftarrow x$
  while $set[r] \neq r$ **do** $r \leftarrow set[r]$
  {r is the root of the tree}
  $i \leftarrow x$
  while $i \neq r$ **do**
    $j \leftarrow set[i]$
    $set[i] \leftarrow r$
    $i \leftarrow j$
  return r

From now on, when we use this combination of two arrays and of procedures *find*3 and *merge*3 to deal with disjoint sets of objects, we say we are using a *disjoint set structure*; see also Problems 5.32 and 5.33 for variations on the theme.

It is not easy to analyse the time needed for an arbitrary sequence of *find* and *merge* operations when path compression is used. In this book we content ourselves [illegible] define two new functions $A(i,j)$ and $\alpha(i,j)$.