Algorithms: Solutions 9

Problem 1
Write a version of depth-first search for identifying the connected components of an undirected graph.

We use a *component* field of a vertex instead of the color. Initially, this field is set to 0. When the algorithm discovers the vertex, it replaces 0 with the component number.

DFS-Components(G)
$k \leftarrow 0$ $\triangleright$ component counter
for each $u \in V[G]$
    do $component[u] \leftarrow 0$
for each $u \in V[G]$
    do if $component[u] = 0$
        then $k \leftarrow k + 1$
            DFS-Visit(u, k)
return k

DFS-Visit(u, k)
$component[u] \leftarrow k$ $\triangleright$ u has just been discovered
for each $v \in Adj[u]$
    do if $component[v] = 0$
        then DFS-Visit(v, k)

Problem 2
Give an algorithm that converts a sorted array $A[1..n]$ into a balanced binary search tree.

The following recursive procedure constructs a balanced tree from a sorted array in linear time; the top-level call to this procedure is Build-Tree($A, 1, n,$ nil).

Build-Tree(A, p, r, y)
if $p > r$
    then return nil
$q \leftarrow \lfloor (p + r)/2 \rfloor$
create a new node x
$key[x] \leftarrow A[q]$
$parent[x] \leftarrow y$
$left[x] \leftarrow$ Build-Tree($A, p, q - 1, x$)
$right[x] \leftarrow$ Build-Tree($A, q + 1, r, x$)
return x