Analysis of Algorithms: Solutions 5

```
                               X
                               X
                               X     X
                               X  X  X
                               X  X  X
                               X  X  X
number of                      X  X  X
homeworks                   X  X  X  X
                            X  X  X  X
                            X  X  X  X
                X     X  X  X  X  X  X
          X     X  X  X  X  X  X  X  X
          -----------------------------
          1  2  3  4  5  6  7  8  9  10
                      grades
```

Problem 1
Give a recursive version of the TREE-INSERT procedure.

The following procedure inserts a new node z into a subtree rooted at a node x. Initially, x must be the root of a tree; that is, in order to insert z into a tree T, we make the top-level call RECURSIVE-INSERT(root$[T], z$). For simplicity, we assume that the tree is *not* empty.

RECURSIVE-INSERT(x, z)
if $key[z] < key[x]$ and $left\text{-}child[x] \neq$ NIL
    then RECURSIVE-INSERT$(left\text{-}child[x], z)$
if $key[z] \geq key[x]$ and $right\text{-}child[x] \neq$ NIL
    then RECURSIVE-INSERT$(right\text{-}child[x], z)$
$parent[z] \leftarrow x$
if $key[z] < key[x]$
    then $left\text{-}child[x] \leftarrow z$
    else $right\text{-}child[x] \leftarrow z$

Problem 2
Suppose we apply the CONNECTED-COMPONENTS algorithm to an undirected graph G, with vertices $G[V] = \{a, b, c, d, e, f, g, h, i, j, k\}$, and its edges $E[G]$ are processed in the following order: $(d, i), (f, k), (g, i), (b, g), (c, e), (i, j), (d, k), (b, j), (d, f), (g, j), (a, e)$. Using Figure 22.1 in the textbook as a model, illustrate the steps of CONNECTED-COMPONENTS on this graph.

The final result of applying CONNECTED-COMPONENTS is the following three sets:

$$\{a, c, e\} \quad \{b, d, f, g, i, j, k\} \quad \{h\}$$

Problem 3
Write pseudocode for MAKE-SET, FIND-SET, and UNION, using the linked-list representation of disjoint sets. UNION must always append the shorter list to the longer one.

We use four fields for each element x of a linked list:

> *next*$[x]$: pointer to the next element of the list; NIL if x is the last element
> *rep*$[x]$: pointer to the set representative, that is, to the first element of the list
> *last*$[x]$: if x is the first element of a list, then this field points to the last element
> *size*$[x]$: if x is the first element, then this field contains the size of the list

If x is not the first element of a list, then the algorithms do *not* use its *last* and *size* fields, and the information in these fields may be incorrect.

MAKE-SET(x)
$next[x] \leftarrow$ NIL
$rep[x] \leftarrow x$
$last[x] \leftarrow x$
$size[x] \leftarrow 1$

FIND-SET(x)
return $rep[x]$

UNION(x)
if $size[rep[x]] > size[rep[y]]$
  then APPEND$(rep[x], rep[y])$
  else APPEND$(rep[y], rep[x])$

APPEND(x, y)
$next[last[x]] \leftarrow y$
$size[x] \leftarrow size[x] + size[y]$
$z \leftarrow y$
while $z \neq$ NIL $\triangleright$ change the *rep* pointers in the second list
  do $rep[z] \leftarrow x$
    $z \leftarrow next[z]$

Problem 4
Suppose that you are using a programming language that allows only integer numbers and supports three operations on them: addition, subtraction, and multiplication. Write an *efficient* algorithm DIVIDE(n, m) that computes $\lfloor n/m \rfloor$, where n and m are positive integers.

Simple algorithm
The following brute-force computation takes $\Theta(\lceil n/m \rceil + 1)$ time and $\Theta(1)$ space.

SIMPLE-DIVIDE(n, m)
$ratio \leftarrow 0$
while $ratio \cdot m \leq n$
  do $ratio \leftarrow ratio + 1$
return $ratio - 1$

Fast algorithm
The following recursive algorithm runs is $\Theta(\lg\lceil n/m\rceil + 1)$ time. The space complexity is *not* constant: the algorithm requires $\Theta(\lg\lceil n/m\rceil + 1)$ memory for the stack of recursive calls.

FAST-DIVIDE(n, m)
if $n < m$
    then return 0
ratio $\leftarrow 2 \cdot$ FAST-DIVIDE$(n, 2 \cdot m)$
if $n - \mathit{ratio} < m$
    then return *ratio*
    else return $\mathit{ratio} + 1$