Data Structures and Algorithms in Java™

Sixth Edition

Michael T. Goodrich
Department of Computer Science
University of California, Irvine

Roberto Tamassia
Department of Computer Science
Brown University

Michael H. Goldwasser
Department of Mathematics and Computer Science
Saint Louis University

Study Guide: Hints to Exercises

WILEY

Chapter 5 Recursion

Hints

Reinforcement

R-5.1) Don't forget about the space used by the method stack.

R-5.2) When the algorithm finds a match, does it know where?

R-5.3) This is probably the first power algorithm you were taught.

R-5.4) Be sure to get the integer division right.

R-5.5) You can model your figure after Figure 5.11.

R-5.6) You should draw small boxes or use a big paper, as there are a lot of recursive calls.

R-5.7) Start with the last term.

R-5.8) Process the string from right to left.

R-5.9) You can rely on bitwise operations to interpret n in binary.

R-5.10) You can use two recursive methods that look like BinarySum.

Creativity

C-5.11) The integer part of the base-two logarithm of n is the number of times you can divide by two before you get a number less than 2.

C-5.12) Consider reducing the task of telling if the elements of an array are unique to the problem of determining if the last $n-1$ elements are all unique and different than the first element.

C-5.13) You need subtraction to count down from m or n and addition to do the arithmetic needed to get the right answer.

C-5.14) Define a recurrence equation.

C-5.15) Start by removing the first element x and computing all the subsets that don't contain x.

C-5.16) Consider first the subproblem of moving all but the n^{th} disk from peg a to another peg using the third as "temporary storage."

C-5.17) Output to System.out one character at a time.

C-5.18) Check the equality of the first and last characters and recur (but be careful to return the correct value for both odd- and even-length strings).

C-5.19) Write your recursive method to first count vowels and consonants.

C-5.20) Consider whether the last element is odd or even and then put it at the appropriate location based on this and recur.

C-5.21) Begin by comparing the first and last elements in a range of indices in A.

C-5.22) The beginning and the end of a range of indices in A can be used as arguments to your recursive method.

C-5.23) Check the last element and then recur on the rest of A.

C-5.24) Look for a geometric series.

C-5.25) Recur on the first $n-1$ positions.

C-5.26) View the chain of nodes following the head node as forming themselves another list.

Projects

P-5.27) Review use of the java.io.File class.

P-5.28) Use recursion in your main solution engine.

P-5.29) Consider a small example to see why the binary representation of the counter is relevant.

P-5.30) Note the recursive nature of the problem.