Data Structures and Algorithms in Python

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 4 Recursion

Hints

Reinforcement

R-4.1) Don't forget about the space used by the function stack.

R-4.2) This is probably the first power algorithm you were taught.

R-4.3) Be sure to get the integer division right.

R-4.4) You can model your figure after Figure 4.11.

R-4.5) You should draw small boxes or use a big paper, as there are a lot of recursive calls.

R-4.6) Start with the last term.

R-4.7) Process the string left to right.

R-4.8) Look for a geometric series.

Creativity

C-4.9) Consider returning a tuple, which contains both the minimum and maximum value.

C-4.10) 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-4.11) Consider reducing the task of telling if the elements of a sequence are unique to the problem of determining if the last $n-1$ elements are all unique and different than the first element.

C-4.12) You need subtraction to count down from m or n and addition to do the arithmetic needed to get the right answer.

C-4.13) Define a recurrence equation.

C-4.14) 1

C-4.15) Start by removing the first element x and computing all the subsets that don't contain x.

C-4.16) You can use syntax `print(ch, end='')` to print one character `ch` at a time, without extraneous spaces.

C-4.17) 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-4.18) Write your recursive function to first count vowels and consonants.

C-4.19) Consider whether the last element is odd or even and then put it at the appropriate location based on this and recur.

C-4.20) Begin by comparing the first and last elements in a range of indices in A.

C-4.21) The beginning and the end of a range of indices in S can be used as arguments to your recursive function.

C-4.22) You can rely on bitwise operations to interpret n in binary.

Projects

P-4.23) Review use of the `os` module.

P-4.24) Use recursion in your main solution engine.

P-4.25) Consider a small example to see why the binary representation of the counter is relevant.

P-4.26) Note the recursive nature of the problem.

P-4.27) Review use of the other methods of the `os` module.