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 5 Array-Based Sequences

Hints

Reinforcement

R-5.1) Although the speed may differ, the asymptotics should be similar.

R-5.2) Keep track of the maximum data size thus far.

R-5.3) You might want to make the list relatively large before you begin to remove entries.

R-5.4) Note that an index such as -3 is equivalent to the traditional index $n-3$ for a list of length n.

R-5.5) Now we are charging more for the growing, so we need to store more cyber-dollars with each element. Calculate the number needed for growing and this will help you determine the number you need to save, which in turn tells you one less than you need to charge each push.

R-5.6) You are going to need two (non-nested) loops.

R-5.7) You don't need to sort A.

R-5.8) Do the observed results match what you expected?

R-5.9) The alphabets for most alphabet-based languages are included in the Unicode character encoding standard.

R-5.10) Review the list comprehension syntax from Section 1.9.2.

R-5.11) You should use nested loops.

R-5.12) Consider how to build a list of subtotals, one for each nested list.

Creativity

C-5.13) Use list comprehension to make an initial list of a specific size.

C-5.14) Consider randomly shuffling the deck one card at a time.

C-5.15) Consider how much cyber "money" is saved up from one expansion to the next.

C-5.16) The existing _resize method can be used to shrink the array.

C-5.17) You can predict precisely when a resize occurs during this process, and take the sum of those costs.

C-5.18) Apply the amortized analysis accounting technique using a monetary accounting scheme with extra funds for both insertions and removals.

C-5.19) Consider, after any resize takes place, how many subsequent operations are needed to force another resize.

C-5.20) Try to oscillate between growing and shrinking.

C-5.21) See Code Fragment 5.4 for an example use of Python's `time` module.

C-5.22) You might design an experiment similar to that in Code Fragment 5.4.

C-5.23) See Code Fragment 5.4 for an example use of Python's `time` module.

C-5.24) Use the `time` module together with loops such as those given on page 205.

C-5.25) You must be very careful if modifying a list at the same time that you loop through its elements!

C-5.26) It might help to sort B.

C-5.27) You might wish to use an auxiliary array of size at most $4n$.

C-5.28) Argue why you have to look at all the integers in L.

C-5.29) Be sure to allow for the case where every pair (x,y) in A and every pair (y,z) in B have the same y value.

C-5.30) You should be able to achieve $O(n)$ time.

C-5.31) Recall that `binary_sum` from Section 4.4.2 computes the sum of numbers in a one-dimensional list.

Projects

P-5.32) The entries $A[i][j][k]$ and $B[i][j][k]$ are the ones that need to be added.

P-5.33) Matrix addition is defined so that if $C = A + B$, then $C[i,j] = A[i,j] + B[i,j]$. Matrix multiplication is defined so that if $C = AB$, where A is a $c \times d$ matrix and B is a $d \times e$ matrix, then $C[i,j] = \sum_{k=0}^{d} A[i,k]B[k,j]$. That is, C is a $c \times e$ matrix.

P-5.34) You will probably need separate encrypt and decrypt mappings for the upper- and lower-case characters.

P-5.35) The original `CaesarCipher` implementation was already effectively a substitution cipher, with a specifically chosen encoder pattern.

P-5.36) If you get the constructor to use the correct encoder string, everything else should work.

P-5.37) A good way to generate a random encryption array is to start with the alphabet array. Then for each letter in this array, randomly swap it with some other letter in the array.