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 3 Fundamental Data Structures

Hints

Reinforcement

R-3.1) Use a calculator to aid in the arithmetic.

R-3.2) You have to have your random number select a random index in the array, so be sure to keep track of the number, n, of entries in the array and do not index past index $n-1$.

R-3.3) The alphabets for most alphabet-based languages are included in the Unicode character encoding standard.

R-3.4) You may want to add a new instance variable to track if the game has been completed.

R-3.5) Make the modification in the code and test it.

R-3.6) It is okay to have an algorithm running in linear time.

R-3.7) There exists a one-line solution.

R-3.8) Consider a combined search from both ends. Also, recall that a link hop is an assignment of the form ”p = p.getNext( );” or ”p = p.getPrev( );”.

R-3.9) Use a loop to traverse the list while counting.

R-3.10) You need to keep track of where you start or your method will have an infinite loop.

R-3.11) Do not include the sentinels in the count.

R-3.12) Carefully relink existing nodes.

R-3.13) Recall that a two-dimensional array in Java is really a one-dimensional array such that each entry is itself a reference to a one-dimensional array.

R-3.14) Recall the ways for doing array copying in the java.util.Arrays class.

R-3.15) You can rely on the size variable to walk the correct number of steps when traversing the lists.

R-3.16) The sentinels are irrelevant to the equivalence of two lists.

Creativity

C-3.17) You don’t need to sort A.

C-3.18) It might help to sort B.

C-3.19) Add items at the “end” of the contiguous run of objects in the array. For removing an object, consider first swapping it with the object at index $n-1$.

C-3.20) Imagine what would happen if $\mathsf{a} = 1$.

C-3.21) Recall the definition of the java.util.nextInt method and note that one of the values returned has a special property with respect to multiplication.

C-3.22) Randomly choose the first element, then the second, and so on.

C-3.23) You might want to consider using a two-dimensional array.

C-3.24) The entries $A[i][j][k]$ and $B[i][j][k]$ are the ones that need to be added.

C-3.25) This concatenation operation need not search all of L and M.

C-3.26) Splice the end of L into the beginning of M.

C-3.27) Performing the swap for a singly linked list will take longer than for a doubly linked list.

C-3.28) Consider changing the orientation of links while making a single pass through the list.

C-3.29) Try to find a matching alignment for the first node of one list.

C-3.30) You are going to have to keep two cursors and count around L.

C-3.31) Adjust the constructor to properly initialize the sentinel.

C-3.32) Blend techniques seen in the existing CircularlyLinkedList and DoublyLinkedList.

C-3.33) You will need to add a prev reference to the node, and maintain it properly whenever the list changes.

C-3.34) Make sure to properly link the new chain of nodes.

C-3.35) Make sure to create new sentinel nodes.

Projects

P-3.36) 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-3.37) You should keep track of the number of game entries explicitly.

P-3.38) You should keep track of the number of game entries explicitly.

P-3.39) You will probably need separate encrypt and decrypt arrays for the upper- and lower-case characters.

P-3.40) The original CaesarCipher implementation was already effectively a substitution cipher, with a specifically chosen encoder pattern.

P-3.41) If you get the constructor to use the correct encoder string, everything else should work.

P-3.42) 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.